Low-Cost Soft Sensors and Robots for Leak Detection in Operating Water Pipes

by

You Wu

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2018

Signature redacted

Author ....
Department of Mechanical Engineering
May 10, 2018

Signature redacted

Certified by ....
Kamal Youcef-Toumi
Professor, Department of Mechanical Engineering
Thesis Supervisor

Signature redacted

Accepted by ....
Rohan Abeyaratne
Chairman, Department Committee on Graduate Theses

Low-Cost Soft Sensors and Robots for Leak Detection in Operating Water Pipes

by

You Wu

Submitted to the Department of Mechanical Engineering
on May 10, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Mechanical Engineering

Abstract

Every day, about 20% of the clean water produced in the world is lost due to pipe leaks. Due to limitations in available technologies, most of the leaks are either not found, or found too late. Every year, there are 240,000 water pipe breaks in the US, and many of them cause sinkholes and other severe damage to the infrastructure. Water utilities need methods for detecting and locating such leaks before they become big breaks, so that they can perform preventative maintenance. This is to save water and protect infrastructure. This thesis presents the design, analysis, fabrication and field test validations of such a solution. I developed soft robots for early detection of leaks in water pipes when the water service is on. This work introduces four key contributions: (1) Design, fabrication and field validations of soft robots for operating water pipes (2) Design, fabrication and field validations of a tactile sensor for detecting leaks in operating water pipes (3) Differentiate leaks from false positives with a low-cost soft bending angle sensor (4) A practical, minimalism approach to the in-pipe localization, specifically for soft robots.

The results are validated in simulations, lab, and field experiments. Those sensors and robots are designed to be low-cost and scalable. They are fabricated with ordinary material with ordinary tools. It is a sub-500-dollar solution to a multi-billion-dollar water and infrastructure problem.

Thesis Supervisor: Kamal Youcef-Toumi
Title: Professor, Department of Mechanical Engineering

Acknowledgments

First, I want to thank my family for bringing me to who I am now. Grown up in a small city named Changzhou in China, I have a wonderful family that constantly encouraged me to think big. My parents made the tough call to send their only child across the world to the US, and supported me through my undergraduate education at Purdue University and then graduate study at Massachusetts Institute of Technology. I am deeply grateful for the opportunities they provided me and the loneliness they endured for all these years. I feel very luck to have the best family in the world.

I would like to express my deepest gratitude to my advisor Prof. Kamal Youcef-Toumi. During my six-year-long journey at MIT, Kamal mentored me in both my research and my life. He guided me to seek orthogonal solutions to challenging problems, aim for disruptive changes instead of incremental improvement. He taught me to keep practical applications in my mind while doing research and development. Kamal also provided me with enough freedom to go beyond being a researcher and explore entrepreneurship in the greater MIT Community. Moreover, Kamal treated me not only as a student but also a friend. Many times we sit down to exchange ideas and craft story lines for presentations together. In this process, Kamal helped me become a better presenter, but more importantly, Kamal taught me that respect and openness brought people into teams.

I would also like to thank my committee members, Professor Michael Triantafyllou and Prof. Xuanhe Zhao. They were very kind to work with me through my fast-paced committee meeting schedules, but more importantly, they helped shaped my thesis with their expertise. I took one of my first three classes at MIT from Michael. He helped me develop solid understanding of hydrodynamics and underwater vehicle design. Those are the foundation of my thesis work. Prof. Xuanhe Zhao was very generous to share his expertise in soft material robots and especially the hydrogel work in his lab with me. It leads to an elegant solution to one of the hard problems in my thesis.

I would also like to say to my lab members at MIT Mechatronics Research

Lab(MRL) that how lucky I was to have the opportunity to work with you all. Dr. Dimitris Chatzigeorgiou and Dr. Dalei Wu took me onto this journey of leak detection robot development When I first started at MIT. My thesis was built on top of their years of early exploration in the same field. I also want to thank Dr. Yi Wang for his company and help in the many long nights of handcrafting robots and the trip to field test in Virginia. If Dr. Wang were not there, I would have to walk this difficult part of the journey alone. I also truly enjoyed work in teams with David Donghyun Kim, Antoine Neil, Solene Demay, Ali Al-Saibie, Kristina Kim, Micheal Finn Henry, Kyle Saleeby, Crystal Winston, Elizabeth Mittmann, Amy Fang, Xiaotong Zhang, Tyler Okamoto and Yip Fun Yeung and many others. You are all rock stars and wonderful team players. Our administrative assistants, Catherine Hogan and Kate Anderson, took care of me for all the past years. MRL is a wonderful group of people that will always have a place in my heart.

I would also like to show my appreciation to all the industry experts that helped me in developing this thesis. In particular, Mr. Mark Gallagher from Cambridge Water Department. For almost two years, Mark generously provided feedbacks to me whenever I went to talk about water pipes and robots with him. Mark helped me understand the importance of my work to the water utilities: it was just about finding smaller leaks, but find leaks earlier. It was about preventative maintenance. It was a game changer. Mark and his colleagues at Cambridge Water Department represent the customers that my robots, when commercialized, are supposed to provide value to. Their active engagement in the research effectively helped me develop the robots in the right direction.

I sincerely appreciate several industry partners that provided invaluable field test opportunities to this project. The first one is Pipetech LLC in Saudi Arabia. I want to thank Mr. Yousef Senan, Mr. Abbas Amralla, Mr. Mowaffak Midani and their colleagues at PipeTech and Dr. Rached Ben-Mansour, our research collaborator at King Fahd University of Petroleum and Minerals for their support and technical cooperation over the course of one year. It was very generous for the technical staff at PipeTech LLC to devote their time and resources to modify their pipeline for testing our robot. The second one is Wise County Public Service Authority (PSA) in Virginia, US. They supported the first-ever pilot test of this technology in the real underground water mains. Many hardworking people at Wise County PSA made tremendous efforts to ensure the success of this project. In particular, we appreciate Alan Harrison for his continuous support since our first phone call. Alan did an amazing job in putting all the people and resources together for this project. Alan and his crews at PSA, such as Barry Meade, Steve Jenkins, Neal Smith and Shane Clark, prepared the test sites, provided important technical support, and they brilliantly solved all the operational issues in the field on the spot. I enjoyed working with Alan and Wise County PSA. I also appreciate Appalachian Regional Commission (ARC) and MIT Lincoln Laboratory (MITLL) for connecting MIT academics with operational end-users. We specifically acknowledge Chris Brazell at ARC and Andrew Weinert at MITLL. They connected my team with Wise County PSA and created this opportunity to test a newly developed technology in solving real problems.

I am very grateful to the research sponsors that made my work at MIT possible. I thank the Center for Clean Water and Clean Energy at MIT and King Fahd University of Petroleum and Minerals, Saudi Arabia for the financial support to my project in the first four years. I also thank Abdul Latif Jameel World Water and Food Security Lab at MIT (J-WAFS) and Skolkovo Institute of Science and Technology(Skoltech) for their fundings and the opportunities to start new projects between 2016 and 2017. I also thank Banxin Corporation in China for supporting a new phase of research in our lab during my last year at MIT.

I want to conclude with the encouraging words from our department head at MIT Mechanical Engineering, Prof. Gang Chen, "Make a difference. Live the dreams. Relish the adventure. Conquer the unknown. Make your mark. Stand strong." To my family and my friends, thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Water leakage is Not Just a Resource Problem

Access to clean water is one of the most challenging problems facing the humanity. The rapid development and deployment of the next generation of water related technologies is key to solving the issue of clean water for the expanding world population under the pressure of climate change. It is predicted that by 2050 about 64% of the developing world and 86% of the developed world will be urbanized [7]. The United Nations also recently projected that nearly all global population growth from 2017 to 2030 will be absorbed by cities, about 1.1 billion new urbanites over the next 13 years [8]. In urban environments, water leakage is at the forefront of the issues within the water access and distribution system.

Every day, the underground water distribution pipe systems in the world loses 20% of its clean water supply due to leaks [9]. Many of those leaks are either not found, or found too late. When most leaks are found, they would have already developed into pipe breaks that cause sinkholes in the streets and severe damage the surrounding infrastructure. Every year, there are about 240,000 accidents as such in the US [10], causing billions of dollars in property damages. When occurred in urbanized areas, each pipe break could cost the local water utility an average of USD 200,000 of property damages and repair expenses [11]. The consequences are much more than the financial loss to the water utility; they also include the intangibles such as water service interruptions to residents and business, blockage of traffic, etc. In comparison, if those leaks were detected before they become big problems, they could be fixed with controlled excavations that typically cost USD 20,000 each [11]. To water utilities, this means a 90% reduction in the financial loss, the prevention of public safety incidences, the savings of millions of gallons of water, and overall a more consistent water service.

The total effect of water leaks is even more than the loss of precious water resources and infrastructure damage. Wetland and wildlife preservation groups reported that leaks force communities to draw more water from local bodies of water than they need, and thus accelerate the decline of local wetland ecosystems. City officials in developing nations said that the poor often suffer the most from water shortages, and water leaks make it even more difficult to secure their access to clean water, a basic human right. Leaks also contaminate the water in the pipes and threaten consumers' health. Policy scholars report that leaks kill economic opportunities for developing nations. Talents, investors and companies turn many cities in developing countries away because they do not have a supportive infrastructure such as a reliable water service. Consequently, these cities lose global competitiveness, suffering from slower economic growth and less funding for infrastructure improvement. This is a downward spiral toward the worst.

1.2 Gaps in Existing Technologies: Early, Accurate, Consistent

Water utilities commonly use a combination of nightline and acoustic leak detection to find leaks. Nightline refers to the lowest hourly water flow rate in a part of the water pipe network, and it is typically measured at 2am in the morning every day. This monitoring is enabled by connected water meters, commonly known as Automated Meter Reading (AMR). If nightline increased drastically in one day and stay high for the following days, it indicates a high probability of leaks in the area. Given a slow sampling rate of once per day and relatively low signal to noise ratio due to random resident water consumption, water utilities only had success in identifying relatively big leaks. Moreover, this method indicates zones of possible leak but does not pinpoint leaks.

After the zone of leaks are identified, water companies will send in technicians to use acoustic leak detection tools to locate the leak. The two most common acoustic tools are listeners and signal correlators. Technicians can walk above the underground water pipes and listen for the signature noise generated by leaks. They can also attach two acoustic sensors a few hundreds of feet apart from each other on the same pipeline. They measure the vibrational signal generated by the same leaks in between them. The phase delay in their measurements can be used to calculate the distance of the leaks from those sensors. Those acoustic techniques are non-invasive and easy to use by skilled technicians, but they suffer from three drawbacks: low sensitivity, lack of consistency, and pipe material dependency. Commonly they can find leaks losing about 10 gallons of water per minute within 10 feet accuracy when it is quiet in the environment. A 10-gallon-a-minute leak is already big; it loses water twice as fast as one person uses in a shower. If there are traffic and noise in the environment, air pockets in the ground near the pipe or a low pressure in the water pipes, the result will have a worse signal to noise ratio. The leaks are either not detected, or the location error can be as much as 100 feet. Last but not the least, those acoustic techniques are developed for metallic pipes, and they are ineffective in pipes made of vibration damping material such as plastics. In UK, China, Mexico, Saudi Arabia and many other regions in the world, half or more of their water pipes are plastic. Those pipes are difficult to maintain because the lack of effective leak detection technology.

In addition to nightline and acoustics, there has been a surge of other new above-ground leak detection solutions from both the industry and academia. Industrialized solutions such as smart sensor networks and aerial imagining can identify zones of possible leak. In a smart sensor network implementation such as Visenti, an MIT-Singapore program spin-off, pressure sensors of high sampling rates are installed all over the water pipes network to monitor the water hammering, the impulse generated by a newly pipe break [12]. This method does not detect the existing leaks. Many leaks grow steadily from small ones into pipe breaks, and they remain undetectable to the smart sensor networks until they break. Aerial leak detection solutions such as Utilis [13], uses radiometers on either planes or satellites to measure ground water content on the received spectral images. They infer leaks when detecting water in the ground, and their performance is affected by rainfall level in the targeted area. In the academia, researchers are also experimenting leak detection in short ranges through measuring the change in conductivity or dialectic properties in the ground due to water leaks[14].

The last category of leak detection solutions are in-pipe robots, and they tend to be difficult to use but produce more accurate result. Because those in-pipe devices can get much closer to the leaks than the above-ground methods do, they are more likely to sense the leaks and pinpoint their locations. The leading in-pipe leak detection robot is Smartball [15], a free-floating device that listens for leaks from inside the pipe. It has two main constraints: size and cost. Smartball can only fit inside pipes of 8 inches in diameter or bigger, while there are more 2, 4 and 6 inches water pipes in the vast water distribution network [16]. Its location is tracked by a series of wireless tracking system placed along the pipeline. On the other hand, tethered robots such as Sahara [17] are easier to use but limited by range. Both Smartball and Sahara use acoustic sensors, and similar to their above-ground counterparts, their performance is poor in plastic pipes. The other in-pipe solutions are vision based, include cabled cameras and cameras on rovers. The technician must be highly trained to spot leaks on the videos. This vision approach is less commonly used in water pipes because it requires shutdown of the water service and empty the pipe in order to get a good visual input.

Effective early leak detection will enable water utilities to actively prevent water pipe breaks, but the existing solutions cannot deliver that. Instead of shower-size leaks, early detection solutions must be able to sense early stage leaks that are a magnitude smaller. In order to be effective, it must also locate leaks fast and accurately. Moreover, as a preventative measure, the leak detection must be carried out when the water service is on. If water service must be turned off for leak detection, it is a major interference to the community as a water pipe break. Lastly, it is not achieved by any existing technology to detect leaks consistently, independent of constraints such as pipe material, in-line pressure, soil condition, and noise in the environment.

1.3 Background Work in New Leak Detection Method

In our group at Mechatronics Research Laboratory, we have been investigating a pressure gradient [18, 19] based in-pipe leak detection method. It addresses the consistency issue faced by the existing leak detection solutions. It can sense leaks in low pressure pipes, pipes of any material and any fluid, and detect leaks independent of any conditions outside the pipe. This method uses a membrane to detect the sharp pressure drop in the pipe at the leak [18]. The sharp pressure drop and the leak flow will draw and pull on the membrane. By measuring the force or motion on the membrane, one can infer the presence of a leak.

There are two main challenges in applying this pressure gradient based leak detection method to real water pipe systems. The first challenge comes from the water flow. Previous implementations were only successfully demonstrated in static air or water pipes [18, 19] . Further experiments showed that none of them could consistently register the leak when water is flowing through the pipe. It is a significant limiting factor to shut down the water service for using this method to find leaks. The second challenge is about false alarms. Obstacles in the pipeline such as dirt, scales, and other irregularities often exert forces on the membrane and trigger false alarms. This issue is unprecedented in acoustics or visual methods.

1.4 Four Proposed Contributions

My goal of this research thesis to develop a practical and consistent early leak detection solution. Building on top of the pressure gradient based in-pipe leak detection method, I will focus on designing the robots and sensors that operate inside real, live water pipes, and differentiate leaks from false alarms. I will also pursue opportunities to collaborate with water utilities to test and validate my robots in real underground water pipes.

Four research contributions are proposed:

1. A soft matter robot for missions in operating underground water pipes. Conventional in-pipe robots are rigid and precise machines, they require convoluted mechanisms to transverse in pipes with obstacles, 90-degree bends and Tee junctions. In this work, a compliant in-pipe robot is designed to address those challenges in a simple but reliable approach, and the robots are validated in field tests.

2. Leak detector with robust performance in dynamic flow conditions. In the previous work, the impact of hydrodynamics on the pressure-gradient based leak detector was not studied. The dynamic water flow rendered the detector ineffective. In this work, the design of a functional leak detector in dynamic fluid flow is studied and validated in field tests. Instead of attempting to minimize the impact of fluid flow on the detector, this design takes advantages of the hydrodynamic effects to improve its sensitivity and consistency.

3. Differentiate leaks from false positives with a low-cost soft bending angle sensor Soft matter tactile sensors are widely used in force sensing. However, in a typical sensor output, effects applied from different directions, for example, normal pressure, shear stress and torque, are always coupled and they could not be differentiated from each other. In this work, I investigate the methods to design single soft matter sensors that can uncouple tension, pressure and bending moments, measure only one of them while rejecting the disturbance from the others. For example, in leak detection applications, leaks bend the sensor down and pull, obtrusions bend the sensor up. I designed and fabricated single piece soft sensor that can tell them apart through measuring the bending direction and angle. Such sensor is constructed with low-cost ordinary material in a low-cost, scalable fabrication process. The outcome is a 1-dollar solution to robustly detect leaks and obtrusions in pipe distinctively at the same time.

4. A practical, minimalism approach to the in-pipe localization, specifically for

soft robots. Existing in-pipe robots are often tracked by external sensor networks or tethers, I propose and evaluate an approach to estimate the robots' path with a minimum number of on-board sensors only. This approach would be much easier to setup in the field, and requires minimum power. I designed the robot so it can identify to the geometric constraints in the active water pipe, such as joints that occurs frequently, with a minimum number of sensors. The localization algorithm fuses three data streams: the identification of those repeating features, the conventional inertia measurement unit outputs and prior knowledge of pipe maps. The algorithm is performed on both field test data and simulated data. The effects of noise and errors in the three data streams on the data fusion process, its limitations and the necessity for tracking with external system are studied in simulation.

Chapter 2

Soft Passive Robot For Operating Water Pipes

2.1 Overview

Before diving into the leak sensors and its algorithms, I am going to present the design and field test validation of the robotic platform for carrying the sensors through water pipes. Deployment, retrieval of an in-pipe device and how it maneuvers at pipe elbows are challenging topics. My solution to these challenges is a soft-body, miniaturized Pipeline inspection Gauge (PIG); it is propelled by the pipe flow and thus cover a long distance with little power consumption. Unlike regular PIGs, it is made of soft material and it can follow the water flow through pipe elbows. Through field tests, the deployment and retrieval procedure for this robot was also demonstrated. The first prototype was successfully tested in a 52-mm-inner-diameter, cast-iron industrial pipe system. Another prototype was successfully tested in an underground 150-mm-diameter, PVC water main.

2.2 Background

A good in-pipe leak sensor is only useful when a mobile platform can carry it through the water pipes. The city water distribution systems commonly consist of small diameter pipes between 50 to 150 mm(2-6 in). There are many T junctions and elbows. They are operating with water flows inside most all the time. The mobile platform must be able to go through small diameter pipe systems with T junctions and elbows, under flow condition. Moreover, the MIT leak sensors use membranes to detect leaks, and those membranes must be kept within a fix distance to the pipe wall in order to detect leaks. The platform then must have position and orientation stability. The existing in-pipe platforms failed to meet both criteria at the same time. On one end of the spectrum, free floating system such as the Smartball [20] can follow the water flow through pipes with elbows. In-pipe swimming robot such as [21] can actively turn at T junctions and elbows. Both systems are small and move in pipe without contacting the pipe walls. However, they are easily affected by the turbulence in the pipe. They cannot maintain the position and orientation of the leak sensor. On the other end, Pipeline Inspection Gauges (PIGs) are flow driven robots, they slide on the pipe walls. They can carry ultrasonic transducers, magnetic flux leakage sensors, and other sensors with similar position and orientation stability requirements, through pipelines [22]. Regular PIGs are rigid and single-piece. Some others are more like trains, having multiple sections connected with joints, such as [23]. Single-piece PIGs cannot make sharp turns around pipe elbows but train-like PIGs can. However, all PIGs have been developed for larger diameter pipes; ones for small diameter pipes are difficult to build.

In-pipe Mobility was the real problem our research group faced. In the development of the leak sensor, our lab partnered with PipeTech LLC in Saudi Arabia, a professional pipeline service company. PipeTech offered their industrial test facility (Fig. 2-1, a 1.5 km (0.93 mile) long, 52 mm (2.05 in) inner diameter, cast iron pipe loop with many elbows, for validating the leak detection technology. To simulate real applications, there would be pressurized water flow in the pipeline during the test. There were no available and effective mobile platforms for such a small diameter and zigzagging pipe system. A design of a mobile platform for those pipe systems is necessary.

Meanwhile, development of soft robotics in recent years provided an inspiration

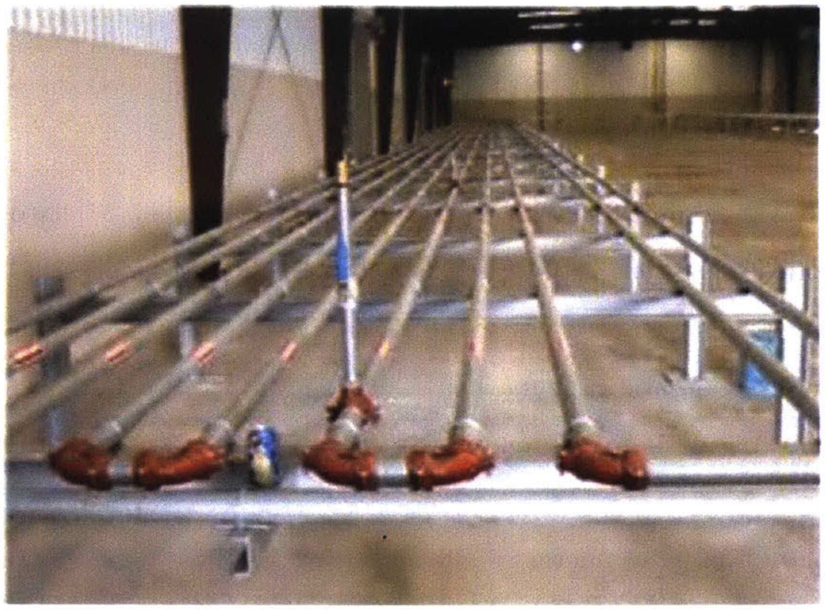


Figure 2-1: A 1.5km long, 52mm inner diameter, cast iron pipe loop for testing pipeline technologies in industrial settings.

for in-pipe systems. Robots made with soft rubber could move while been squeezed or bent [24, 25]. Soft sensors [26, 27] measured conveniently strains in multiple directions. Soft material was also used to build swimming robots that mimic real fishes swimming [28, 29]. Soft material provides many possibilities for building in-pipe systems; it is resilient, deformable, waterproof, easy to tune, and easy to embed electronics.

In this chapter, I present the design of a soft-body robot as an effective solution for carrying sensors through small diameter water pipe systems. It is driven by the pipe flow so it can go a long distance with little power consumption. It is soft and it can follow the water flow through pipe elbows. Simple and effective deployment and retrieval method for this soft-body robot are also developed. The prototype robot successfully carried a leak sensor through the 52mm inner diameter industrial pipe loop at PipeTech LLC as shown in Figure 2-1.

2.3 Robot Design

2.3.1 Soft and Passive

The goal is to design a robot for carrying the MIT leak sensor [30, 31] through a small diameter, complicated water pipe system. Since the leak sensor works in both plastic and metallic pipes, this robot should also be able to work in both type of pipes. The PipeTech's industrial testing pipeline has a single entrance, a single exit and many elbows. The following design requirements applies to this testing pipeline. The robot should be able to

(1) move in a 52 mm inner diameter pipe when there is a pressurized flow in the pipe. The pressure is 2-4 Bar gauge and the flow rate is 0.3-0.7 m/s.
(2) go through T junctions
(3) go through pipe elbows
(4) go through pipes with mild obstacles
(5) maintain the position and orientation stability for the sensor it carries
(6) be untethered and have a range larger than 1.5 km

While designing this robot, many features can be learned from oil pipeline robots known as PIGs. PIGs are flow driven, so it consumes no power for mobility. Given the flow in the pipeline and the range requirement, this robot can be flow driven like regular PIGs. Thus, although powered by batteries, an industrial PIG can go through tens of kilometers of pipeline without recharging. There can be fewer electronic components in the robot, since its power consumption is low and it needs no actuators. This allows the robot to be very compact, a much desired feature for going into small diameter pipes. PIGs are usually of the same diameter as the pipeline, so it moves like a piston in the pipe, with perfect position and orientation stability. Thus a PIG-like robot will be able to satisfy design requirement (1),(5) and (6).

Making the robot out of soft material can help it additionally meet design requirement (2), (3), and (4). A soft material is appealing for its capability to squeeze and bend. In a water pipe, it is common to see dirt, scales and other pipe diameter reduc-

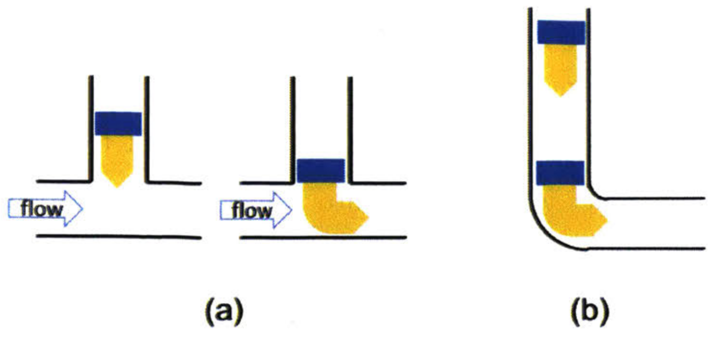


Figure 2-2: Conceptual soft-body robot bends and makes through a T junction and pipe bents. The blue part is the leak sensor and the yellow part is the robot.

tions obtruding the path for the robot. Being able to squeeze through those regions makes the robot more reliable. A soft-body robot can naturally follow the flow and bend to turn around elbows. Moreover, with the correct head design, a soft-body robot can bend at T junctions, as shown in Figure 2-2. In this figure, the yellow robot carries a blue leak sensor. The leak sensor has little adaptability or flexibility. When the system enters the T junction from the vertical branch, its head will touch the bottom of the T junction and bend along the direction of the flow. As the frontal part of the robot bends and aligns with the horizontal pipe, it will pull the leak sensor into horizontal pipe. The head of the robot must facilitate the turn; it should slide in the horizontal pipe, guide the entire system to turn rather than putting a brake on it. In a different case when the robot enters the T junction from the left side of the horizontal pipe and intends to go up to the vertical branch, the robot will not be able to do so without actuation. Even with actuation, if the flow speed in the horizontal pipe is high and the robot enters the T junction with large momentum, it would still have a hard time turning vertical. Thus a soft-body robot can only go through T junctions in certain cases. Thus when using this robot in a pipe system, the places where it can go will be limited by the layout of the T junctions and the pipe flow. However, being able to turn at elbows and T junctions in some cases is already a big leap forward when compared to regular PIGs. Moreover, this limitation makes it easier to predict where this flow driven robot can go.

Thus the concept of a robot is formulated as shown in Figure 2-3. Its main body is soft and can bend to go around elbows and T junctions. It has no actuation and it is flow driven. It is very compact. Electronics will be embedded in the soft body for integrity and waterproofness. It has a solid cap in the front to guide the robot and reduce friction upon contact when it runs into T junctions and elbows.

The robot carries the leak sensor in the back. Details of the leak sensor will be presented in the next chapter. Structurally, it consists of four blue membranes, and four yellow supports and they form a circular pattern when viewed from the back of the robot. The yellow supports are like umbrellas; they are spring loaded and forced to expand. They keep the membrane sensors close the pipe wall.

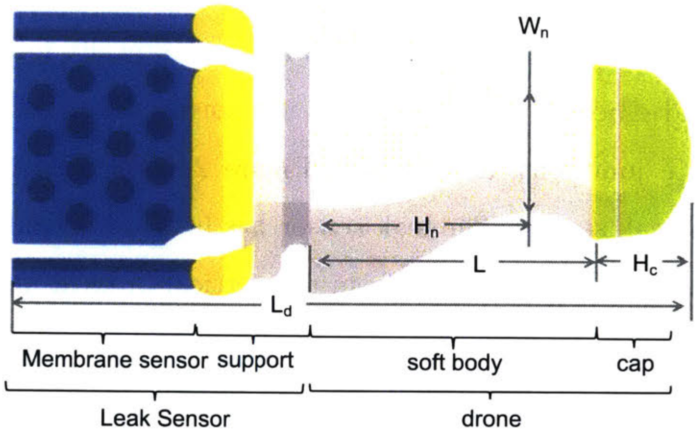


Figure 2-3: Concept of the soft-body robot carrying a leak sensor.

2.3.2 Material and Geometry

The robot's ability to turn is determined by its flexibility and its flexibility is dependent on ts material and geometry. The material choice is first to be addressed. Then the shape factor, L, H_n, and W_n in Figure 2-3 are discussed. From the T junction case shown in Figure 2-2, it can be seen that the length of the robot, L, must be similar to the diameter of the pipe so it can bend in the horizontal pipe before the leak sensor enters. The place that deforms the most easily on the robot is its neck, the thinnest part of its body. Thus the neck location H_n and the neck width W_n affects the robots ability to bend.

A set of experiments were conducted to find the best available soft material for this robot. The Ecoflex and Moldstar product lines from Smooth-on LLC are a wide range of well-documented and easy-to-make silicone rubber material. However, hardness does not exactly transfer into spring constants analytically because the shape of the soft body matters. Six products of adjacent Shore Hardness values were experimented. Half ellipsoidal shaped dummy robots as shown on the left of Figure 2-4 were made for each material. The soft part is 50mm long, 45mm in diameter at the base. On the tip was a rigid cap of 15mm in height. In the test, each soft body was fixed on the base while its tip was being pulled 3 cm to the left with a dynamometer. The steady state force was measured and plotted in Figure 2-4. A low force requirement was preferred, since that translated to low pressure requirement for the pipe flow to push the robot through bends. The softer silicone rubbers of Shore 00-30 and 00-50 hardness required little force to bend, while the harder ones of Shore A 10-20 required more than twice the force to bend. The last one of Shore A-40 could not be bent and thus not plotted. It was also observed that the soft body made of Shore 00-30 rubber would buckle first with an axial force, while the others bend first given the same loading. Buckling is not desired for turning at T junctions. Thus the next easiest-to-bend material, Ecoflex silicone rubber of Shore 00-50 hardness, was chosen for the robot.

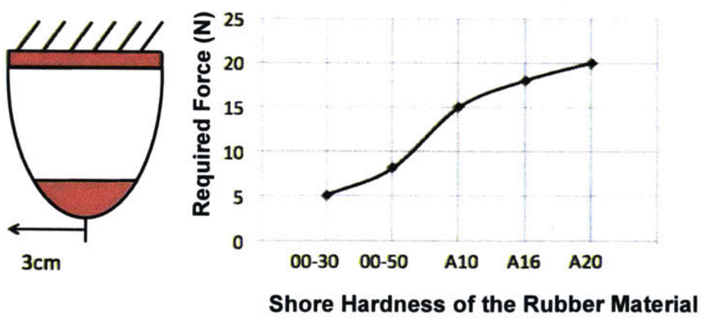


Figure 2-4: Experiment to determine the best available material for robot. Silicone rubbers of different Shore hardness values are made into the same soft body shape and tested in the same way.

Similar experiments were also used to determine a feasible robot geometry. It was a hypothesis that if there was a neck in the geometry, a soft body would always bend at the neck, and the required force to bend would be dependent on the size of the neck. A few trials confirmed that the soft body always bent at the neck. Then a set of experiments were designed to determine the neck width. As shown in the left of Figure 2-5, dummy robots with concave shapes of different width to height ratio were made and tested. The Height of the soft part, H, were all around 50 mm, and the base was 45mm in diameter. The neck was set to be 35mm from the base given the space between the neck and base was needed to contain electronics. The rigid cap was 15 mm high. In the test, each soft body is fixed on the base while its tip is being pulled 2 cm to the left. The steady state force is measured and plotted in Figure 2-5. The first data point was of the same half ellipsoidal soft body from the material test earlier and it is convex. The other three body were concave and they all had much lower bend force requirements. As the neck got thinner, the force required to bend was much lower, and the soft body could buckle before it bent. Buckling is undesired because it can prevent the robot from going around Tee junctions. Moreover, thin neck meant less space in the robot. Thus the median ratio around H:W=2 is chosen for both easy to bend and large space in the body. At this H:W ratio, most deformation during a bend occurred at the neck, and the space between the base and the neck was little affected. If electronics were placed in that space, they would not be squeezed or stretched significantly during a bend.

Neck in the soft body acts like a joint; it allows us to dimension the robot like a

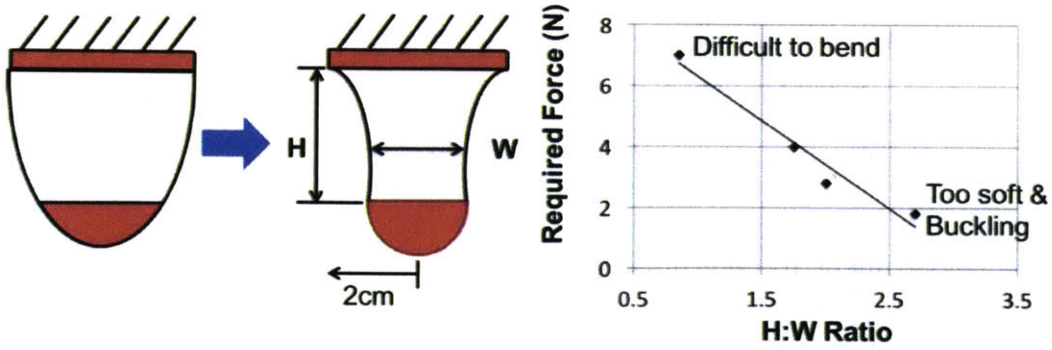


Figure 2-5: Experiment to determine the effect of H:W ratio of the soft body on its bending capability. Four soft bodies of different width to height ratios are made of the same 00-50 silicone rubber.

rigid linkage. In the design requirement, the robot is desired to go around Tee junctions and pipe elbow. The robot's dimension decides if it can meet this requirement. In the case of a multi-module, rigid-body robot, each of its module has to be shorter than the diameter of the pipe in order to go round the Tee junction. This soft robot, being able to bent, can be longer than the pipe diameter. How to determine the range of feasible dimensions for this soft robot?With the neck in the soft body, the soft robot always deforms first at the neck when bending. This neck can then be modeled as the joint, and the robot body can be modeled as rigid linkages as indicated in Figure 2-6. Now we can apply the classic principle from the rigid body robot design in this soft body robot design. The main body part of the robot that contains electronics and other inflexible components, should have a length H_n less than the pipe diameter, which is 52mm in this case. As long as this dimension constraint is met, the robot can maneuver around Tee junctions and 90 degree bends.

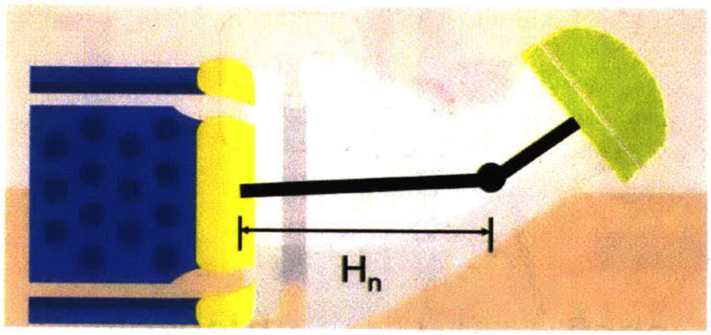


Figure 2-6: Modeling soft body robot as a linkage

2.4 2" Robot and Its Field Test in An Industrial Facility

The leak detection robot was first built and pilot tested in an industrial facility in Saudi Arabia. The field test required the robot to transverse through a 52-mm-inner-diameter, cast-iron industrial pipe system and locate leaks. In this section, the robot prototype and the operation part of the field test are presented in detail. The leak detection results are not presented here but kept for the next Chapter which focuses on the leak sensor.

2.4.1 The 2" Robot Prototype

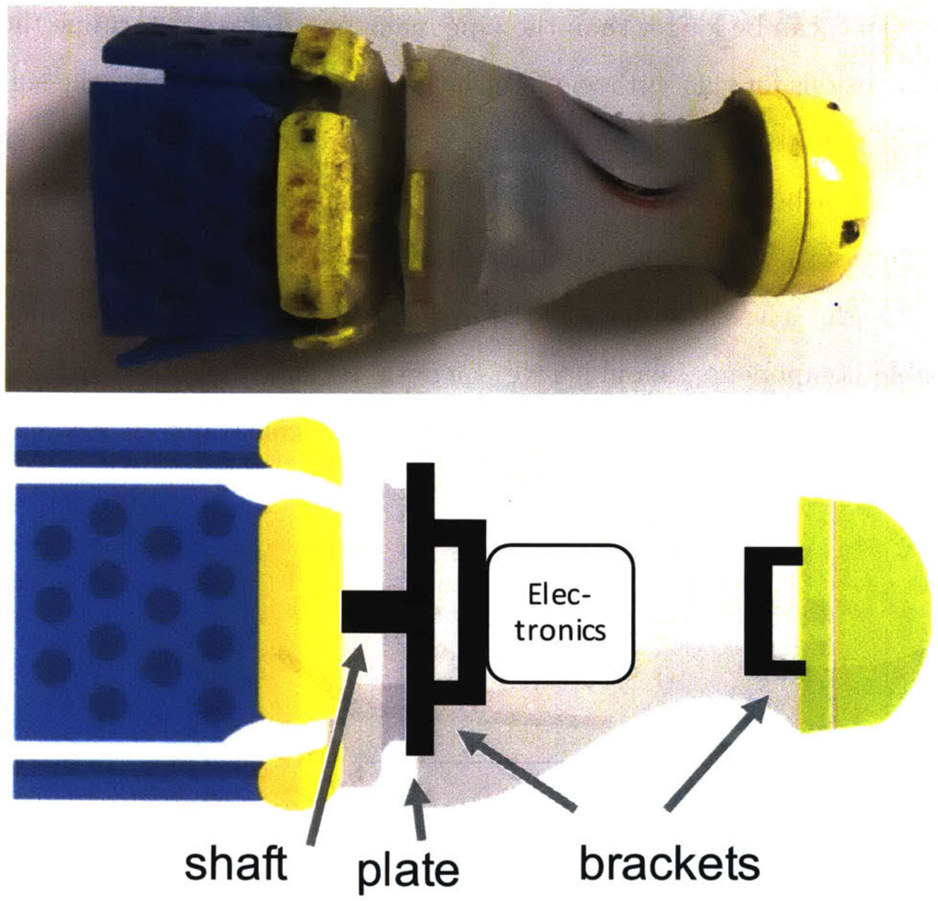


Figure 2-7: 2 inch robot prototype and Inside it.

From the above analysis result, a prototype for the 52mm diameter pipe system was built. It is shown in Figure 2-7-top. The outer diameter of the robot was 50 mm, in order to accommodate possible rust and dirt in the pipe. In the front of the robot is a rigid, smooth plastic head. It guides the robot around Tee junctions and pipe bents. In the back of the robot, there was a rigid plastic plate of 44 mm in diameter embedded in the silicone rubber. On the left side of this plate connects the shaft and the support for the leak sensor. The support of the leak sensor was 52 mm in diameter. This means in the 52mm diameter pipe, the support is in contact with the pipe surface all the time and maintain the robot's orientation. The detailed dimensions of the robot are listed in Table 2.1.

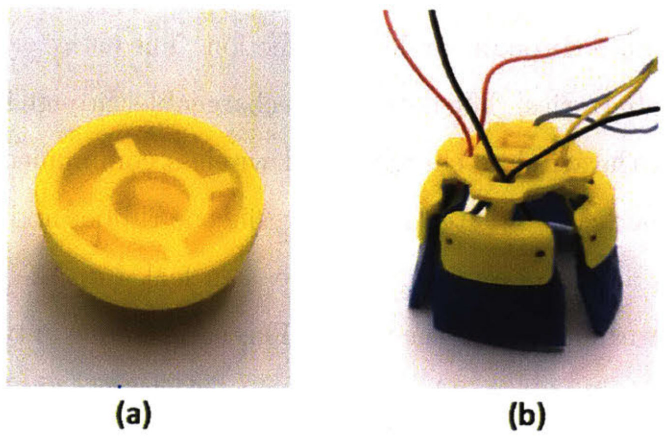


Figure 2-8: interface between plastic components and the rubber part in the robot prototype

The plastic components and the rubber robot body were joined together through brackets. Plastic parts cannot be bonded to the silicone rubber body via adhesives. Neither can they be bonded with screws and nuts. I propose a way to connect them without a third material. As shown in Figure 2-8, there were brackets similar to trusses in the back of the robot head and on the plate connecting to the support. When these brackets were embedded in the silicone rubber, the silicone rubber would fill in the empty spaces in the brackets and grab onto the brackets. Thus the plastic parts were connected to the rubber body.

The robot prototype can be described as a low-cost, wireless data logging device. It contains just enough electronics for it to record its motion and the leak sensor outputs. The motion sensors are accelerometers, gyroscopes and compass, all built

Table 2.1: Parameters of the new leak sensor.

Dimensions	value
L	50 mm
H_n	35 mm
W_n	25 mm
H_c	15 mm
L_d	117 mm

into one inertial measurement unit(IMU). The leak sensor, which will be described in more details in the next chapters, are essentially analog tactile sensors. A micro controller collects all the data and stores in a memory device. Upon request, it can transmit and receive data through wireless connection. The entire electronic system is powered by on-board batteries. The battery is rechargeable through inductive wireless charging coil. Given the wireless charging and wireless communication, the robot's electronic system is perfectly isolated inside the silicone rubber and thus waterproof. There is no physical access point, and no water can leak into the electronics. It is though necessary to turn on and off the robot. This can be done in two different ways. It can be a software power switch. The robot can be programmed to be waken up from a ultra-low power, sleep stage into fully functional stage. It can also be a hardware power switch. A button can be embedded inside the soft body. When pushed in a specific way, it turns on the electronics. In this robot prototype, I implemented the hardware power switch. All electronics components are off-shelf. The total cost is around 60 US dollars.

Table 2.2: Bill of Material in the prototype robot electronics system

Function	Item
IMU motion sensor	Pololu MinIMU-9 v5
Analog Digital Converter(ADC)	Knacro ADS1015
Micro controller	Wemos D1 mini with ESP8266 WIFI core
Data storage	MicroSD shield Wemos D1 mini + 16GB microSD card
Power control	Adafruit Micro Lipo jack+Adafruit push button
Wireless charging	Adafruit Inductive Charging Set
Power supply	Adafruit 3.7V 350mAh Lithium Ion Polymer battery
Voltage divider	50kOhm Resistors x4

The robot prototypes were fabricated in a simple molding process. All components of the robot, including the plastic parts and the electronics, were placed inside a mold. Then liquid form silicone rubber was poured into the mold. After 4 hours, the liquid rubber solidified and the robot was complete. The visual details of this molding process is illustrated in the Appendix.

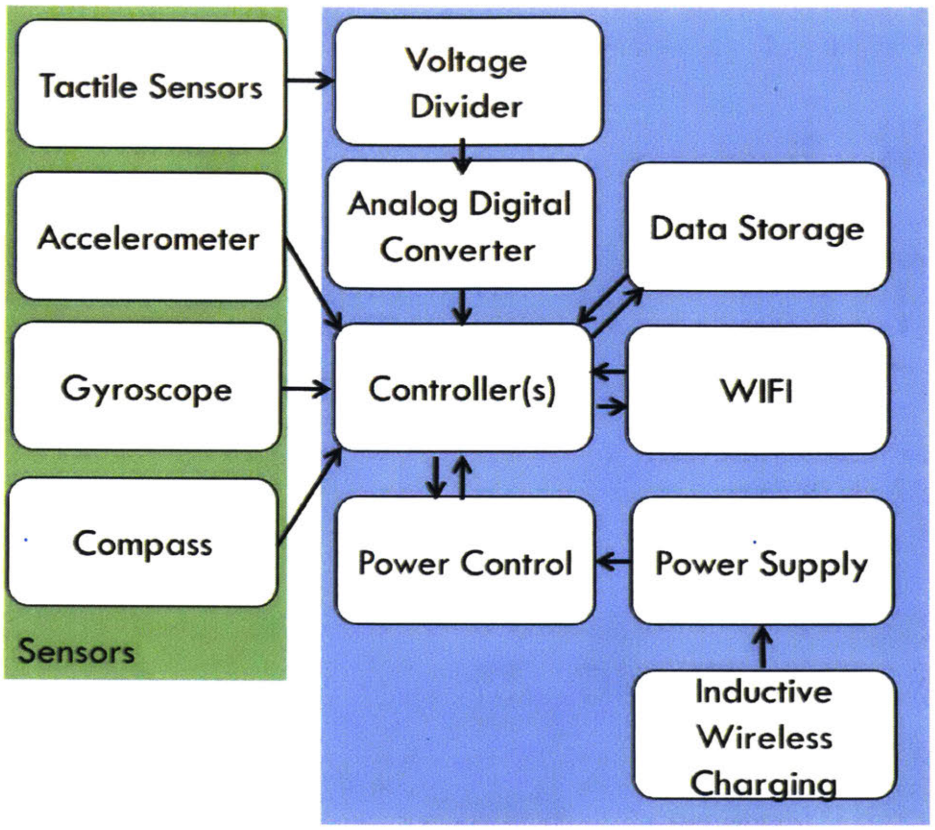


Figure 2-9: Electronics Diagram of the complete soft leak detection robot.

2.4.2 Field Test Setup, Insertion And Retrieval Methods

The 2" robot was tested in the industrial facility (Figure 2-1) provided by Pipetech LLC in Saudi Arabia. The goal of the tests was to verify that this robot was a good mobile platform for carrying sensors through small diameter water pipe systems with bends and Tee junctions. In the facility, a section of the 1.5km of pipe loop was isolated for the tests. This section covered 221 meters, and there are had four bends in it(Figure 2-11). The entire pipeline was in horizontal plane. The engineers at Pipetech LLC generously customized the facility for robot insertion and retrieval. The insertion tool (Figure 2-13) was connected to this segment at the entrance, and the retrieval tool (Figure 2-15) at the exit.

The robot was inserted into the pipes with a by-pass. Th by-pass is a parallel loop is an addition to the pipeline to give the water stream two route options to go from point A to point B. The loop is described in Figure 2-12. Before the robot insertion,

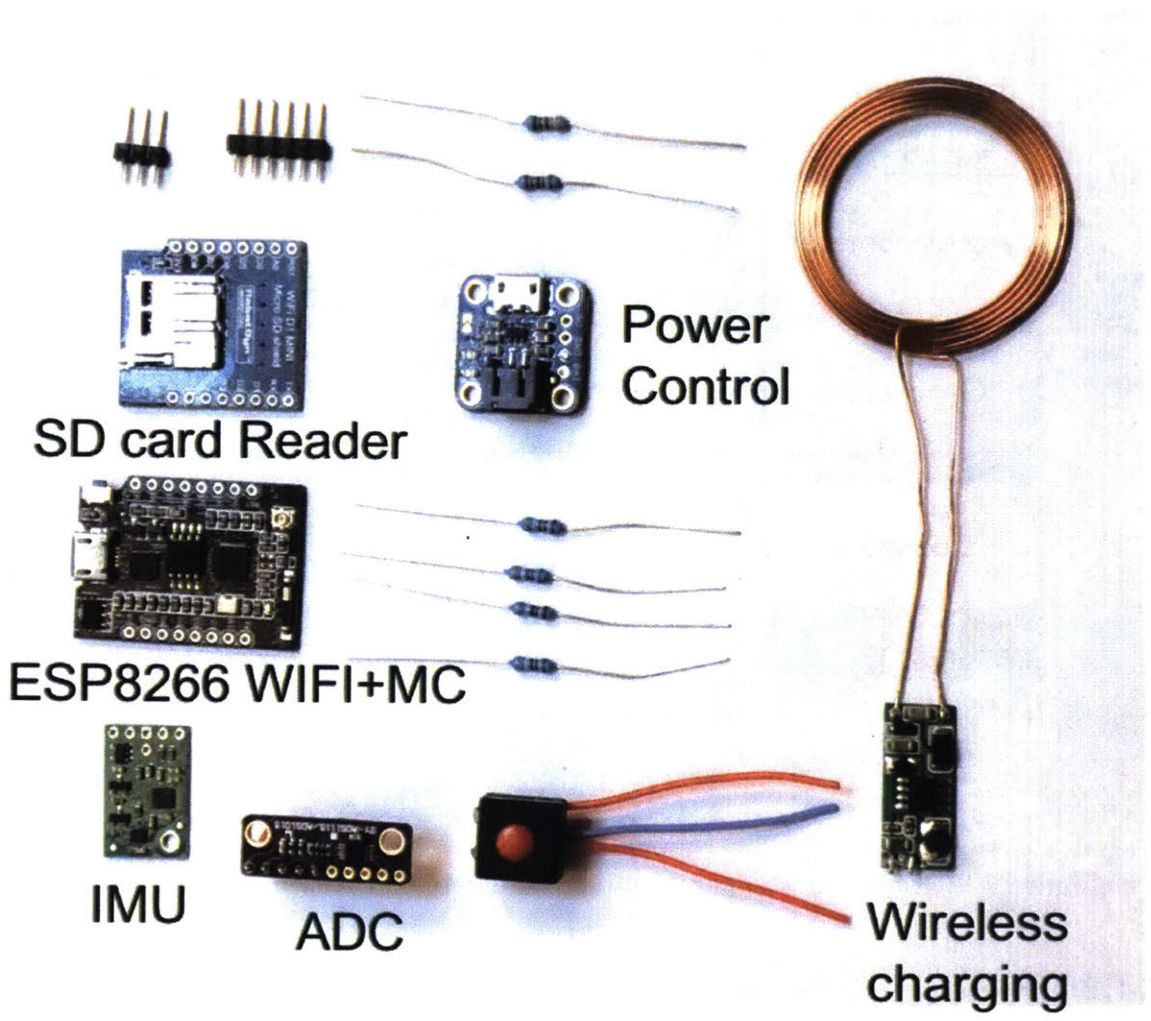


Figure 2-10: Electronic components inside the prototype robot

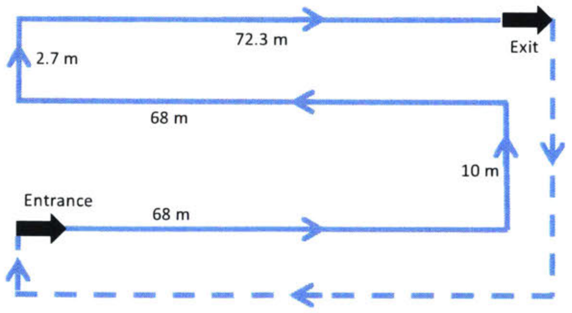


Figure 2-11: Sketch of the 221 m segment of the pipeline for testing the robot.

valve 1, 2 and 3 were closed and valve 4 was open. The water flow skipped the loop and went through valve 4 to the outlet. Then valve 1 was open, and the robot was inserted through valve 1 and passing the T junction to a point close to valve 2. Then valve 1 was closed and valve 3 was open. This action replenished the loop with water and pushed the robot against valve 2. Valve 2 was then open and valve 4 was closed at the same time. The water flow went through the loop and carried the robot toward outlet. This kind of parallel loop can be added easily to existing bends or U turns in the water pipe system. This method was implemented at the facility for field test shown in Figure 2-13 and used during the field test.

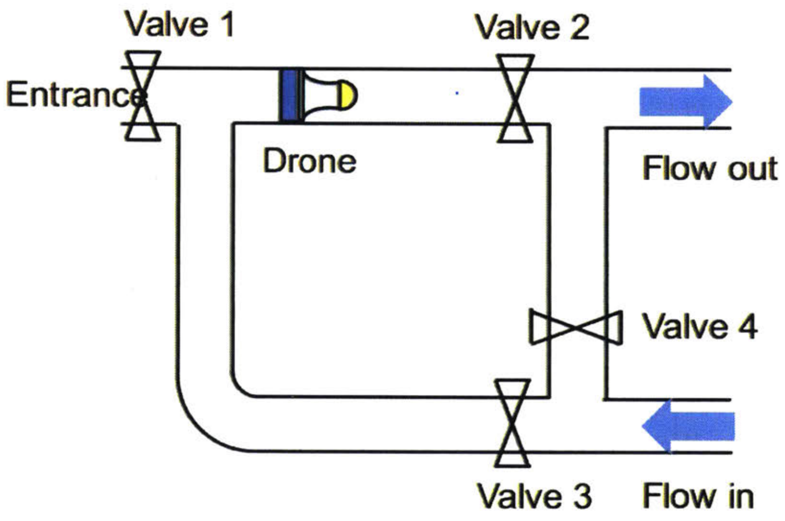


Figure 2-12: The concept of inserting the robot into the pipeline through a by pass.

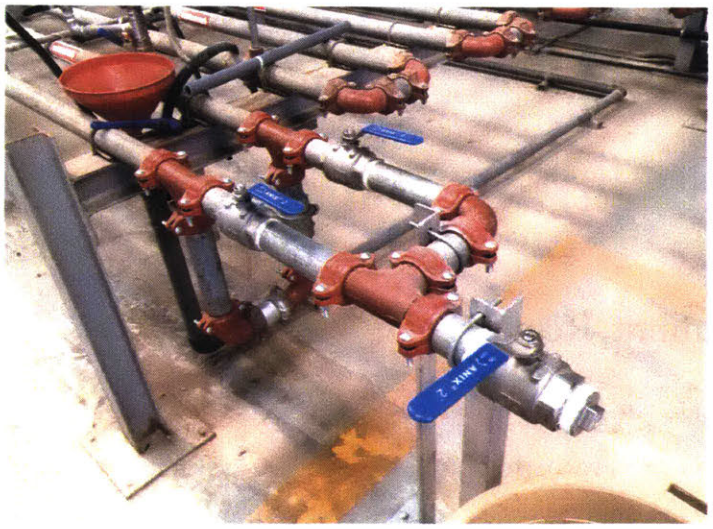


Figure 2-13: Example of the by-pass built for robot insertion.

On the other end of the pipe system, another by-pass with a Y junction was implemented for retrieving the robot. The retrieval tool setup was shown in Figure 2-14. Before the robot entered T junction on the left which is the entrance to the parallel loop, valve 1 and 3 were closed and valve 2 and 4 were open. This forced the flow and the robot to enter the parallel loop and move toward the Y junction. When the robot reached the metal mesh at the Y junction, its momentum and the fluid force behind it pushed it toward valve 3. Meanwhile, the flow went through the mesh and continued through the parallel loop. The robot then hit valve 3 and produced a clear "dong" sound. After detecting the sound or sensing the arrival of the robot with other methods, the operator opened valve 1 and then closed valve 2 and 4. The flow then skipped the parallel loop and moved through valve 1 toward outlet. Afterward, it was safe to retrieve the robot from valve 3 as shown in Figure 2-15.

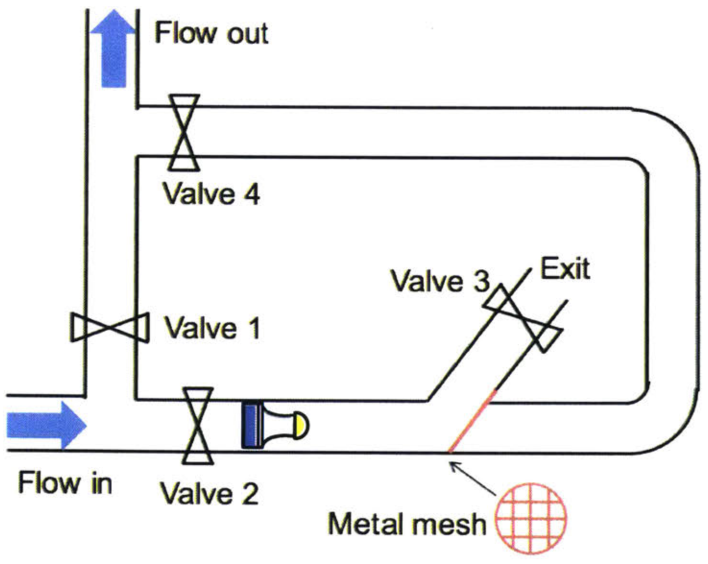


Figure 2-14: The concept of retrieving the robot through a Y junction.

2.4.3 Experiment Results

In a total of 20 tests, the robot transversed through the pipe system at two different speeds with 100% success rate. In the first set of experiments, the pipeline input pressure was 4 bar gauge. The test procedure was as follows: the operator deployed the robot with the insertion tool, waited for a few minutes, listened for the robot's arrival inside the retrieval tool, took out the robot and downloaded the motion information from the robot. This test was repeated for 13 times, and the robot was successfully launched and retrieved in all 13 tests. The average runtime was 345 seconds, and it put the average speed of the robot at about 0.64 m/s. The same tests were repeated 7 times for 2 bar gauge pressure at the pipeline inlet. The average runtime was 550 seconds, and the average speed of the robot was 0.40 m/s. In a total distance of 4,420 meters, the robot went through all 80 elbows and 40 T junctions (one T junction in each insertion and retrieval tools) at 100% success rate. It enabled the successful collection of leak measurement for validating the leak sensor's performance, which will be presented in the next Chapter. To the best of our knowledge, this was the first untethered robot that successfully ran through a long distance of small diameter water pipe under operating conditions.

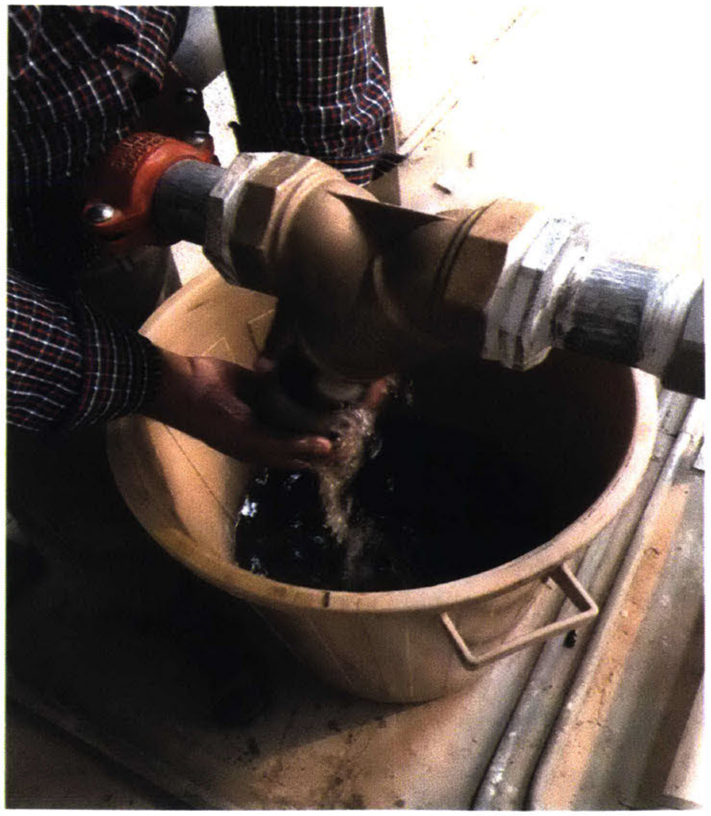


Figure 2-15: Example of the robot retrieval from a Y junction. The metallic mesh, in the shape of a tube, traps the robot inside. The mesh and the robot are being taken off the pipe system in this picture.

2.5 Adaptable 6" Robot and Its Field Test in Virginia, US

After the success in industrial facility tests, I moved on to the next milestone: field test in real, underground water mains. Our group was very fortunate to partner with a municipal water utility, the Wise County Public Service Authority(PSA) in Virginia, US, to conduct this field test. New robots were designed to fit in the 6-inch pipes in the field. To account for the pipe diameter inconsistency in real water mains, the new robots had more adaptability, so they were unlikely to get stuck in smaller-than-expected pipes. In this section, the design of this more adaptable robot is first presented. Then the details of the field test are also presented. Under the non-disclosure agreement with the partner utility, the leak detection result is withheld from this document.

2.5.1 More Adaptable Robot

The design of the 2" robot has a tight tolerance to pipe inner diameters, and that can be a problem. In the field test in the industrial facility, the exact inner diameter of the pipeline was known and accurate. The 2" robot was designed to fit in that pipe, and it succeeded. It may not always be the case that the exact inner pipe diameter is known, or the pipe inner diameter is consistent throughout the pipeline. It happens often that municipal water companies know the outer diameter of the pipeline but not the inner diameter. In another test, the current 2" robot was deployed into a smaller pipe, 1.9"(49mm) and it was stuck. Although the robot is compressible as it is made of soft rubber, it cannot transverse through the pipes smoothly when it is in a compressed state. Compressibility is different from adaptability for these soft rubber robots.

Adhesive friction on soft rubber

The robot was stuck in a smaller pipe because of friction force, and more specifically, adhesive friction. Elastomer has the advantage over rigid material in many applications because it is flexible, but in this case it also has the disadvantage of being too flexible. When a rigid material is sliding on another rigid surface under zero normal pressure, it has a friction coefficient of c_0. When a large normal pressure is imposed, the friction coefficient is still c_0. In comparison, a soft to rigid contact is completely different; the friction coefficient increases with the normal pressure as shown in Figure 2-16. When the silicone rubber is sliding on the rigid pipe surface under zero compression, it has a friction coefficient of c_0. When the normal pressure is increased, the silicone rubber will be compressed into any of the microscopic grooves on the rigid surface as if it forms a perfect seal, as illustrated in Figure 2-17. The distance between the rubber particle and the pipe material particle at the interface are significantly reduced. This leads to a large increase in the Van Der Vaal force between the two surfaces. As a result, the friction coefficient of a soft rubber on a rigid surface increases as the normal pressure increases. Even when the silicone rubber is compressed by a small percentage, the friction force can be very large. This is known as the adhesive friction[32, 1] effect on the soft rubber and other elastomer. It is the reason that when the 2" soft robot could not move while being compressed inside a 1.9" pipe.

It is necessary to design the robot to be adaptable so it can get through pipes smaller than what the robot is designed for. Even when the pipeline the robot is deployed into is smaller than expected, the robot should be able to transverse through it smoothly. This makes the robot more robust in the field. At the same time, the robot should still be soft and flexible, so it can maneuver through Tee junctions and pipe elbows. It seems to be a challenge to meet both the flexibility requirement and the adaptability requirement.

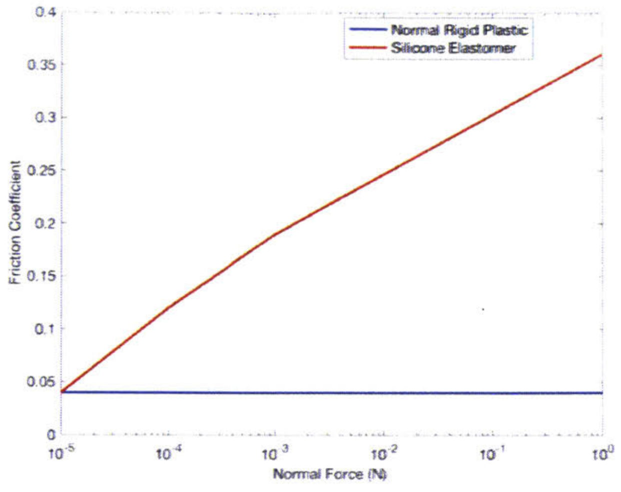


Figure 2-16: Example elastomer's adhesive friction curve [1]

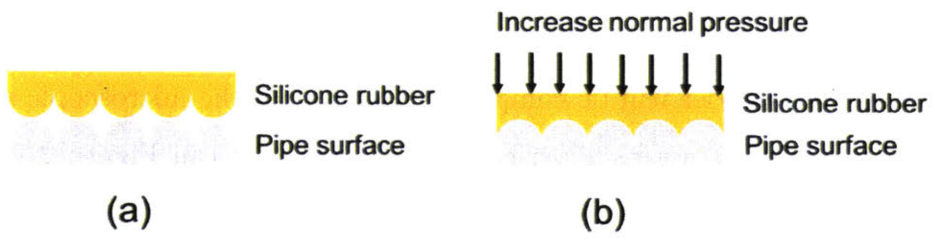


Figure 2-17: Concept of adhesive friction on elastomer

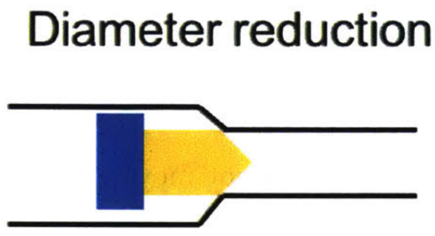


Figure 2-18: Additional design requirement on the robot for real water mains: adaptability

Reduce rubber surface area

I propose two solutions to add adaptability to the robot. The first one is to reduce the rubber surface area. When the 2" robot as shown in Figure 2-19 is compressed, more of the rubber body surface, in addition to the rigid support, comes into contact with the pipe surface. The rubber surface is where the adhesive friction occurs. To reduce and limit the adhesive friction, one can reduce all the rubber surface area that can possibly come to contact with the pipe surface. Therefore, when making a 6" robot for the field test in Virginia, I did not simply enlarge the 2" robot. They shared the same electronic system, but their geometries are different. Instead of a solid trunk of soft body, the 6" robot has a thin soft body with blue fins. The blue fins are made with knit fabric; they are more stiff than the silicone rubber 00-50 and they strengthen the thin rubber body. In addition, the leak detector is now in two layers rather than one layer. Each layer has four membrane sensors that covers 180 degrees of the 360 degree circumference. Given these two layer design and the thin soft body, this 6" robot can be compressed down to 4" in diameter without any rubber surface in direct contact with the pipe. The adhesive friction problem is thus avoided. This 6" robot was used in the field test in Virginia.

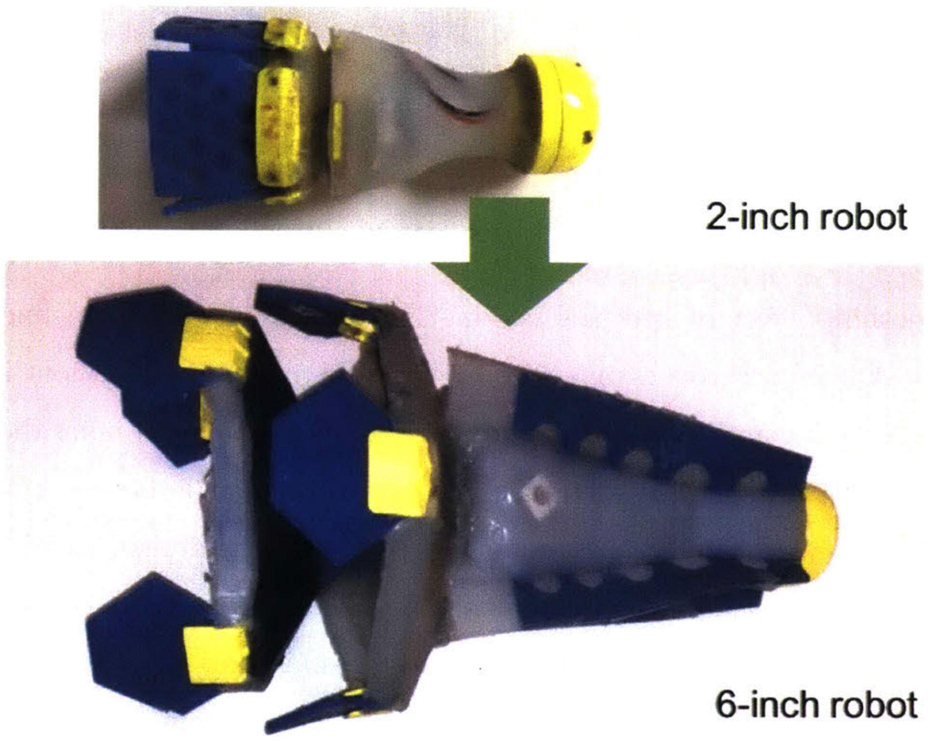


Figure 2-19: Comparison between the 2 inch robot and 6 inch robot

The second solution is to add a low friction coating to the robot. This solution suits robots of all sizes, in particularly the small size ones that is too difficult to apply the last method. If the soft body of the robot is coated with a low friction material that has relatively constant friction coefficient within a large range of normal pressure, then the robot could move within a smaller pipe smoothly. This coating material also has to be soft. One of such material is hydrogel[33]. Hydrogel is hydrophilic. When coating on the robot, it acts like a layer of water stick to the robot surface. This layer of water is incompressible and not dispersible. When the rubber body of the robot being compressed, this hydrogel layer will always stay between the rubber and the pipe surface, as illustrated in Figure2-20. There is no more contact between the rubber and the pipe surface and thus no more adhesive friction. Moreover, hydrogel has the friction property as low as that of water. With the hydrogel coating, the robot can slide along the pipe surface even at a large percentage of compression.

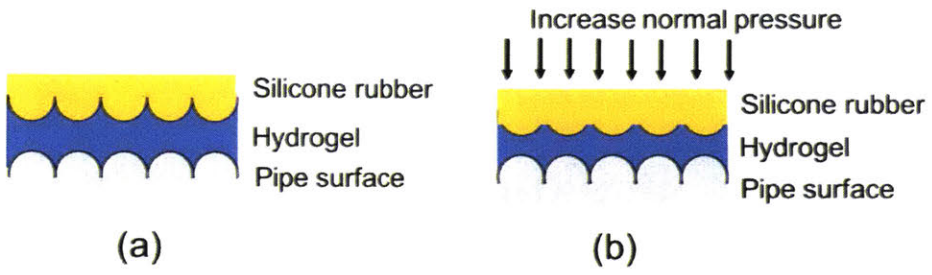


Figure 2-20: Concept of hydrogel coating on friction reduction

The smoothing effect of hydrogel coating is outstanding in the pipeline robot applications. A pair of 2" robots, one with hydrogel coating and one without it, were compared in experiments. When they were deployed in a 2"(51mm) inner diameter, schedule 40 clear PVC pipe, they all transversed smoothly through the pipe given very low pressure. This pipe had the inner diameter the robots were designed for. A video of this experiment is available at this link: http://mechatronics.mit.edu/hydrogel/. However, when the robots were deployed into a 1.94"(49.25mm) inner diameter clear PVC pipe, the results diverged significantly. The robot without hydrogel coating was constantly vibration in the direction of the pipe while moving through it. When studied in slow motion, it was observed that the robot was stuck momentarily, then the water pressure in its back built up and pushed it forward a little, and then it was stuck again. This process repeated till the end of the pipeline. In comparison, the robot with hydrogel transversed through the pipeline smoothly, as if the pipe was not smaller than the robot's size at all. The video of this experiment is also available at this link: http://mechatronics.mit.edu/hydrogel/. Both robots were compressed in this smaller pipe. While the regular soft robot's motion became unstable, the hydrogel coated robot's motion was unaffected by the reduction in pipe diameter.

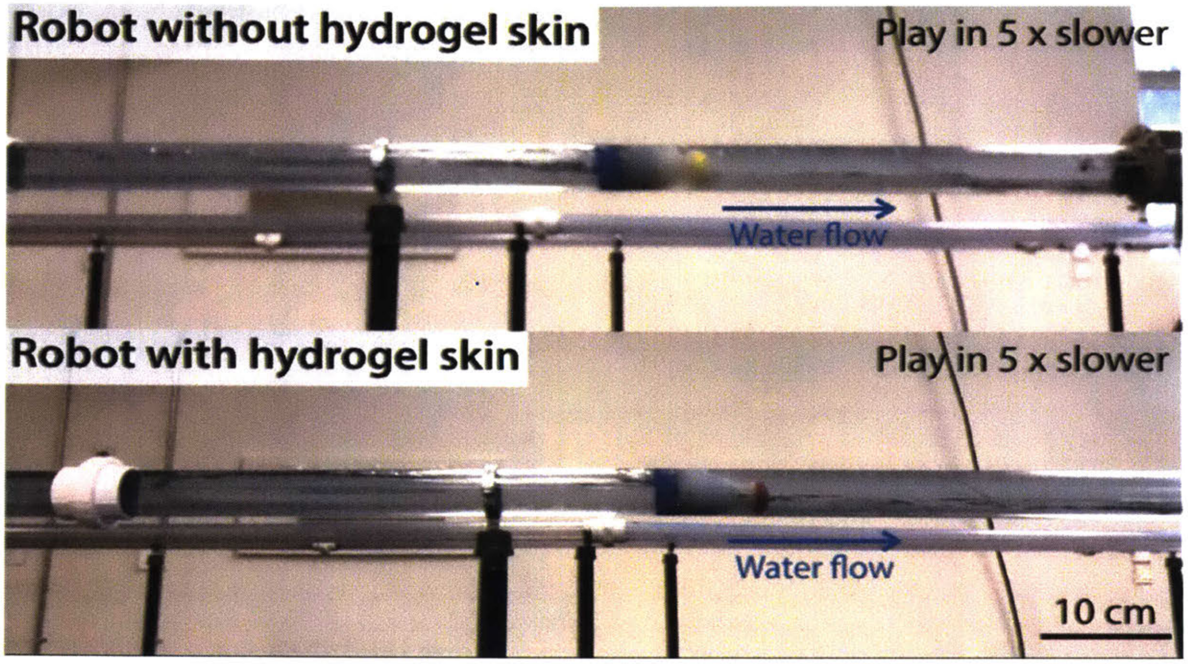


Figure 2-21: Comparison between the regular robot and hydrogel coated robot, Video link: http://mechatronics.mit.edu/hydrogel/

2.5.2 Field Test Site Preparation

The first field test in the US was conducted in a 6-inch water main in Wise County, VA, on Wednesday January 10, 2018. This pilot project was supported by Wise County Public Service Authority in Wise County, VA. The inspected water main is about 1.2 miles in length and built with PVC pipes in the early 1970s. The inner diameter of this water main is 6 inches, and the pipe is of SDR21 specification. It is completely buried underground. It was known to be leaking but none of the exact locations of any leaks were known. The water pressure inside was at least 20psi gauge pressure. The water flow rate was estimated to be around 200 gallons per minute. The 6" robot as shown in Fig. 2-19 was used in this field test. During the test, the robot and the water flow entered from the Nash Chapel end of the pipeline and exit from 5599 Pole Bridge Rd end, as indicated in Figure 2-22.

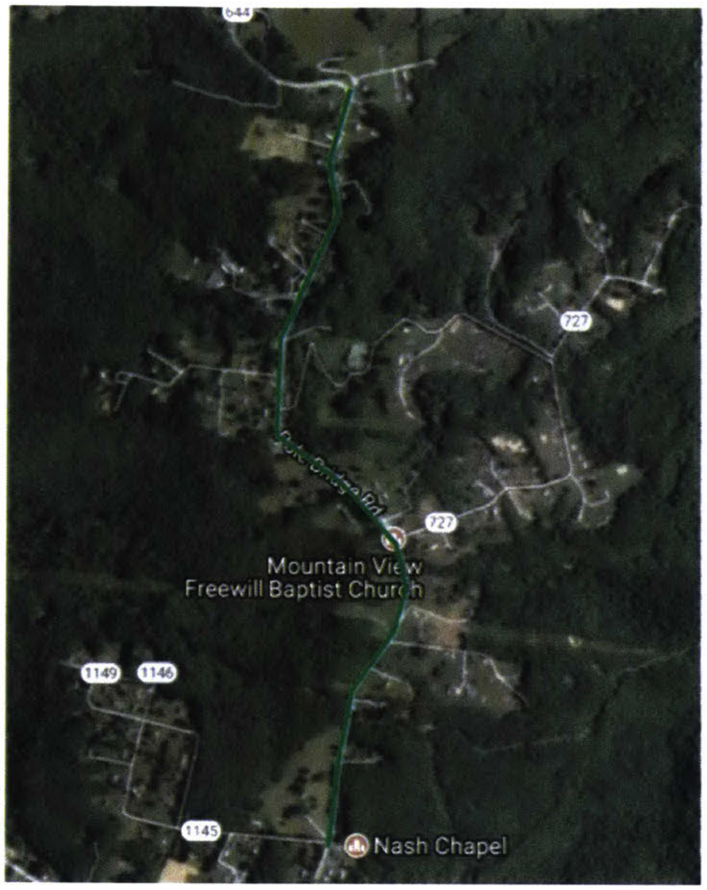


Figure 2-22: Satellite map of the 6 inch pipe experiment site

Wise County PSA made two reconfigurations to the existing water main for testing the robot, and those modifications are illustrated in Figure 2-23:
1) Access points were installed on the buried water main to allow the robot to enter and exit. Two Tee junctions were installed at each end of the water main and made accessible from above ground. The first Tee junction was installed after a valve near Nash Chapel as shown in Figure 2-22. The robot would be inserted into the water main through this Tee junction. A second Tee junction was installed near 5599 Pole Bridge Rd, right before a pipe dead end. The robot would be retrieved from this Tee junction. Both Tee junctions were of 6 inches in inner diameter in all three ways.

2) Valves on any Tee junctions along the water main that may draw the robot in

were turned off. The leak detection robot flows with water, and it has no active control of which way to go at a Tee junction. It simply flows with the water. The robot is designed for pipes of an inner diameter of 4 - 6 inches. To prevent the robot entering any pipe branches and guide the robot toward the designated exit point, Wise County PSA was asked to shut off the water flow to any 4 inch or larger branches along the 6-inch water main. There are a few Tee junctions and many service connections, but among them there is only one 4-inch connection. It is the Road 727 connection in Figure 2-22. The valve on this 4-inch connection was turned off and all the other smaller connections were left on.

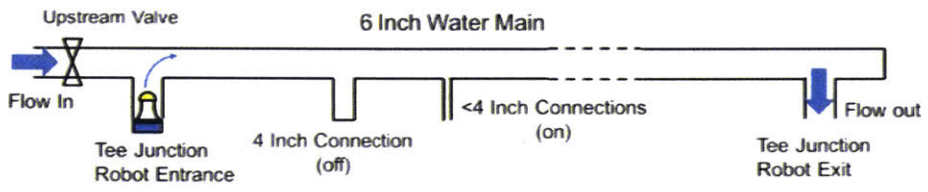


Figure 2-23: The concept of inserting the robot into the pipeline through a by pass.

2.5.3 Field Test Procedure

The field test was performed in the following procedure, and this procedure was also documented in Figure 2-24 and Figure 2-25:

1) The upstream valve was shut and the 6-inch water main was drained until the robot entrance Tee could be opened safely.

2) The robot was disinfected with bleach spray.

3) PSA technician Steven Jenkins step into the trench, put the robot into the entrance Tee junction after this Tee was opened fully. He put his hand and the robot into the Tee, and manually aligned the robot so its head pointed downstream.

4) The entrance Tee junction was closed, and the upstream valve was gradually turned on to increase the water flow in the pipe.

5) The exit Tee junction was 25% open in order to ensure a strong water flow in the pipe.

6) After 42 minutes of waiting, the robot arrived inside the exit Tee junction. Its yellow head appeared behind the gate valve at the Tee. The Tee junction was fully open to let the robot out.

7) The exit Tee junction was gradually turned off to complete the experiment.

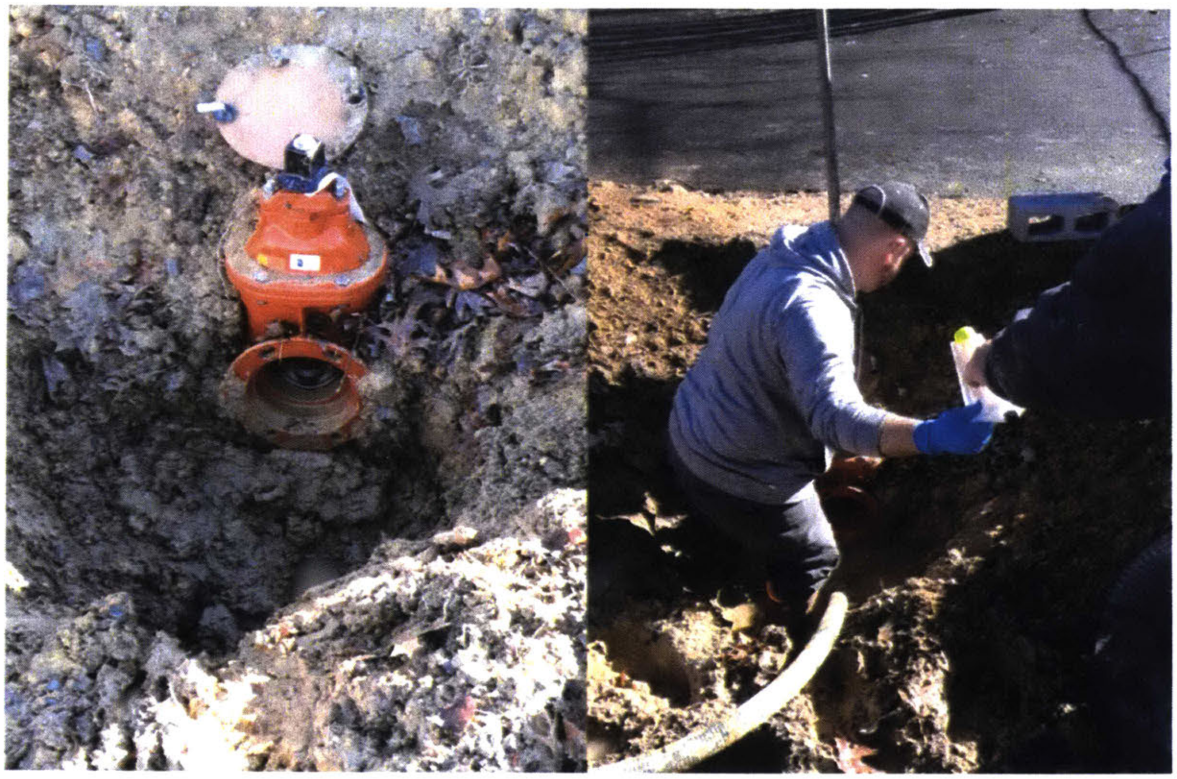


Figure 2-24: Picture of the Tee junction for robot entrance (left) and PSA Technician Steven Jenkins manually placed the robot in this Tee

2.5.4 Field Test Result and Discussion

The 6-inch robot was successfully deployed into this 6-inch water main and retrieved at its end. The robot was successfully inserted into the water main manually through a Tee junction, and successfully retrieved from the same water main from another Tee junction. It took the robot 42 minutes to transit through the 1.2 miles of pipes without human intervention. The average speed of the robot was 2.5 feet per second, slightly slower than human walking speed.

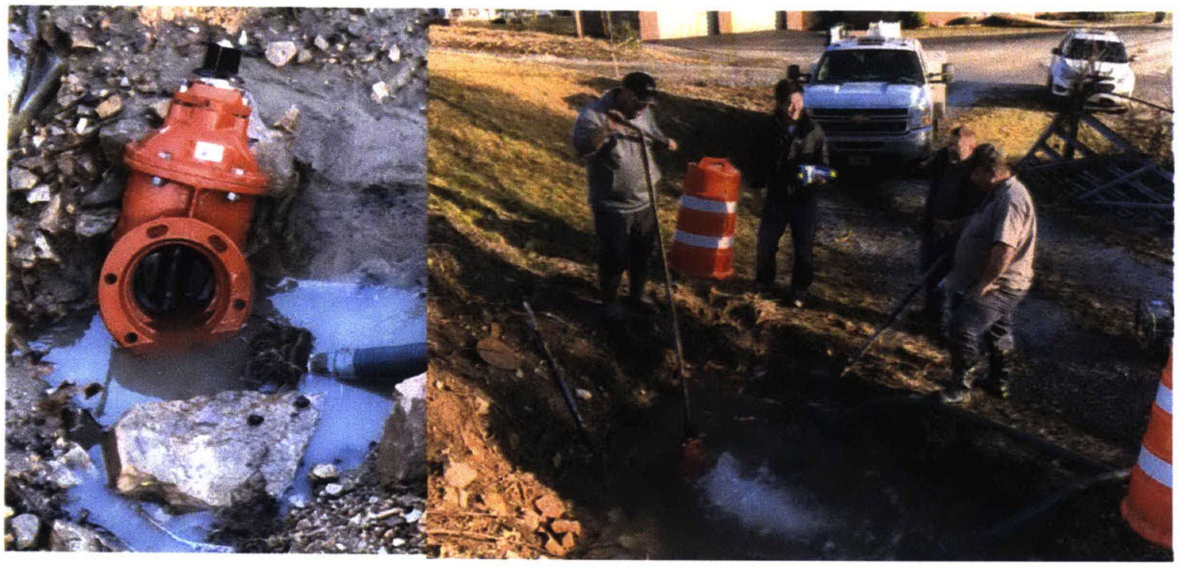


Figure 2-25: Left: Picture of the Tee junction for robot exit. Right: Picture of shutting the Tee junction off after the robot was retrieved. The persons in the picture from left to right are Steven Jenkins(PSA), You Wu(MIT, holding the retrieved robot), Alan Harrison(PSA) and Shane Clark(PSA).

The most significant finding in this field test is that the robot can be retrieved from the pipelines without a capturing tool. In the Saudi Arabia experiment, a net was placed inside the pipeline to intercept the robot, as shown in Fig. 2-14. In this Virginia field test, the robot simply followed the out flux of water flow and exited from the opening Tee junction. This was possible because the robot is soft. While traveling inside the water pipe, the robot is as if part of the water flow. It goes where the majority of water goes.

However, this simple retrieval comes at a cost. While retrieving the robot without a capturing tool, the technicians had to leave the exit Tee junction partially open for a while. A significant amount of water was coming out from the open Tee junction and wasted. The technicians have to setup drainage pumps to remove the wasted water and preventing the site from being flooded. In comparison, with the capturing tool in the Saudi Arabia experiment, the retrieval process was clean and little water was wasted. This clean retrieval process is better suited for robotic inspections in busy urban areas. Although more complex, it prevents flooding of the streets.

2.6 Conclusion

Passive soft robots are developed to solve the in-pipe mobility challenge. Those robots are flow driven, so they can inspect long distance pipelines with minimum power requirement. They are soft and flexible, so they can maneuver around pipe elbows and Tee junctions. Those robots can also be adaptable, so they can be used in pipes they are sized for and also pipes that are slightly smaller. The robots demonstrated their mobility and reliability in field tests. The first robot prototype successfully ran through a 52-mm-inner-diameter, cast-iron industrial pipe system 20 times. Another prototype successfully ran through an underground 1.9km long, 150-mm-diameter, PVC water main. With this reliable robotic platform, we can now design, field-test and optimize the leak detection sensor.

Chapter 3

Leak Detection In Operating Water Pipes

3.1 Overview

In recent years, an increasing amount of effort has been put into developing effective leak detection solutions for water pipes. Among them, the pressure gradient based method developed at the Mechatronics Research Lab at Massachusetts Institute of Technology excels for its sensitivity in low pressure, small diameter pipes[34, 35, 31, 36, 30]. It can also work in both plastic and metallic pipes carrying gas or water. However, the method was only verified in static fluid pipes, and the previous sensor designs were unable to detect leaks when there is a significant water flow in the pipe. This is undesired as the inspection can only be performed when water service is shut down. A modeling analysis shows that fluid dynamic effects in the water pipe make the original sensor's dynamics too slow to react to leaks[37]. Moreover, this leak detection method is prone to false alarms such as obstacles in the pipes, but there is a lack of studies on this topic. In this chapter, I present three things: the design of a new leak sensor that is fast enough to detect leaks in dynamic fluid environments, a prototype for 52mm-inner-diameter pipeline tested in an industrial facility, and a method to differentiate leaks from false alarms supported by the test results.

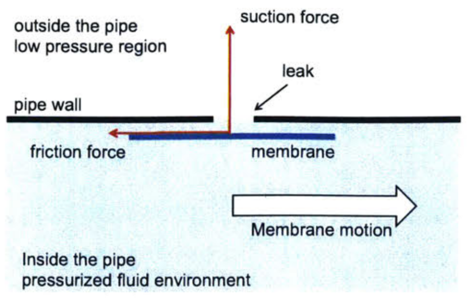


Figure 3-1: Illustration of the leak detection method.

3.2 Background

In recent years, there has been a surge of new leak detection technologies from both academia and industry. Among them is a predominance of acoustic/wave based leak detection technologies. For example, a network of hydraulic and acoustic sensors can be instrumented on the pipe to look for the occurrence of new leaks [12]. A free floating acoustic sensor can travel with the flow and listen for leaks from inside the pipe [20]. Those acoustics and pressure based methods suffer from loss of signal-to-noise ratio or accuracy in low pressure water pipes and pipes made of vibration damping material such as plastics. This is where pressure gradient [30, 31] based in-pipe leak detection method developed at Massachusetts Institute of Technology fills in. Its sensitivity in low pressure, small diameter pipes is outstanding, and it can be used in pipes of any material and any fluid. This method uses a membrane to detect the sharp pressure drop in a small region around the leak [30]. When the membrane moves in the pipe and arrives at the leak (Figure 3-1), the pressure drop on the leak side of the membrane results in a suction force on it. The suction force will press the membrane against the pipe wall, and this results in an increase of friction. By detecting the effects of the suction force, the increase in friction force, the change of motion in the device the membrane is attached to, or any combination of the three, one can infer the presence of a leak.

There are two main challenges in applying pressure gradient based leak detection method to real water pipe systems. The first challenge comes from the water flow. Previously, prototypes of this method was only successfully demonstrated in static air or water pipes [30, 31, 38]. Further experiments showed that none of them could consistently register the leak when water is flowing through the pipe. This is a significant limiting factor for the commercialization, as it requires the water service being shut down for the entire duration of the inspection job. Moreover, the leak sensor needs to be transported through the pipeline, and fluid-driven robots such as the Pipeline Inspection Gauges (PIGs) [22] are the most energy efficient platforms to do so. On a fluid driven robot, the leak sensor will be moving at the speed of the flow and it is required to sense leaks in such dynamic fluid environment. The theory behind the pressure gradient based method is sound and it should work similarly well with or without the flow. A new leak sensor design for pipes under operating condition is worth exploring.

The second challenge is about false alarms. This method requires a membrane moving parallel to the pipeline wall at a close distance. The membrane will mechanically interact with obstacles in the pipeline such as dirt, scales, misaligned pipe connections and other irregularities. These obstacles can trigger false alarms. This issue is unprecedented in acoustics or pressure based non-mechanical methods. How this sensing technique treats obstacles needs to be studied.

A new ingredient to the leak sensor design comes from recent advancement in soft materials in robotics applications. In particular, soft sensors for haptic application and assistive technologies demonstrated transferable features that is desired for this leak sensing technology. Those soft sensors are elastomers with sensing elements embedded in. The sensing elements can be micro fluid channels [39], or conductive particles [27]. They can be made with different sensitivity in different directions. They are compact, flexible, single piece sensors. In contrast, implementations of the pressure gradient based method have been rigid and mechanical. Those ideas of soft sensors can be applied to the new leak sensor design.

In this chapter, I present the design of a new leak sensor that can detect leaks in dynamic fluid environments and differentiate leaks from false alarms. System dynamic analysis was first performed on the original leak sensor [30] to study why it did not work in this environment. From this analysis, a new leak sensor design was proposed, and a prototype for 52mm inner diameter pipes was successfully tested in the lab and in an industrial facility. With the data of the sensor's response to obstacles from the test results, a method to differentiate leaks from obstacles was developed.

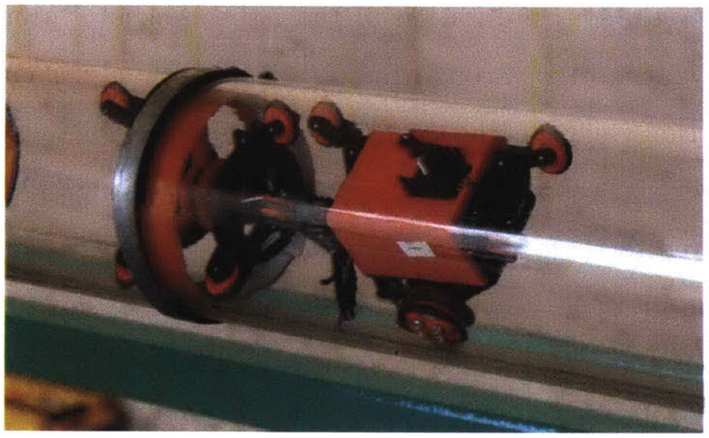


Figure 3-2: Picture of the original leak detection system. It is a locomotion module pulling the leak detector.

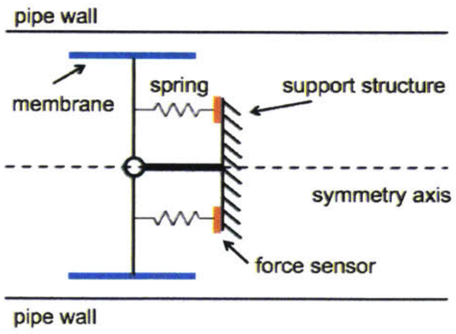


Figure 3-3: Schematics of the cross-section view of the original leak detector

3.3 Leak Sensor Design

3.3.1 Design Requirements

The goal is to design a sensor that utilizes the pressure gradient based method described in Figure 3-1 to detect leaks in operating water pipes. The researchers was honored to partner with PipeTech LLC in Saudi Arabia in this project. PipeTech LLC generously provided their industrial facility, a 1.5km long, 52mm inner diameter

metallic pipeline for the tests. Three design requirements are derived with specifications for this pipeline. The leak sensor must be able to
(1) generate an indicative leak signal while moving through a pipe with a significant water flow. The minimum flow speed and sensor speed is 0.3 m/s.
(2) detect small leaks under low pressure. The target leak is a hole of 4mm in diameter and the line gauge pressure below 2 Bar.
(3) go over small obstacles in pipes without getting stuck, and differentiate them from leaks.

3.3.2 System Modeling and Analysis

In the following parts of the chapter, I want to first make the following distinctions in names to avoid confusions. The leak sensor is referred to as the system, and it is the complete module that includes all the components. One of these components is the sensing element. It refers to the component that converts mechanical effects to electrical effects, and it can be force sensitive resistors, encoders or other sensors.

Before generating new sensor designs, it is worthwhile to understand why the previous leak sensor design failed to meet the new design requirements. Why can it not detect leaks in a pipe with a significant water flow? The original design[30] is illustrated in Figure 3-2 and 3-3. The design is axial symmetric around the pipe's centerline. The membrane is attached to a gimbal mechanism; the gimbal is maintained in a normal configuration by a set of springs. The sensing elements are force sensitive resistors at one end of the springs. When the system passes a leak, the increase of friction force on the membrane is transferred through the gimbal to the sensing elements. The support structure which maintain the radial position of the system is simplified in Figure 3-3. This design and its variations have been demonstrated to work in pressurized stagnant air pipes, even at line pressure as low as around 1 Bar[30] . However, when the leak sensor is attached to a flow driven robot and tested in a pipe carrying a water flow of similar line pressure, it was unable to detect leaks.

This design seems to work from a static point of view, but a dynamic analysis gives insights of why it underperforms in water flow. The system can be approximated

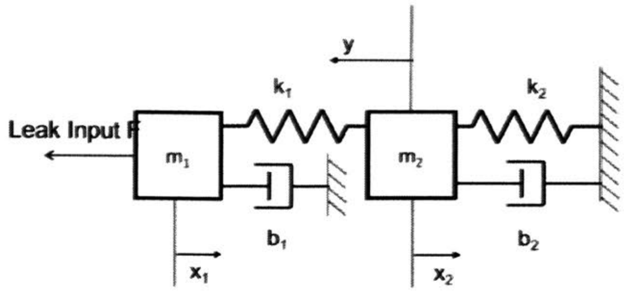


Figure 3-4: The original and the new leak sensor can both be modeled as a two mass system.

by a two-mass-spring-damper lumped parameter model, as shown in Figure 3-4. The membrane has a mass of m_1, a spring constant of k_1 and a damping coefficient b_1, due to its elastic properties and the fluid drag force. The mass of the gimbal is m_2, and it has a damping coefficient b_2, due to fluid drag force and mechanical friction. The spring constant of the springs is k_2. The support is treated as a the ground. The entire system is assumed to move only in the axial direction of the pipe, and the angular displacement of the gimbal is very small and the system can be linearized. The input is an increase in friction force, F, on the membrane m1. The reference frame is fixed to the support in Figure 3-2. The output y can be interpreted as the displacement of the gimbal m_2. It is proportional to the spring force on the sensing element. The values of these parameters, either measured or estimated based on fluid dynamics, are summarized in Table. 3.1.

$$\begin{bmatrix} \dot{x}_1 \\ \ddot{x}_1 \\ \dot{x}_2 \\ \ddot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-k_1}{m_1} & \frac{-b_1}{m_1} & \frac{k_1}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_1}{m_2} & 0 & \frac{-(k_1+k_2)}{m_2} & \frac{-b_2}{m_2} \end{bmatrix} \begin{bmatrix} x_1 \\ \dot{x}_1 \\ x_2 \\ \dot{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{m_1} \\ 0 \\ 0 \end{bmatrix} F \tag{3.1}$$

$$y = -x_2 \tag{3.2}$$

In a dynamic flow environment, the system does not have a high bandwidth and that is the main problem. When the leak sensor is attached to a flow driven robot, they will move at approximately the speed of the flow. When moving at a high speed,

Table 3.1: Parameters of the original and the new leak sensor.

System	Original leak sensor	New leak sensor
k_1 (N/m)	8600	8600
m_1 (g)	2.0	2.0
b_1 (kg/s)	0.6	3.0
k_2 (N/m)	2208	2150
m_2 (g)	5.6	0.5
b_2 (kg/s)	3.0	3.0
m_a (g)	14.5	0

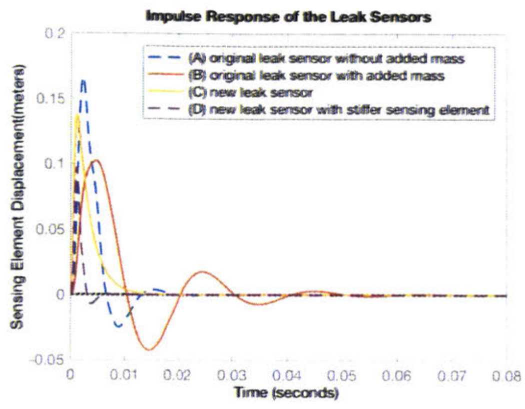
the occurrence of a small leak is similar to a high frequency input or an impulse input on the system. The leak sensor must have a high bandwidth in order to react to the leaks, or it will pass the leak without a change. If only the mass of the system's components is considered, the original sensor has an impulse response as curve (A) in Figure 3-5, and its dominant damped natural frequency at 88 Hz. This is true for the system in air. However, in a water environment, it is more difficult for the system to change motion since it has to move the fluid in its way. This effect is known as added mass in fluids[40]. In the original leak sensor, the gimbal is approximately a thin plate moving perpendicular to the direction of water flow. Its large frontal area displaces a blob of water as it moves. Thus it has a large added mass m_a proportional to its frontal area.

$$m_a = \rho \pi r^2 \tag{3.3}$$

The estimated m_a is almost three times of m_2 in this case. Adding m_a to m_2 in the system model, the original sensor's impulse response slows down and becomes curve (B) in Figure 3-5. Its dominant damped natural frequency reduces from 88 Hz to 28 Hz. The system has a much smaller bandwidth in water, and thus it cannot detect leaks when moving with a fast water flow. On the other hand, the membrane is like a thin plate moving parallel to the flow, displacing little water as it moves. Thus its added mass is approximately 0.

The second problem is that the original system cannot differentiate leaks from obtrusions. In pipes, obtrusions are verycommon. These includes the o-ring at every pipe joint, reduction in diameter at every valve, and occasional tuberculation and rust

Figure 3-5: Simulated impulse response of the leak sensors.



build-ups. These obtrusions interact with the system in a similar way as leaks. Leaks pull on the membrane m_1 and indirectly displace the gimbal m_2. Obtrusions in pipes push on the gimbal m_2 directly. Both of external forces are in the same direction, and their effect cannot be differentiated from the displacement of m_2 or the output y. Moreover, obtrusions push on the gimbal m_2 directly, unlike the leak input F that needs to go through the intermediate mass-spring-damper before reaching the gimbal m_2. This means the system has more gains for the disturbance from obtrusions than the desired input from leaks. Not only is the signal from this system indistinguishable from the noise from disturbances, the signal is also weaker than the noise.

3.3.3 Proposed New Leak Sensor Design

The previous dynamic analysis provides insights into the design of a new leak sensor. To detect leaks in a pipe with a significant water flow (design requirement 1), the leak sensor should have a high bandwidth. To do so, m_2 of the system must be small, and this m_2 includes added mass. One way to minimize the added mass on m_2 is to eliminate any components between the membrane and the sensing element that may have a large frontal area moving perpendicular to the flow. One particular design to realize that is shown in Figure 3-6. In this design, one side of the sensing element is connected directly to the membrane, and the other side of the sensing element is fixed on a umbrella-like support structure. When there is a flow moving from left to

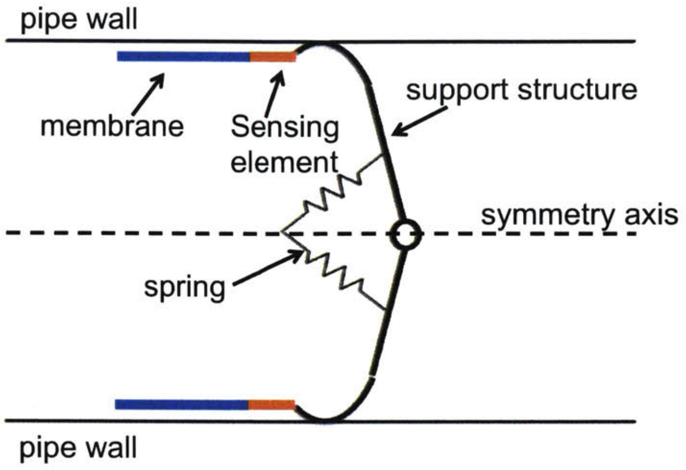


Figure 3-6: Cross section view of the new leak sensor design. The design is axisymmetric around the pipe's centerline. It has two features: 1) both membrane and the sensing element are parallel to the flow for minimal added mass; 2) an umbrella-shaped support structure that keep membrane close to the wall while adapt to pipe irregularities.

right in the pipe in Figure 3-6, the flow will push the umbrella-like support to expand and maintain its contact with the wall at all time. Given a strong fluid drag force, it is reasonable to assume that the support does not move due to leak induced forces, and can be treated as the ground in the system model (Figure 3-4). Then m_2, b_2 and k_2 becomes the mass, damping and spring constant of the sensing element. Since the sensing element now moves parallel to the water flow in the pipe, and like the membrane, the added mass on the sensing element is close to zero. With less mass, the new leak sensor will have a higher bandwidth and thus it can detect leaks in a higher speed water flow.

The support structure helps the system to detect small leaks (design requirement 2). The membrane needs to be as close to the pipe wall as possible to effectively detect small leaks[30]. The support structure is like an umbrella; it is spring loaded, and at its far end is a 180 degree rigid bend as shown in Figure 3-6. When the system is placed in the pipe, the spring loaded support will keep the bend in contact with the pipe wall. The bend always keep the attached membrane within a fixed small distance from the wall, and this distance is equal to the height of the bend. When

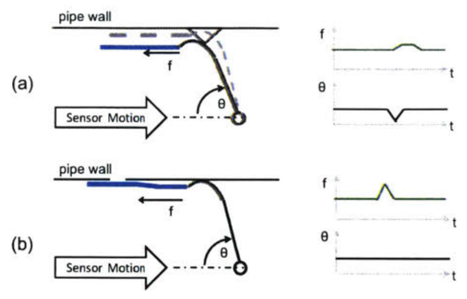


Figure 3-7: New leak sensor in two scenarios: (a) obstacle and (b) leak.

there is a flow moving toward the right in the pipe in Figure 3-6, the flow will push the umbrella-like support to maintain expanded. Given the strong fluid drag force, it is safe to assume that the support does not deform due to leak induced forces. Thus, this umbrella-like support not only keep the membrane close to the wall but also acts as a stable base for the membrane and sensing element. It can then be treated as fixed in the system model (Figure 3-4.

This support design allows the system to adapt to and differentiate obstacles from leaks. This spring loaded support can adapt to pipe diameter changes with a push from the water flow on its back. With a position encoder, it can measure these changes. Figure 3-7 shows two scenarios where the leak sensor encounters an obstacle and a leak. An obstacle is equivalent to a pipe diameter reduction, it will compress the support and leave the membrane little perturbed. When the obtrusion is significant, it will bend the membrane radially inward. In comparison, a leak will not compress the support but pull the membrane. obtrusions bend and leaks pull. These two inputs cause distinctively different interaction with the new system, and can be measured independently. With this design, the motions and forces on the membrane and the support can be measured separately to indicate leaks and obtrusions. For example, strain sensors can be used to monitor the membrane, while encoders or force sensors can monitor the support. By searching the different signal patterns in both the force (or motion) on the membrane and the configuration (or force) on the support, one can tell leaks apart from obstacles and other false alarms.

Two implementations of the membrane and sensing element are shown in Figure 3-8. The membrane is made of silicone rubber (blue), MoldStar 30 from Smooth-On

Inc, in a mold. The sensing element can be any force or strain sensor; in this case, it is a conductive silicone rubber cord (black) from Adafruit. It is a strain sensor with increasing electric resistance when being stretched. Both are silicone based material and they bond together chemically to form a single piece of membrane sensor. The MoldStar 30 rubber is also waterproof, and it wraps the non-waterproof conductive rubber cord inside to protect it from short circuiting. The fabrication of this membrane sensor can be seen in Appendix A. The membrane and the sensing element are loosely fit inside a rigid slot on the support structure. The two points on membrane where the wires connect to the ends of the sensing element is bonded to the bottom of the slot with superglue. The slot fully covers the sensing element, so the sensing element can only be stretch in y direction(Figure 3-8 but not bend. Each membrane is 37mm wide, so 4 piece can cover about 90% of the circumference of a 52mm diameter pipe. The membrane is 2mm thick so it can wrap the conductive rubber cord (1mm thick, 2mm wide) inside. There are dimples on the side of the membrane facing the pipe wall, to increase the contact friction coefficient. A preliminary test in air shows a 30% increase in friction coefficient when the membrane is pressed against smooth PVC surfaces.

The length L of the membrane is important to the leak sensor's performance, and it is determined the flow speed and system's sampling rate. For a small leak, The leak input to the system is an impulse of duration $t \leq L/V$ where V is the speed the membrane is moving at. When the membrane is carried by a flow driven drone [22], V is the same as the flow speed. Assume the system's physical bandwidth is very high, its output is then determined by its sampling rate f_s. It would help study the characteristics of different type of inputs by capturing $n > 1$ frame of the input. Thus the length of the membrane can be determined by

$$L = \frac{nV}{f_s} \tag{3.4}$$

In the prototype, the lower end of the sampling rate is $f_s = 20Hz$ and the target flow velocity is around $V = 0.3m/s$. n is set at 2. Thus the length of the membrane

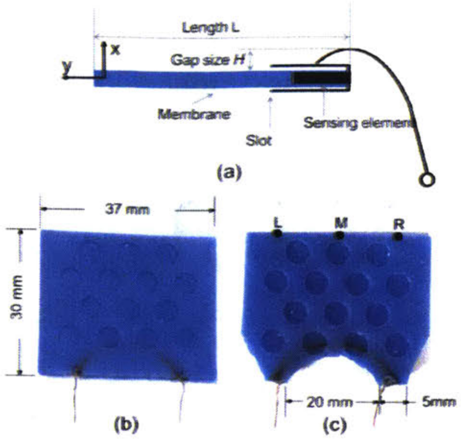


Figure 3-8: Implementations of the membrane+sensing element. (a) detail of attachment between the membrane, the sensing element and the support. (b) square membrane, low sensitivity. (c) Trimmed membrane, high sensitivity. Points L, M, R are points of application of pulling forces in the calibration process

is determined to be 30mm.

The membrane's shape and material dictate its sensitivity. Implementation in Figure 3-8-c is more sensitive than the one in Figure 3-8-b. For rubber material, its material composition and geometry affect its stiffness. MoldStar 30 rubber (membrane) shows higher stiffness than the conductive rubber cord (sensing element). Thus the stiffness of such implementations is dominated by the MoldStar 30 rubber (membrane). In (c), the portion of the membrane around the sensing element are removed, and the smallest width of the MoldStar 30 rubber at the sensing element is reduced from 37mm to 10mm. Reducing the width is similar to removing the number of springs in a multiple parallel spring system, and thus reducing the spring constant k_2 of the sensing element. Then for the same amount of strain, the implementation of Figure 3-8-c requires less stress than Figure 3-8-b, and this is validated by the calibration data in Figure 3-9. In the calibration, The conductive rubber cord, as a sensing element, converts its strain to electrical resistance change. In the calibration process, the pulling force is applied on the membranes in the y direction at the left(L), middle(M) and right(R) point as shown in Figure 3-8. As the calibration results shows, the system's sensitivity, defined as the percentage of resistance change per the percentage of pulling force change, is about proportional to the the smallest

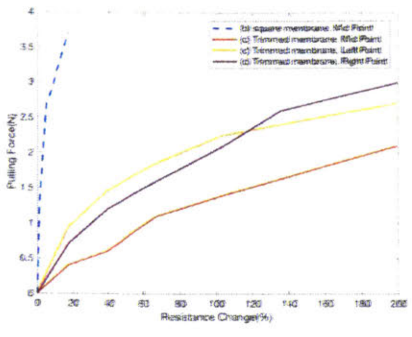


Figure 3-9: Calibration of membrane sensors in Figure 3-8-b and c.

width of the rubber at the sensing element. This is useful for calibrating the system for different sensitivity; for example, in low pressure pipes, high sensitivity is desirable. On the other hand, the system is more sensitive to leak force applied along the middle line of point M on the membrane in Fig. 3-8-c than that applied to the far sides, point R and L. The difference can be as much as a 1N offset. In the next sections, the average value of the sensitivities calibrated at point M and point L is used as the sensor's overall sensitivity value.

Simulations shows that both implementations have a much higher system bandwidth. The properties of new membrane sensors in Figure 3-8-c and also in Trace C of Figure 3-5 is listed in the second column of Table 3.1, and its simulated impulse response is shown as Trace C in Figure 3-5. Its damped natural frequency is 131Hz, much faster than that of the original sensor (Trace B in Figure 3-5). The impulse response of implementation in Figure 3-8-b, is also displayed as Trace D in Figure 3-5. Most of its parameters are the same as that of Figure 3-5-c except its higher sensing element stiffness $k_2 = k_1$. The rubber width is the same for the membrane and the sensing element part. Higher stiffness leads to higher damped natural frequency, 274 Hz. Although Figure 3-8-c was chosen for its higher sensitivity, this dynamics analysis shows it is possible to push the system bandwidth even higher.

The umbrella-like support in Figure 3-6, when built into a prototype, is a circular array of discrete pieces encased in silicone rubber. The support of a leak sensor for 52 mm inner diameter water pipe is shown in Figure 3-10. The four discrete rigid plastic pieces are like the umbrella frame (Figure 3-10-a); each of them have a slot on top and they are connected to a common hub by four shafts. Each of the slots can hold a membrane that spans 90 degrees of a circle; so four of them can cover the entire circumference of a pipe cross section for leak detection. The circumference can be divided into a more than four pieces if more accurate radial position of the leak is needed. With four pieces, the leak can be located to one of the four quadrants in the pipe cross section. One end of the membrane is connected inside the bracket on the support structure. The thickness of the bracket and the height of the bend on the support add up to a $H = 2mm$ gap between the membrane and the pipe wall (Figure 3-8). The frame of this support is enclosed in a single piece of soft rubber, as shown in the center of Figure 3-10-b. This casing is like the cloth of an umbrella, allowing the pipe flow to effectively push the robot from the back. Since the casing is a rubber, it is elastic and can additionally serve as the spring for the support (Figure 3-6). Without any loose parts, it is more robust than a set of springs. The rubber casing also waterproofs the wires inside. The complete leak sensor has a 52 mm nominal outer diameter, and it can expand and contract. The stiffness of its compression and expansion is determined by the elasticity of the rubber material. In the prototype, the rubber enclosure is made with soft silicone rubber, Ecoflex 0050 from Smooth-on Inc. It is soft enough for the leak sensor to go over obstacles in pipes.

3.4 Experimental Results

The new leak sensor was attached to a flow driven drone as shown in Figure 3-11 and tested both in the lab and in a industrial facility. The detailed design of this

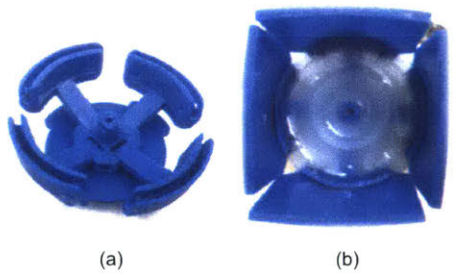


Figure 3-10: (a) The frame of the support structure (b) completed leak sensor

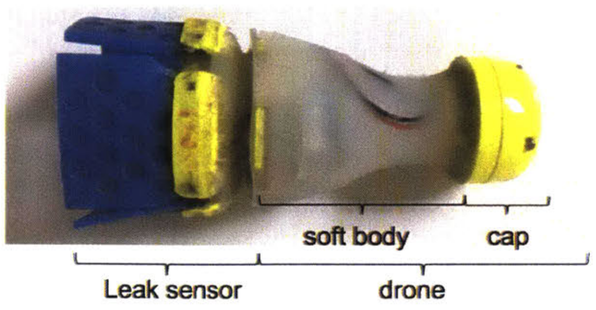


Figure 3-11: The leak sensor is attached to the back of a flow driven drone sized for 2 inch pipes.

drone and a video description can be found at http://mechatronics.mit.edu/leak-detection-system-for-city-water-distribution-systems/. Its maximum outer diameter was 52mm, and its length was 117mm. Similar to a PIG [22], it had no actuation, and it was propelled by the water flow in the pipes. Unlike regular PIGs or any other kind of pipeline robot platforms, this drone was soft. It was made of silicone rubber Ecoflex 0050 from Smooth-on Inc. The soft body could be bent, allowing the drone to turn around pipe elbows with ease. Embedded in the soft body of the drone is the electronic system for data logging purpose. Details of the electronic system can be found in the previous chapter. This particular prototype has a 3.7V Lithium Ion battery powered an Arduino Mini Pro 3.3V/8MHz micro-controller, a Pololu 9 DoF Inertia Measurement Unit (IMU), and a Pololu SD card writer. The robot has four membranes, so the leak sensor had a four channels of output, one for each membrane. Those channels were connected to the Arduino through voltage divider circuits, so

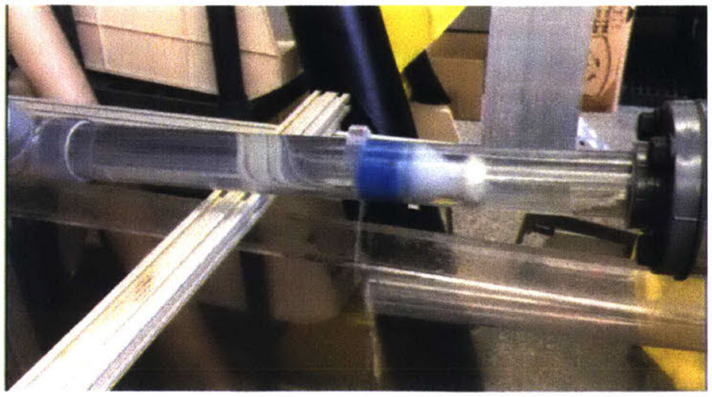


Figure 3-12: The drone and leak sensor passing a leak during a lab test.

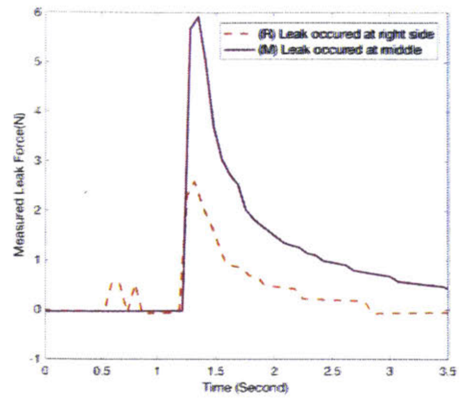


Figure 3-13: Leak sensor reading (1 channel) for the leak occurred at (R) the right side, (M) middle of the membrane, in a lab test.

the Arduino could read the voltage values on the sensing elements which were strain sensors. Resistance values of the sensing elements were then calculated. With the calibration in Figure 3-9, the magnitudes of the pulling forces on the membranes were estimated. This electronic system made the drone untethered. However, its computing power is limited; it records the motion information from the IMU(9 channels) and the leak sensor reading(4 channels) at 20Hz. The encoders in the support of the leak sensor for measuring pipe diameter changes were not implemented in this prototype, or it would further reduce the drone's sampling rate. The robot did not use wifi for real time data feed because wireless communication was unreliable in the cast iron pipes at the test site. Instead, the robot's data was downloaded all at once after the robot was retrieved from the pipe.

First, the leak sensor was tested in a lab setting to study how leak positions relative to the membrane could affect the leak sensor's output. The leak sensor calibration in Figure 3-9 clearly indicated that the leak sensor was less sensitive to leaks occurred on the far side of the membrane(point L and R in Figure 3-8-c) than on the middle (point M in Figure 3-8-c). In this test, cases of the leak occurring at point R and point M on the membrane was produced. As shown in Figure 3-12, a transparent 50mm (2 inch) inner diameter, plastic pipe with smooth interior was used for this test. A pinhole leak of 4 mm in diameter was drilled on the bottom side of the pipe. Water flow from the water tap filled up the pipe, and the line pressure was regulated

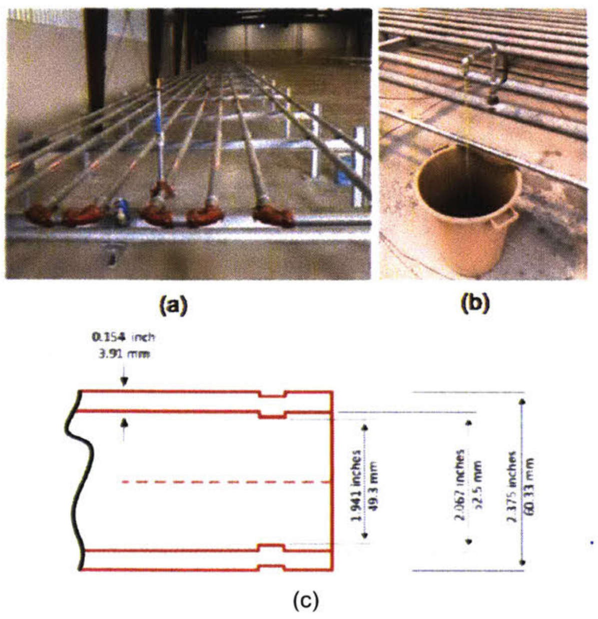


Figure 3-14: (a) The industrial pipeline used for testing (b) a leak on the pipe loop(c) schematics of the pipe segment.

at about 0.8 Bar gauge. The drone was propelled by the water and moving at about 0.1 m/s. At this speed, it took the membrane 0.3 second to pass the leak, and the leak sensor should see the leak as slower input rather than an impulse. The flow rate through the pinhole leak was measured to be about 4.1 L/min (1.08 Gal/min).The experiments were repeated with point M on the membrane (Figure 3-8-c) aligned with the leak and point R. The new leak sensor was calibrated with the average values of the calibrations for trimmed membrane measured at the middle point and the right point(Figure 3-9). The leak sensor readings from the membrane closest to the leak are plotted in Figure 3-13, with its steady state value subtracted. It proved that the leak sensor could detect leaks occurred at both the far side and the middle of the membrane, though the reading was lower for the same leak occurred at the far side.

Second, the leak sensor was tested in the industrial water pipeline (Figure 3-14-a) provided by PipeTech LLC in Saudi Arabia. The pipeline was made of cast iron, and it measured 1.5 km in length and 52mm in nominal inner diameter. The entire pipeline was in the horizontal plane. A section of 221 m long and 1.2km away from the inlet was isolated with drone launch and receive tools installed on both ends.

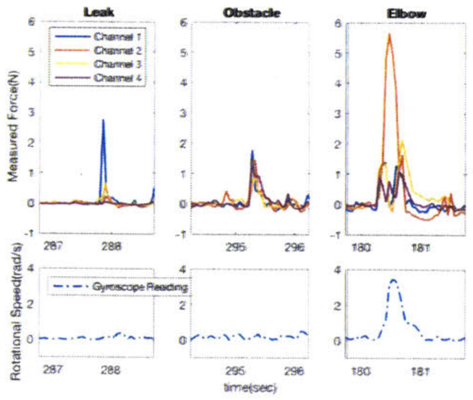


Figure 3-15: Leak sensor readings and gyroscope reading for rotational speed in the horizontal plane in three cases: 4mm pinhole leak, pipe joint as obstacle and pipe elbow.

During the test, the inlet pressure was 2 Bar gauge and the flow rate was about 0.4 m/s. With pipe head loss considered, the line pressure at the test section was about 1.7 Bar gauge.This pipeline provided enough in-pipe features for testing. The first feature was a 4 mm pinhole leak on the pipe. There was a water tap welded on top of it to turn it on and off, as shown in Figure 3-14-b. A bucket was used to collect the leaked water and measure the leak flow rate. The leak flow rate was measured to be about 5.6 L/min (1.48 Gal/min).This pipeline was constructed with hundreds of 6m long, 52mm inner diameter, metal pipe segments. At each pipe joint, there was a ring of 3mm diameter reduction, as shown in 3-14-c. Thus each pipe joint, 6m apart, was an obstacle. Additionally, there were many pipe elbows. The pipeline had been in service for 6 years so there was a small amount of rust and dirt inside.

The test result showed that various in-pipe features could be differentiated from the measured signals. The drone travelled through the 221m long test section in an average time of 550 seconds. The average speed was 0.4 m/s. The drone completed 2 tests, and in each test, it passed 1 leak, 4 pipe elbows and 41 pipe joints and recorded data for all of them. The readings from the leak sensor for these features were plotted in Figure 3-15. Since leaks were expected to show up as high frequency, impulse-like signals, only the changes in pulling forces were studied. The steady-state values of the leak sensor readings were removed with a high-pass filter (2Hz cutoff frequency).

At the leak, the leak sensor registered a high frequency, large magnitude change. This change was only in the channel corresponding to the membrane that was right on top of the leak, and the changes were much smaller for the other channels since their membranes didn't touch the leak. The frequency of the observed signal for the leak was 10Hz, which is at the aliasing limit for this 20Hz sampling rate drone. If the drone has more computing power, it may be able to observe more content in the leak signal.

At the obstacles on pipe joints, the leak sensor registered slower changes than it did at the leak. Since the diameter reduction at the joints was on all sides of the pipe(Figure 3-14-c), all four channels detected changes. The signal at one of such obstacle was shown as thc main peak in the mid plot of Figure 3-15. The dominant frequency was about 4Hz. The average magnitude was 1.2N and the standard deviation was 0.7N. However, its distribution was approximately a uniform distribution.

The drone can easily spot pipe elbows. The gyroscope part of the on-board IMU measured the drone's rotational speed in the horizontal plane, as shown in the bottom row in Figure 3-15. The gyroscope plots here were low pass filtered (5Hz cut off frequency) for reducing the noise. At the elbow, the drone registered a significant change in its rotational speed. At leaks or obstacles, the rotational speed change was multiple order of magnitude lower. The leak sensor measured multiple oscillations in all four channels at the elbow, while single pulses at leaks and obstacles. These oscillations are off similar dominant frequency as that of the obstacle signal.

3.5 Discussion

As predicted with system modeling, a leak sensor with a high bandwidth was able to detect leaks in an operating water pipe. During the test at the industrial facility, both the new leak sensor and the water flow were moving at V=0.4 m/s. At this rate, the membrane of length L=30mm was on top of the leak for about L/V=75 ms. The membrane could react to the leak for the entire 75 ms or only a fraction of it. The leak input to the system could be modelled as an impulse function of duration

T. Depending on the size of the leak and the robot's speed in the experiment, this duration T was estimated to be between 10 ms to 75 ms. In response to this leak input, the drone recorded a 100 ms (10 Hz) impulse output from the leak sensor. This was a good indicator for leaks, but much information about the leak that was not captured. The drone had a sampling rate of only 20 Hz since its Arduino micro-controller had limited computing power. Any signal of higher than half sampling rate cannot be accurately measured due to aliasing. Thus the actual leak response should be faster than 10 Hz. By design the leak sensor had a damped natural frequency of 131 Hz. With a more potent micro-controller such as Raspberry Pi to increase the sampling rate above 260 Hz, the drone can surely record a more accurate leak responses with rich details. Then the exact duration of the leak input can be determined, and how to design the membrane geometry to capture leaks most effectively can be studied.

The new leak sensor is not ideal yet, since there are false alarms. The leak sensor directly measures the pulling force, or equivalently the increase in friction force on the membrane. Leaks can cause the increase in friction force, so can obstacles and pipe elbows as indicated in Figure 3-15. Leaks from those false alarms need to be differentiated, possibly through data fusion and frequency domain analysis. For example, additional sensor can be added to monitor the pipe diameter change and thus indicate obstacles. With the IMU in the drone, rotational speed of the robot can be monitored to detect pipe elbows. There are also four channels on the leak sensor; small leaks triggers one channel of the leak sensor, while the elbow triggers all four channels. Correlation among those data streams can be useful to eliminate some false alarms. More details of this correlation approach can be found in Appendix B. Another approach is transient response analysis. As described in the Experiment Section, the leak, obstacle and elbow have different transient response on the leak sensor output. The leak sensor's response was damped more in the case of leaks than in the cases of obstacles and pipe elbows. It has the most oscillations at the elbow. The response to leaks had the highest damped natural frequency among the three cases. Further study could lead to a method to differentiate leaks and false alarms from those frequency domain patterns.

Chapter 4

Telling Leaks Apart From False Positives: A Low Cost Soft Bending Angle Sensor

4.1 Overview

In this chapter, I present the design, fabrication and validation of an improved leak detector that can robustly differentiate leaks from false positives. Each individual fin sensors on the detector can differentiate leaks from obtrusions independently. Leaks bend the sensor down and pull, obtrusions bend the sensor up. The sensor, through measuring the bending direction and angle, can tell them apart. Such sensor is constructed with low-cost ordinary material in a low-cost, scalable fabrication process. The outcome is a 1-dollar solution to robustly detect leaks and obtrusions in pipe distinctively at the same time. The applications of this low-cost soft bending angle sensor are not limited to leak detection; an example in fish tail motion tracking is also demonstrated.

4.2 Motivation: Differentiate leaks from false positives

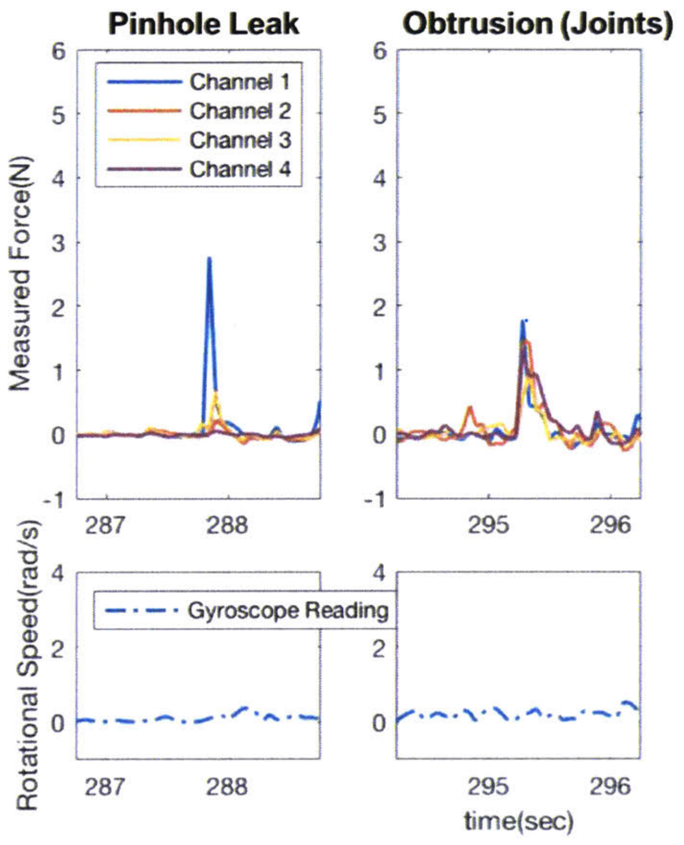


Figure 4-1: Challenges in Differentiating Leaks and Obtrusions with Simple Leak Sensor Illustrated with 2017 Saudi Arabia field test result

The first-of-its-kind leak detector for operating water pipes presented in Chapter 3 still had one imperfection: false positives. To the leak sensor, obtrusions causes false positives. From the results(Fig. 4-1) from the field tests in Saudi Arabia, leaks and obtrusions appeared to be very similar in individual sensor outputs, and it could only be differentiated with multi-sensor data correlation (Appendix B). It worked because, in the field test, the main type of obtrusions were pipe joints, and the only type of leaks were pinhole leaks. The pipe joints pushed inward the membrane sensors on all sides and caused readings on all sensors. The pinhole leak could only pull on one piece of the membrane sensors that is the closest to the leak, so it caused readings on only one sensor. The underlining assumption is that leaks are small and pull on one membrane sensor, while obtrusions compress the entire detector on all sides and bend all membrane sensors. If there were radial leaks or pipe breaks where the cracks along the pipe circumference were long enough to be detected by multiple membrane sensors, this approach may register the leaks as false obtrusions. If there were small pieces of obtrusions that only triggers one piece of the membrane sensor, this approach may register the obtrusions as false leaks.This correlation approach to differentiate leaks from obtrusion is not robust and only works conditionally.

Leaks and obtrusions interact with the membrane sensor in different ways, so it should be possible for the sensor to differentiate them. As illustrated in Fig. 4-2, a leak bends the membrane down, toward the pipe wall given the suction force, and then pull on the membrane given the friction force. In comparison, an obtrusion displaces the membrane up, away from the pipe wall. If the membrane sensor generate different outputs for bending up and pulling down, then from its output we can tell leaks and obtrusions apart. It can be used to detect leaks while rejecting the false positives from obtrusions. Then this will be a robust leak sensor. It can also be used to detect both leaks and obtrusions at the same time, with distinctively different outputs. Now this becomes a multi-purpose sensor.

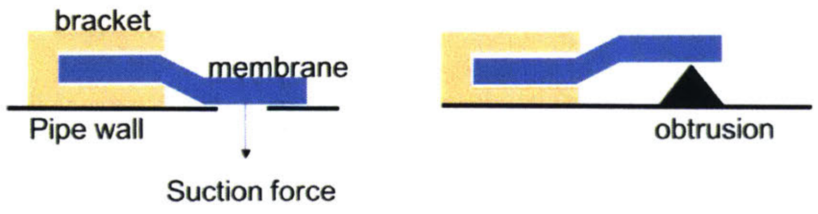


Figure 4-2: Different Dynamics: Leaks bend the sensor down and pull, and obtrusions bend the sensor up.

I am going to build this multi-purpose soft bending angle sensor. Following the theme in this leak detection robot–low cost, robust and practical, the following design requirements are defined:

1. A Single-piece, fully integrated soft membrane sensor that can output distinctively different signals for bending up and bending down

2. Minimum number of sensing elements and connections

3. Build with low cost material

4. Build in a low cost but scalable fabrication process

4.3 Method: Engineering the Neutral Axis

Bending angle sensors can typically be built with carefully chosen placement of the sensing element with respect to the device's neutral axis. The neutral axis of a cantilever beam is a plane along which there is no elongation or compression when the cantilever beam is bent. The membrane sensor in the leak detector can be treated as such a cantilever beam. As illustrated in Fig. 4-3-a, one end of the membrane sensor is fixed in the yellow bracket, and the other end is free and can be bent. When the membrane sensor is bent downward, the part of it above the neutral axis will be stretched and the part below the neutral axis will be compressed. As the membrane sensor is made uniformly with the same rubber material, the neutral axis is then right at the center height of the entire device, and the strain distribution is symmetric about the neutral axis but in opposite directions. The placement of the sensing element determines what its output indicates.The sensing element can be a electrically conductive rubber. It increases electrical resistance while experiencing elongation, and it decreases electrical resistance while experiencing compression. Its sensitivity in the two directions may differ. When this sensing element is placed on the neutral axis such as in Fig 4-3-b, it will experience equal amount of elongation and compression during a bending motion. It cannot reliably tell which direction the sensor. In contrast, if the sensing element is placed far above the neutral axis such as in Fig.4-3-c, it will experience elongation and increase its resistance when the device is bending downward, and it will experience compression and thus decrease in resistance when bending upward. The reserved relationship applies to the case when the sensing element is placed on the opposite side of the neutral axis. With this kind of input and output relationship (Fig. 4-4), one can simply measure the direction of resistance change to estimate the bending direction and the input. Bending down the device in Fig. 4-3-c leads to elongation on the sensing element and thus increase in resistance. Bending up the same device leads to compression on the sensing element and thus decrease in resistance. If the membrane is estimated to be bent up, the input must be an obtrusion. If the membrane is estimated to be bent down, the input must be a leak. There could even be two sensing element, one on each side of the neutral axis of the device as illustrated in Fig. 4-3-d. In this case, the output would be the difference in resistance between the top sensing element and the bottom one. When the change in the difference is positive, the device is bending down. When the change in the difference is negative, the device is bending up.

To build a soft bending angle sensor that has the performance of Fig.4-3 may not sound difficult, but to build a thin membrane thickness one in a low cost way is a significant challenge. The challenge is the size constraint and commercial availability of the material. One of the key requirement for this type of bending angle sensor to work is that the sensing element must be precisely placed on one side of the neutral axis. Thus the thickness of the sensing element, h, must be less than a half of the thickness of the entire device, H. In the leak sensor case, H=2mm. To build the bending angle sensor in this way requires a sensing element of thickness h<1mm. At the time of this work in late 2017, electrically conductive rubber with less than 1mm thickness are not commercially available. In comparison, the thinnest sheet of this type of material available on the Internet is 1.5mm, and it is less than 15 US dollars per square foot (equivalently 0.016 US dollar per square centimeter). If this conductive rubber is to be used in this 2mm-thick membrane sensor directly as shown in Fig.4-5, it will experience a combination of compression and elongation when the entire device is bent rather than only compression or elongation separately. This renders the bending angle sensor ineffective. The original membrane sensors were built this way, and each individual membrane sensor cannot produce distinctively different signal for leaks and obtrusions as shown in Fig. 4-1.

The state-of-the art technique to produce this type of membrane-thickness bending angle sensor requires high precision, high cost manufacturing equipment and process.

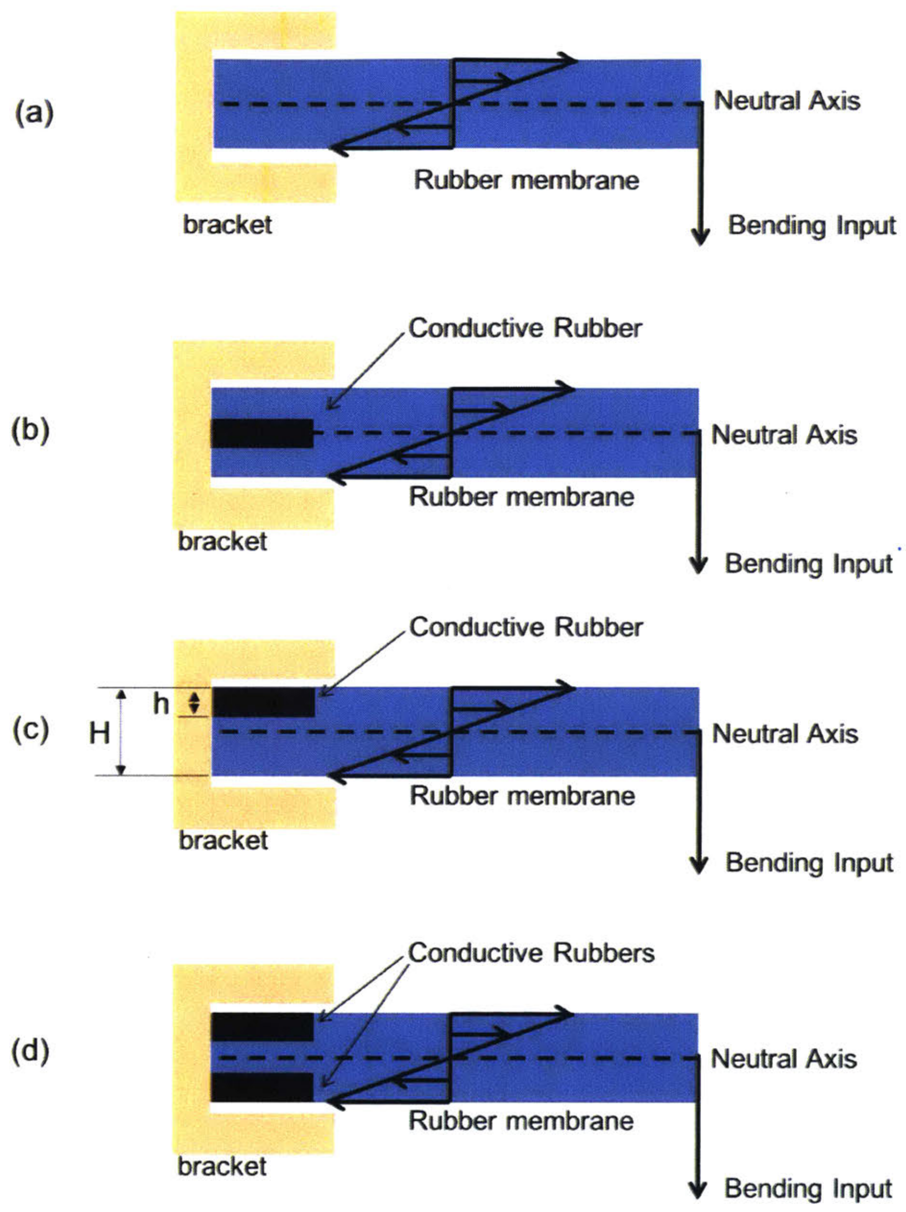


Figure 4-3: An typical soft bending angle sensor construction: Placement of sensing element with respect to the neutral axis matters. The horizontal arrows indicate the direction and magnitude of shear stress and strain.

Most of the techniques in literatures originated from Micrometer-scale Microelectromechanical Systems(MEMS) applications, and researchers at Harvard University is pioneering in the field [2, 3, 4]. The popular practice is to print a thin layer of conductive sensing element on the surface of the device via photo-lithography[2], or hybrid 3D printing [3]. The device can also be 3D printed with materials of different property at different depth[4]. It allows fine control over the local property such as

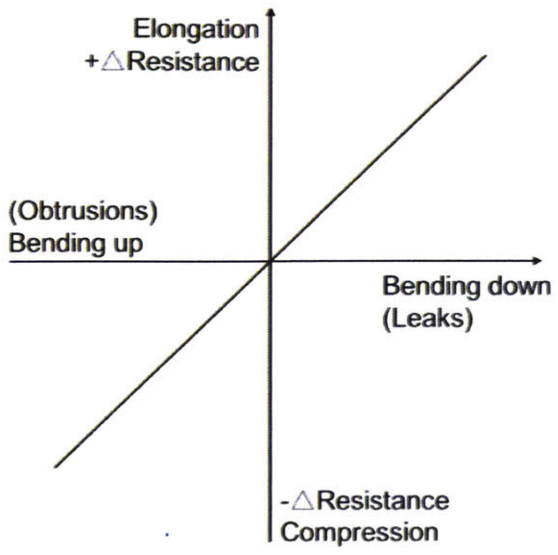


Figure 4-4: Ideal output from a soft bending angle sensor.

stiffness within the three-dimensional space of the device, and produces high performance sensors. However, these manufacturing processes are of high precision and require expensive machines. To a sensor of area size on the order of 10 millimeter, micrometer level precision may be excessive. 3D printing devices layer by layer is also slow and not scalable, especially when making large surface area devices. In this project, our goal is to design and build low cost, scalable and practical devices for leak detection. The state-of-art fabrication technique does not meet our low-cost and scalability requirements.

In contrast to the conventional high precision, expensive ways, I present an extremely low-cost approach to produce a millimeter-thickness soft bending angle sen-

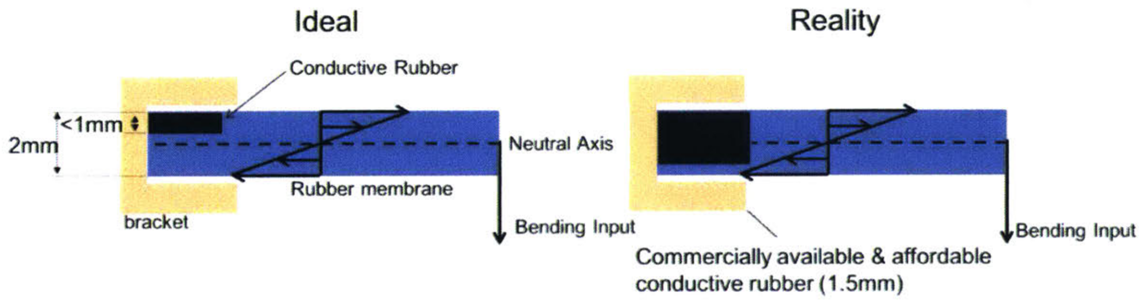


Figure 4-5: Difference between Ideal and Reality of a typical soft bending angle sensor: thin, commercial conductive rubber is not available at an affordable price.

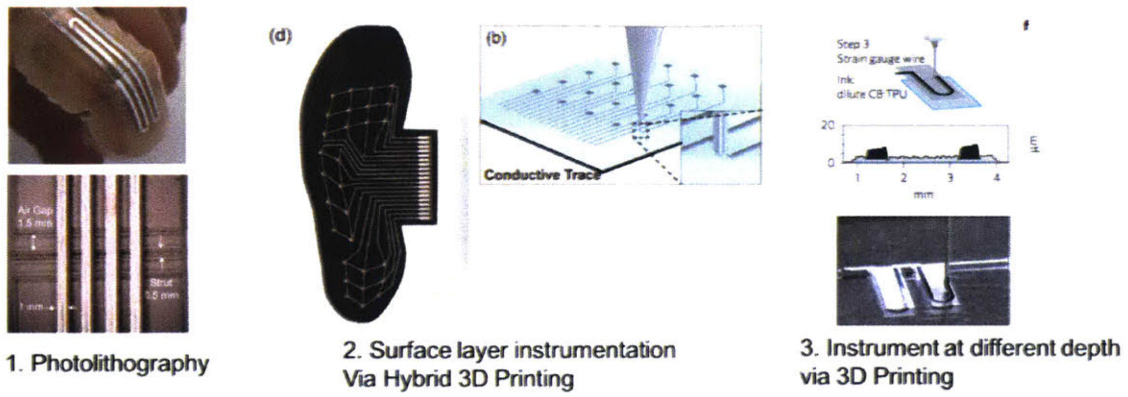


Figure 4-6: Three state-of-the-art manufacturing techniques for soft, thin bending angle sensor: Photolithography[2], hybrid surface 3D printing[3], hybrid depth 3D printing[4]

sor. In the typical construction of the soft bending angle sensor, the neutral axis is fixed by the geometry and material first and then the sensing element is placed on one side of it. We can engineer the neutral axis of the device to be on one side of the sensing element, by simply adding a layer of material of higher stretch stiffness on one side of the sensing element, spanning from one end of the device to the other. This new material can be a stiffer rubber, but it then requires a multi-step molding process to produce this membrane sensor. There is another low cost, commonly available, ordinary material and we can simply bond it inside the rubber membrane sensor in a one-step manufacturing process. This material is fabric.

Fabric is a great choice for engineering the neutral axis in a soft membrane sensor. The first advantage is the non-isotropic material and structural property of the fabric. Woven fabric can have very high stiffness in stretching and appear to be almost unstretchable. Meanwhile it is extremely soft and easy to bend and fold. The hierarchical structure of the fabrics also make them very easy to bond to soft rubber material during the molding process. Given the high contrast in the high stretch stiffness of the fabric and low stretch stiffness of the other soft material filler, this bending angle sensor's neutral axis is no longer in the center of the device but in the proximity of the interface between fabric and the soft material, as illustrated in Fig. 4-7. When the device is bending down, the sensing element is stretched and

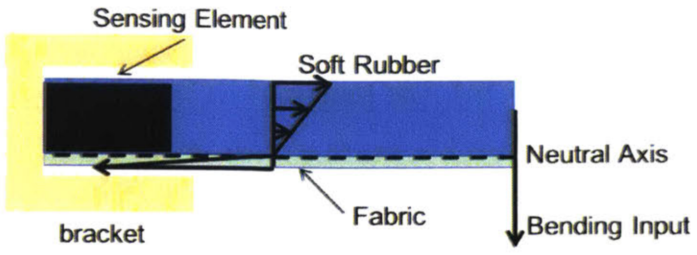


Figure 4-7: The proposed design of a low cost, membrane-thickness soft bending angle sensor.

produce an elongation signal. When the device is bending up, the sensing element is compressed and produce a compression signal. If the fabric is very stiff in the stretch direction, such design will isolate the sensing element from any horizontal stretch load.

The advantage of this new design is low cost and low requirement on manufacturing precision. The high contrast of stretching stiffness will dominate the placement of the neutral axis and thus the sensing element's ability to differentiate bending up and downs. The sensing element is no longer required to be thin which need precise and expensive manufacturing process. Very thin fabrics are widely available. This design allows us to use thick but commercially available conductive rubber to build previously impossibly thin soft bending angle sensors. Moreover, the performance of the device by design is consistent and irrelevant to the manufacturing imprecision in either the sensing element or its placement in the device.

In the next two sections, The principle of this sensor will first be demonstrated through a simplified fish tail motion tracking application. This design has high tolerance toward the errors in the manufacturing process. This aspect will be demonstrated in leak detection applications.

4.4 Demonstrate Principles of The Soft Bending Angle Sensor in A Fish Tail Motion Sensing Application

4.4.1 Motivation: Effective Minimum Sensing For Under-actuated Miniature Soft Robot Fishes

Soft material has enabled the design of many bio-inspired robots in unconventional ways. For example, the soft robot fish [5, 41] can swim like a real fish, and do so in an under-actuated way. The robot fishes are about 10cm long, and they have only one servo motor inside to control the swing frequency and magnitude of the fish body. Through controlling the change in the swing motion, this single-actuator robot fish achieved two degrees of freedom in motion: it could swim forward and backward, and turn left or right, in a controlled way. Since the robot fish is small in size, it requires a low number of actuators. The under-actuated control made it feasible. It leads researchers to think if it is also possible to instrument a small soft robot fish with a minimum number of sensors while still able to perform feedback control. The sensors will enable the robot to control its motion. In literature, the minimum requirement of the number of sensors reported is two [42].

With my soft bending angle sensor, I believe we can instrument the smallest robot fish with only one sensor and enable feedback control, and do so with minimum cost. A single bending angle sensor attached to the side of the robot fish will be able to tell both the direction and magnitude of the fish body swing motion. Then the fish robot can know if it is subject to external turbulence and adjust its actuator output accordingly. That will be a single actuator, single sensor robot fish for controlled swimming in changing flow conditions.

Figure 4-8: Prof. Kamal Youcef-Toumi holding two examples of under-actuated soft robot fishes[5]. Question: can we also achieve under-sensing on these robots?

4.4.2 Setup Experiments to Validate Two Hypotheses

To demonstrate the effectiveness of the proposed low-cost sensor in the simplistic manner, two versions of instrumented fish tails are fabricated and compared in performance. Both fish tails are very thin and small, with 2mm in thickness and 5cm in length. One fish tail is made with only the non-conductive rubber and the 1.5mm-thick conductive rubber as the sensing elements, as shown in Fig. 4-9. The non-conductive rubber used here was Smooth-on Mold Star 30 which has a Shore A hardness of 30. When compared to standard hardness silicone rubber from Smooth-on, it is estimated that the Shore A hardness of the purchased conductive is around 30-35, similar to that of the non-conductive rubber. In all fish tails, the conductive rubber was cut into U-shape sensing elements with dimensions shown in Fig. 4-10. This is the smallest size one can cut the rubber into manually with scissor. It will require additional tools to cut even smaller sensing elements. Two braided electrical wires are stitched to the sensing element at the two tips of the U shape to form a circuit. In the other fish tail, a layer of very thin woven fabric is embedded and molded into the back of the non-conductive rubber. As shown in the cross-section view of the fish tails in Fig. 4-11, the rubber-only fish tail has its neutral axis along the center line of the entire device. The sensing elements placed off-center so it is possible for them to sense the difference in the entire device bending up or bending down. In the rubber-fabric fish tail, the neutral axis is near the interface between the fabric and the non-conductive rubber which is above the top of the sensing elements. In this way, when the fish tail bends downward, the sensing elements will sense compression and reduce their electrical resistance values. When the fish tail bends upward, the sensing elements will sense elongation and increase their electrical resistance values. The sensing elements' resistances are measured and recorded.

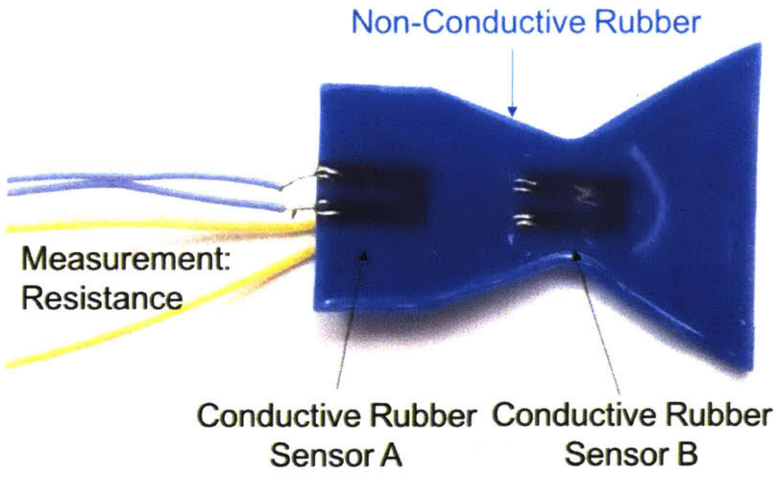
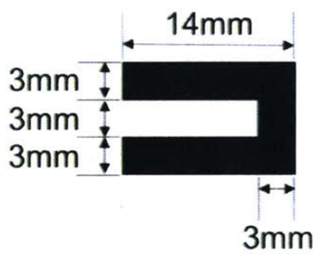


Figure 4-9: Soft fish tail with soft bending angle sensors embedded

Figure 4-10: Geometry of the sensing element inside the fish tail. It is cut out from 1.5mm thick conductive rubber



In the experiment, two hypothesis are to be tested. The first hypothesis is that

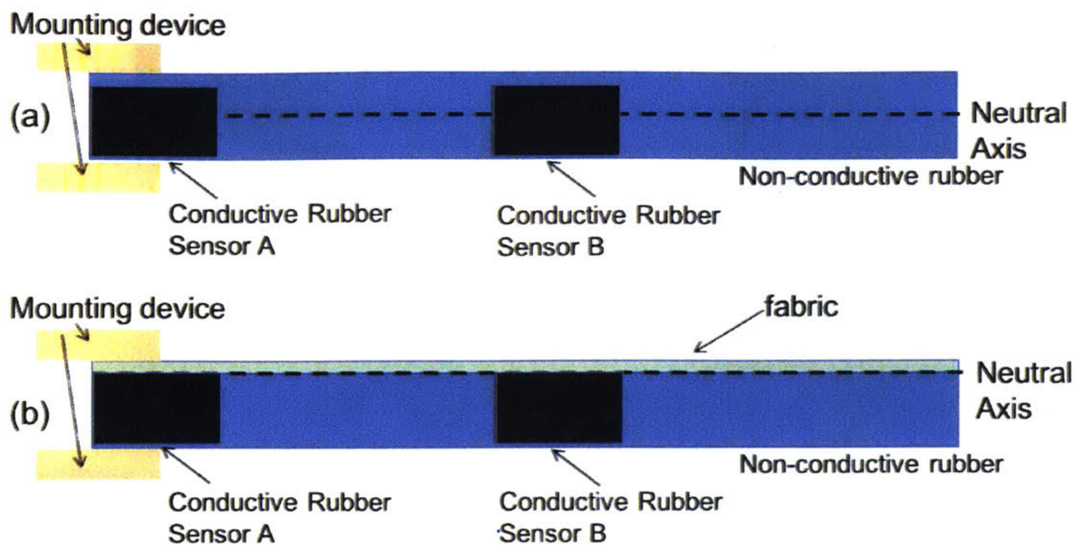


Figure 4-11: Cross-section view along the longitudinal direction of the two soft fish tails. (1) rubber only version (2) rubber-fabric version.

the inclusion of a fabric effectively enables building thin soft bending angle sensors without even thinner sensing elements. For this hypothesis to be true, the rubber-fabric fish tail should clearly show more differentiable sensor outputs for bending up and bending down. In comparison, the rubber only fish tail should show none or little differentiable outputs for bending up and down.

The second hypothesis is that this directional sensing capability is not affected by other external interactions. In the experiment setups as shown in Fig. 4-12, the left end of the fish tails are mounted in between two plates. when the fish tail bends, it may press against the corner of one of the plates and thus cause changes in the sensor outputs. To ensure the differentiable sensor outputs are affected mainly by the neutral axis rather than the mounting device, two independent sensing elements are placed in each fish tail. Sensor A is at the left end of the fish body where the fish tail will be mounted, so it will be affected by the change in normal pressure from the mounting tool. Sensor B in the middle of the fish tail, unaffected by any normal pressure. This one serves as the benchmark.

The experiments on the fishtails were conducted in three steps: bend, release, record. The fishtails was held in-between two clear acrylic sheets, as shown in Fig.

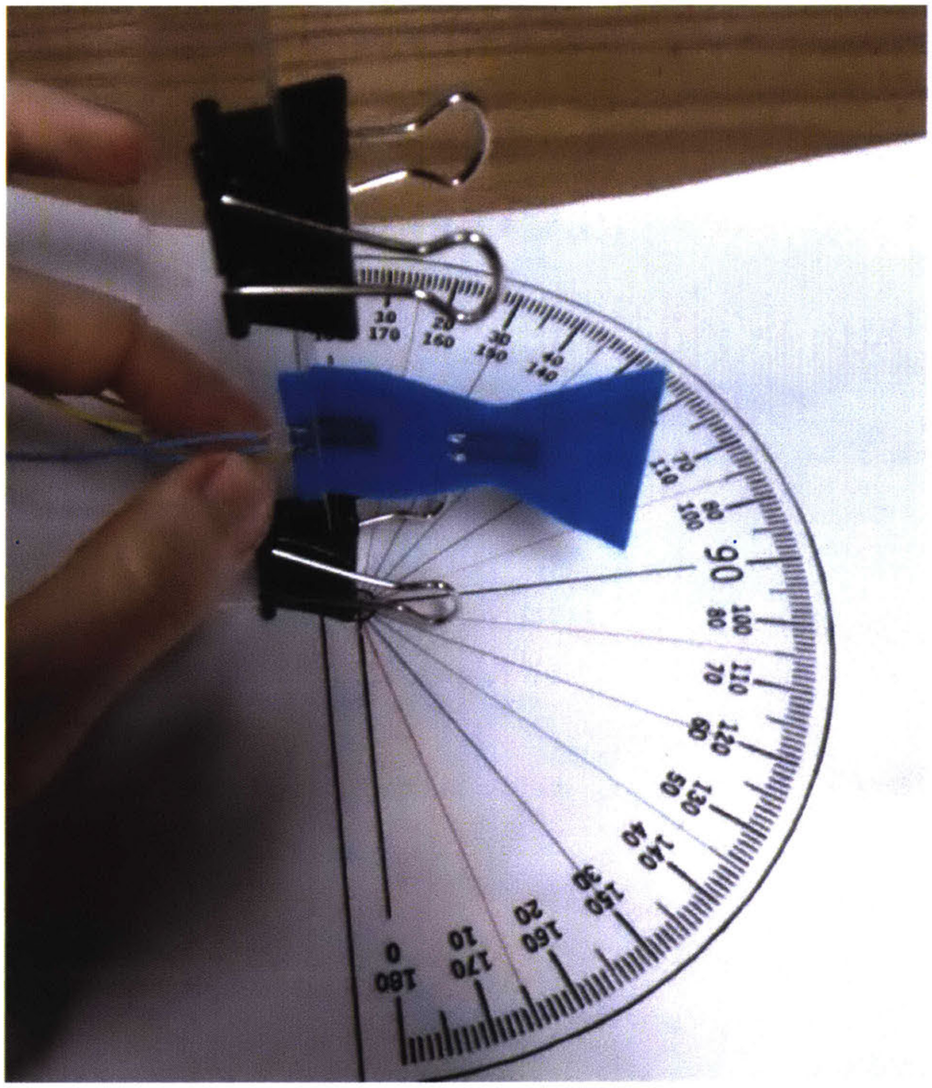


Figure 4-12: Setup for the fish tail experiment

4-12. Two clippers were used to held the acrylic sheet and the sensing element tight. The total force exerted by the clippers was measured with a Vernier dynamo-meter, and it was 50N total over a 5mm by 30mm area on the fish tail. The fish tail was then bent to 15 degree into the page in Fig. 4-12 slowly within 2 seconds and then released. The angular displacement was measured with the angular ruler as indicated in the figure. We denoted this test as bending up 15 degrees. Similar tests were performed for bending up 30, 45 degrees, as well as 15, 30, 45 degrees in the opposite direction. The tests in the opposite direction were denoted as bending down. The resistance value outputs were recorded by a micro-controller (Arduino Mini Pro 328) at a sampling rate of 50Hz.

4.4.3 Experimental Results

The experimental results clearly validated the first hypothesis. We compared the output of Sensor B in the rubber only fish tail (Fig. 4-13) and the same sensor in the rubber-fabric fish tail (Fig. 4-14. We found the device with fabric in it clearly indicated distinctive and different measurements in bending up and down. Sensor B, being in the middle of the fish tail, experienced only bending moments during the tests. In the rubber only fish tail, the sensor B showed small change during the bending phases, but the difference between the output of bending up and that of bending down was minimal. This is as expected since the sensing element overlapped both sides the neutral axis of the fish tail (Fig.4-11, a). It does not work as a bending angle sensor if the commercially available thick conductive rubber is simply embedded in the thin, rubber-only fish tail.

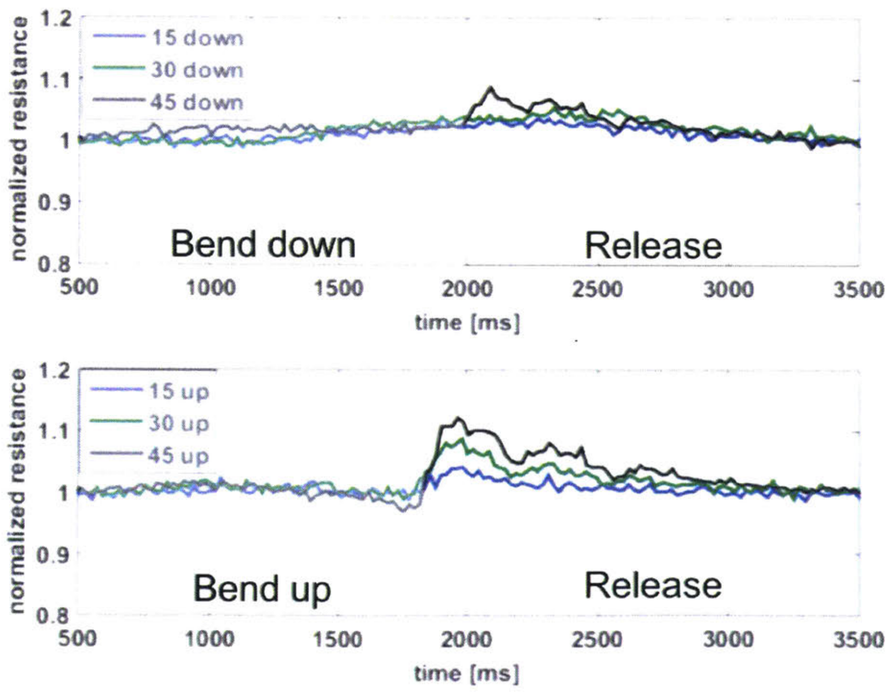


Figure 4-13: Rubber only fish tail, Sensor B output

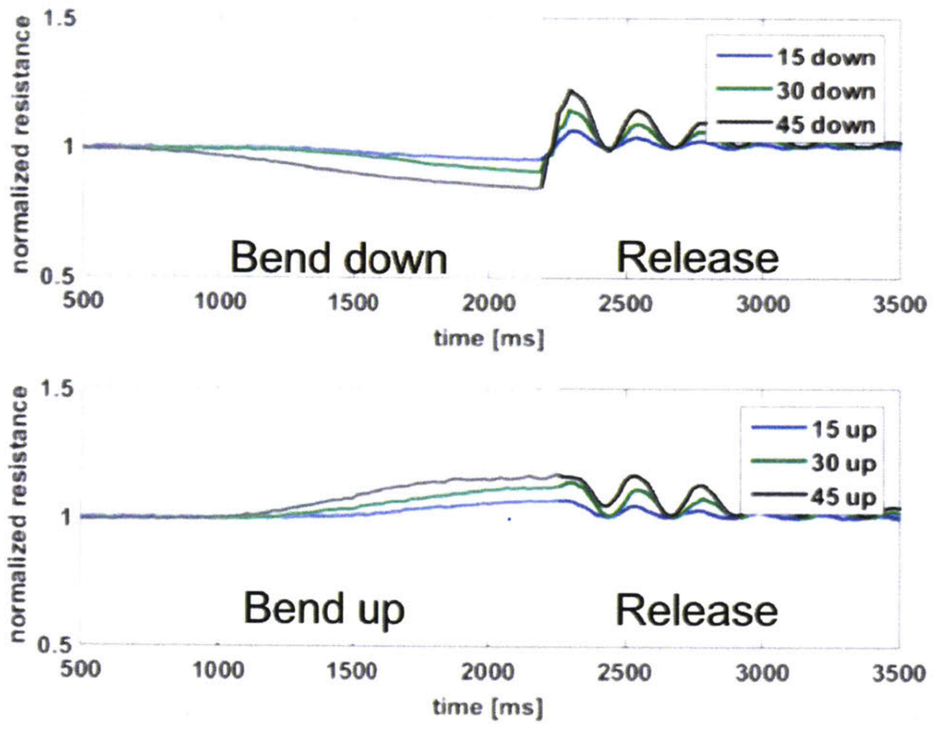


Figure 4-14: Rubber -fabric fish tail, Sensor B output

In contrast, the same sensor at the same position in the rubber-fabric fish tail displayed clearly different outputs for bending up and bending down modes. As shown in Fig. 4-11, the fabric was embedded in the top side of the sensor (into the page in Fig. 4-12). Thus when the fish tail was bent downward (out of the page in Fig.4-12), the sensing element was predicted to be compressed and its resistance reduced. When the fish tail was bent upward (into the page in Fig. 4-12 and toward the fabric), the sensing element was predicted to be stretched and its resistance increased. The measurement indicated in Fig. 4-14 agreed with the prediction. In comparison to the same conductive rubber sensor in the fish tail of the same thickness but no embedded fabric, the fabric version could tell not only the direction of the bending motion, but also the magnitude of the bending, as indicated in Fig. 4-14. Through embedding a ordinary fabric, we successfully relocated the neutral axis of the entire device. We used a single piece of commercially available, low cost but thick conductive rubber in this thin fish tail to enable bending angle and direction measurement. There is no need to fabricate thin, sub-millimeter thickness conductive rubber sensors to make such a bending angle sensor. It is not the thickness of the sensing element and the device that determines the effectiveness of the bending angle sensor. What really matters is the placement of the sensing element with respect to the device's neutral axis.

The experimental results showed the additive effect from the device's interaction with the mounting tool on the sensor outputs. Because the fish tails are soft, local deformation is higher at the mounting location and gradually reduces toward the tip of the fish tail. Sensor A was placed at the mounting location while sensor B was placed at the middle of the fish tail. Thus, in comparison to Sensor B, Sensor A was expected to be compressed more and produce a larger magnitude resistance drop when the fish tails were bent down. When the fish tails were bent up, Sensor A was expected to be stretched more and produce a larger magnitude resistance increase. These were reflected by the overall trend in the experimental results. When sensor A output in the rubber-only fish tail Fig. 4-15) is compared to that of sensor B (Fig. 4-15) , it was visible that sensor A was able to tell the bending direction much better than sensor B. Being at the pivoting point of the cantilever structure, Sensor A experiences much bigger local deformation than Sensor B in the middle of the cantilever. Similar trends were also observed in the rubber-fabric fish tail when comparing Fig. 4-16 with Fig. 4-14. However, there is a noticeable difference in Sensor A's output and Sensor B's output when the fish tails were bent up. Sensor A, during the bend-up tests, demonstrated a two-phase effect. As Fig. 4-15 indicates, after an initial phase of increasing resistance as sensor A was stretched, the resistance dropped gradually. The resistance drop while the fish tail was bent further up indicated that compression was more significant than elongation at Sensor A. The resistance drop kicked in earlier when the input bending angle was larger. The question was then where the compression came from.

The mounting tool adds an additional normal pressure to this process. The corner of the mounting tools would press against the fish tail and the sensing element in it in the same way regardless of which direction the fish tail bent. In the case of the

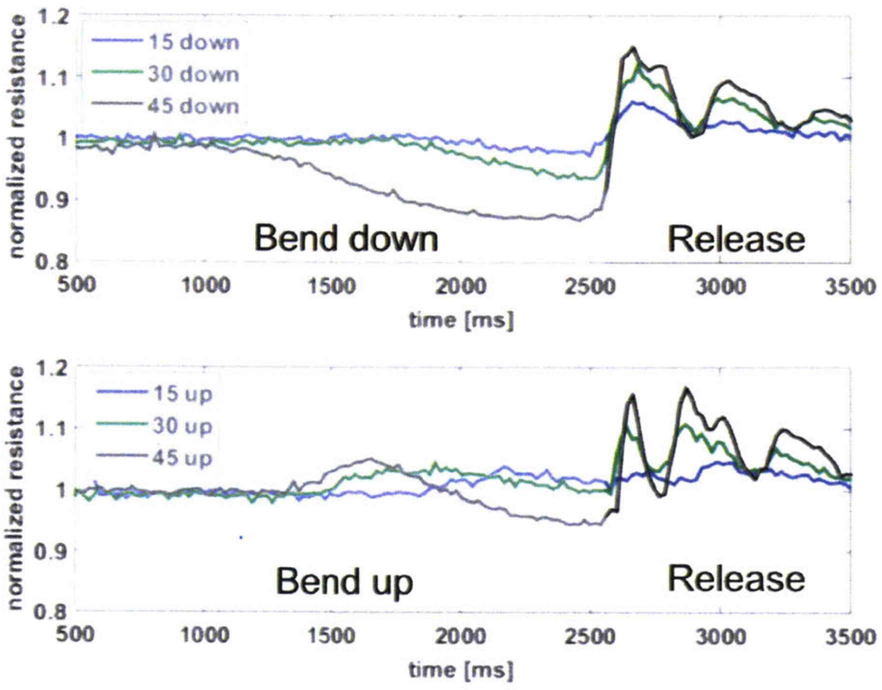


Figure 4-15: Rubber only fish tail, Sensor A output

rubber-only fish tail, sensor A was compressed by the corner of the mounting tool on the top (as indicated in Fig. 4-11 when the fish tail was bent up. This compression was concentrated and it was in the normal direction rather than the longitudinal direction of the fish tail. This increase of normal pressure on the conductive rubber sensor would cause a reduction in the sensor's resistance output. While the fish tail was bending up, the normal pressure from the mounting tool occurred only after the fish tail was bent over a certain angular threshold. Before the threshold, Sensor A was mainly stretched and its resistance increased. After the threshold, the normal pressure increased faster than the stretch on Sensor A, and thus the resistance of Sensor A reduced. This explained the two phase effect in the bend up phase on the rubber-only fish tail (Fig. 4-15). In the case of the rubber-fabric fish tail, the resistance drop phase was less significant because the fish tail was stiffer. The inclusion of the fabric on the top layer of the fish tail increased the bending stiffness of the fish tail. The concentrated normal compression on top did not effectively reach the sensing element on the bottom side of the neutral axis. The same normal compression

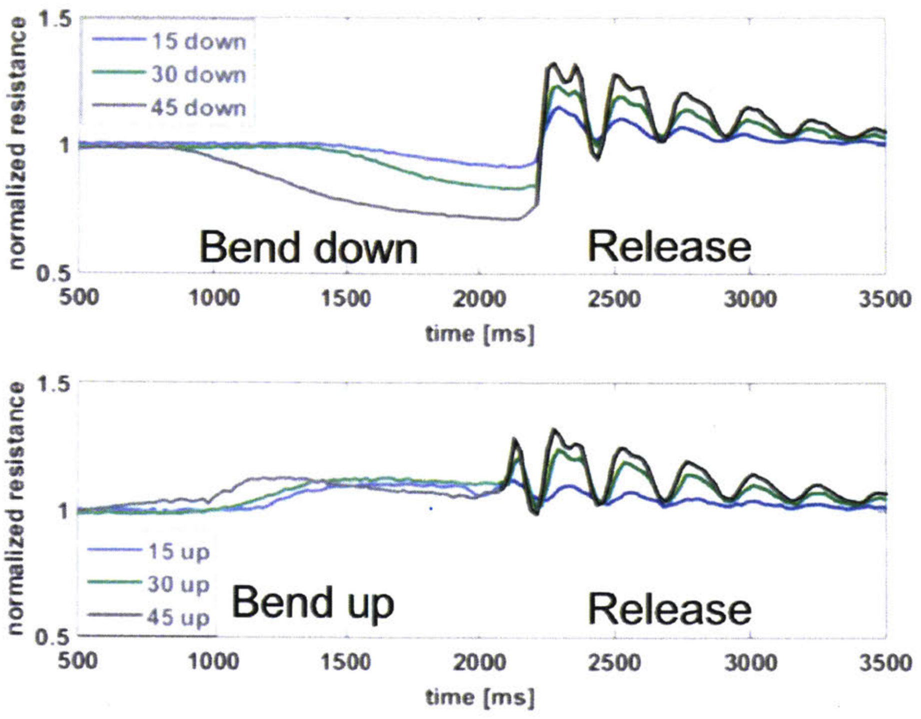


Figure 4-16: Rubber -fabric fish tail, Sensor A output

was also expected to occur when the fish tails were bending down; the corner of the mounting tool in the bottom would compress Sensor A and reduce its resistance. However, Sensor A was already compressed due to the downward bending moment on the fish tail. These two compression effects added to each other, and Sensor A outputted a larger, single-phase resistance reduction in response to the downward bending moment on the fish tail.

4.4.4 Summary: Effective Low-Cost Way to Break The Thickness Limit For Soft Bending Angle Sensors

The design of the low cost soft bending angle sensor was demonstrated in a simple fish tail motion tracking application. To instrument a 2mm thin fish tail, conventional design requires a sub 1mm thin conductive rubber that are not commercially available and hard to fabricate precisely. However, it is not the thickness of the sensing element and the device that determines the effectiveness of the bending angle sensor. What really matters is the placement of the sensing element with respect to the device's neutral axis. Through the inclusion of more stiff material in the fish tail, such as ordinary fabrics, the location of the neutral axis in the device can be designed in a low-cost way. It also significantly reduced the precision requirement in manufacturing those sensors. Because the material are all ordinary and widely available, bending angle sensors made this way cost less than $1 each. A single rubber-fabric bending angle sensor allowed us to track both the direction and the magnitude of the fish tail displacement due to external forces. It is an effective approach to instrument robot fish and it only needs one of these sot bending angle sensors.

4.5 Demonstrate High Manufacturing Tolerance of The Soft Bending Angle Sensor In The Application Of Leak Detection

4.5.1 Use Soft Bending Angle Sensor In Leak and Obtrusion Detection

Based on the design principle of the soft bending angle sensor presented in the Fish Tail section, new membrane leak sensors were designed and fabricated. As shown in Fig. 4-17, the new leak sensor was still 2mm in thickness, with 1.5mm thick electrically conductive rubber embedded inside as the sensing element. Underneath the conductive rubber is a layer of fabric, indicated by the green region. The rest of the leak sensor is made with soft silicone rubber; more specifically, Smooth-on Mold Star 30. This silicone rubber and the sensing element shared similar stiffness. A layer of fabric was embedded inside the soft silicone rubber, right underneath the sensing element. When viewed from the top, this fabric layer has to be the exact same shape and size as the entire sensor, as shown in the Bill of Material in Fig. 4-18. The sensing element was cut into a U shape, so that two wires could be connected to each end of the sensing element allowing the measurement of electrically resistance. The end of the leak sensor where the sensing element resided was placed inside a yellow loose bracket. As shown in Fig. 4-17, the left end of the sensing element was bonded with super glue to the vertical wall of the bracket. With this configuration, any kind of deformation on the right hand side of the leak sensor would be transferred to the sensing element and measured. The sensing element must be fully enclosed inside the bracket. As described in the Fish Tail section, it is undesirable to have any part of the sensing element to come into contact with the corner of the bracket while the leak sensor is bent. It results in the additional normal pressure on the sensing element and noise to the measurement.

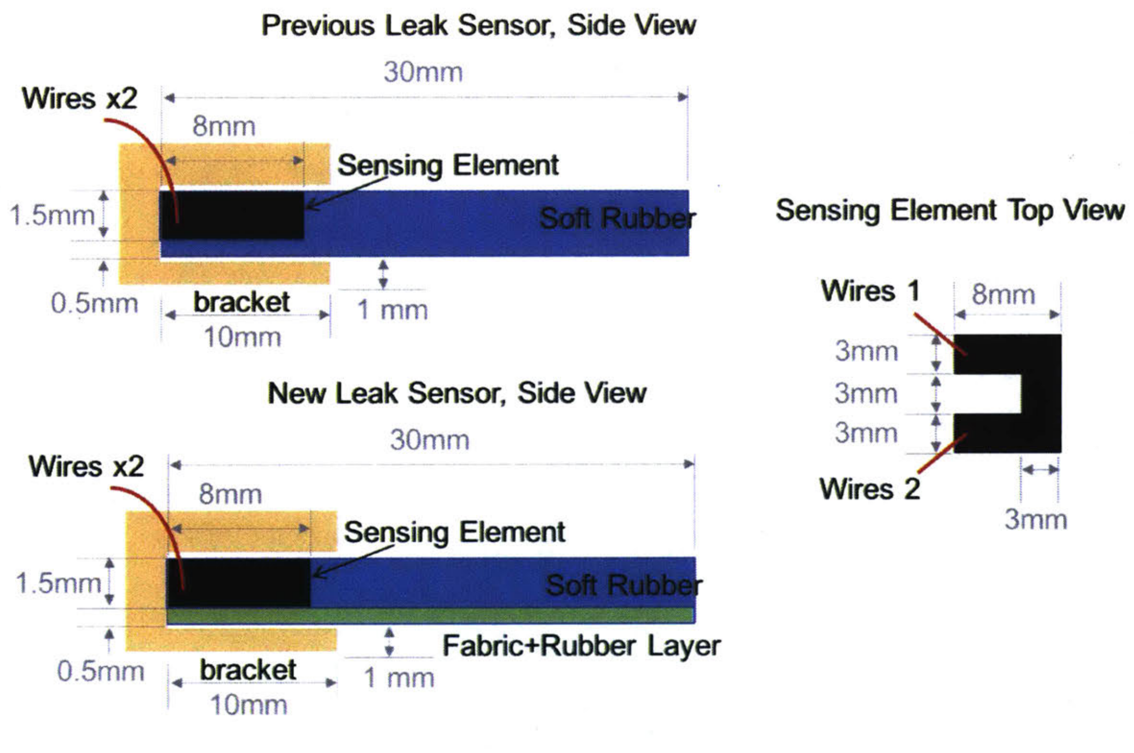


Figure 4-17: Illustration of the design of the new leak sensor in comparison to the leak sensor in last chapter

Several versions of the new leak sensors were compared in experiments to demonstrate the key design parameters and manufacturing requirements. The variables to test include the choice of fabric, the length of the sensing element in comparison to the length of the bracket, and the length of the fabrics. The bill of Material in Fig.4-18 shows the prototype leak sensors with those varying parameter. The outcome, including what works and what does not work, will be presented in the following sections. The first thing to highlight is the choice of fabric. There were many choices of fabrics; the only selection criteria on the fabrics was that they should be stiffer in the elongation direction than the soft rubber. It can be the thin woven cloth which cannot be stretched at all but can be bent easily. In Fig. 4-18, the nonstretch fabric with linear patterns is an example of the woven cloth. It can also be knit cloth which is softer, stretchable and equally easy to bend. In Fig. 4-18, the stretch fabric with mesh patterns is an example of the woven cloth. Those two types of fabrics can be made with the same threads, but the structure of threads inside these fabrics determines the overall stiffness of the fabrics (Fig.4-19). Fabrics of different stiffness affect the leak sensor's performance differently. This will be shown first in the experiment. Afterward, I will present two failure modes caused by inappropriate length of the sensing element and the length of the fabrics. They define the manufacturing requirements.

4.5.2 Experiments On Bending Angle Sensor Tuning

Experimental Setup

A set of experiments were designed to evaluate the new leak sensors' capability to tell leaks and obtrusions apart. The membrane leak sensors were placed inside the black rigid brackets as shown in Fig. 4-20. The test vehicle had two brackets to hold two membrane leak sensors. This design allowed the test vehicle to slide on flat surfaces in a stable manner, and ensures the quality of the measurement. It also improved the data collection rate; two sets of data could be collected in one experiment. In the experiment, the test vehicle was placed on top of a flat surface in a large water tank filled with water. On the flat surface there were artificial obtrusions and leaks. In the experiments, the water was not moving but the test vehicle was. The test vehicle was pushed at a speed about 100mm per second in a straight line and slide over the obtrusion. As illustrated in the sketch at Fig. 4-21, the obtrusions were long triangular prisms, with a cross-sectional profile of 4mm high and 5mm wide.

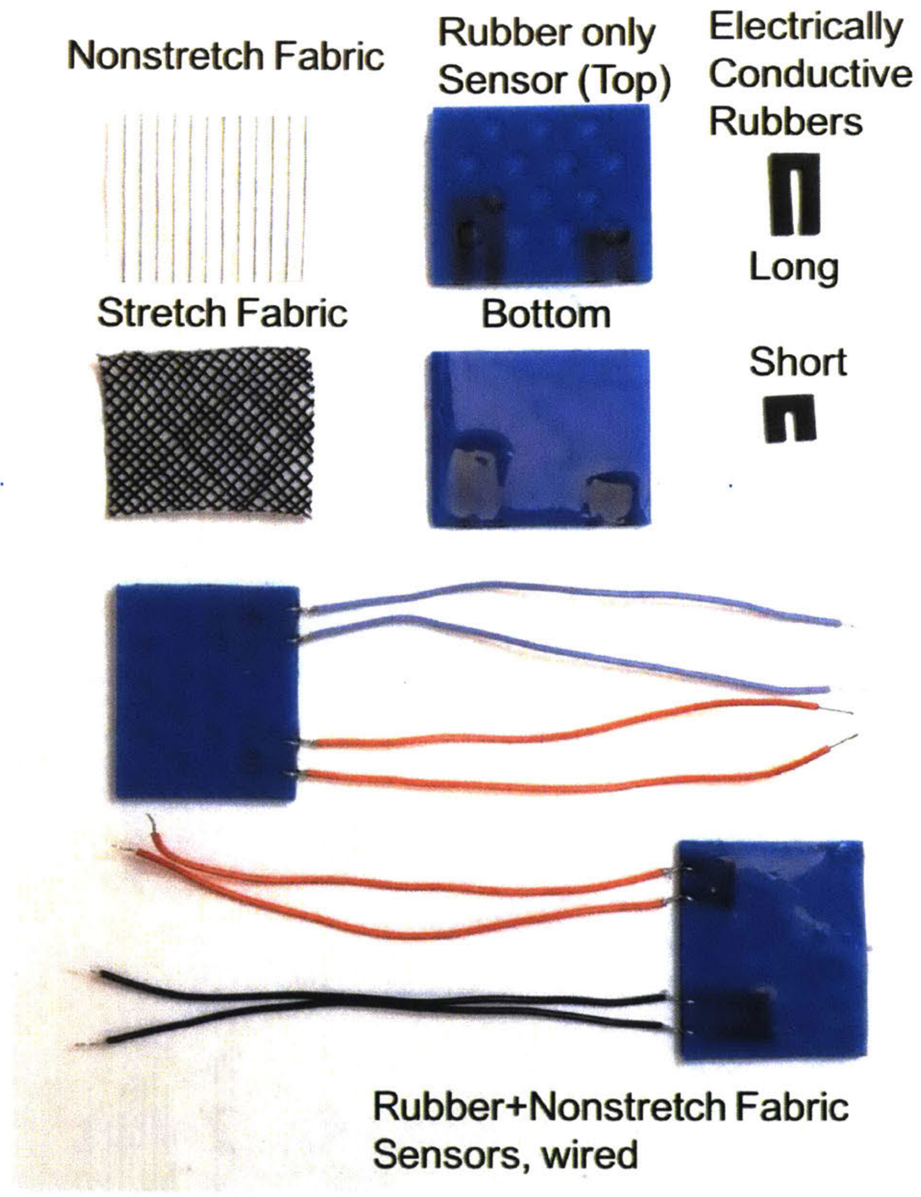

Nonstretch Fabric
Rubber only Sensor (Top)
Electrically Conductive Rubbers
Long
Stretch Fabric
Bottom
Short
Rubber+Nonstretch Fabric Sensors, wired

Figure 4-18: Bill of Material for the different kinds of new leak sensors

It was 45mm in length, and that was longer than the width of the membrane leak sensor(37mm). In other experiments, the test vehicle was pushed at the same speed of 100mm per second and slide over a leak. The leak was a 8mm long crack aligned with the direction of the vehicle's motion. It simulated longitudinal crack leaks in water pipes. The bottom side of the crack is connected to a 1/4 inch (6.35mm) inner diameter hose. The other end of the hose connects to a vacuum chamber that

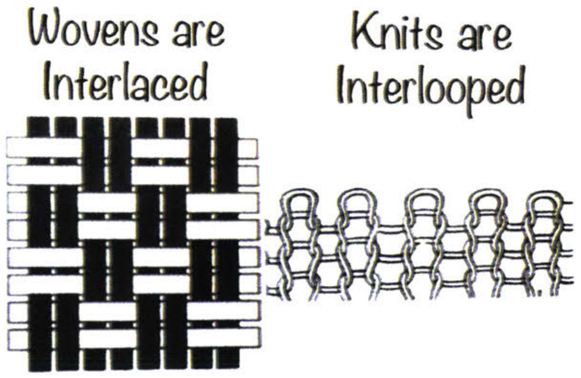


Figure 4-19: Two Basic Types of Fabrics [6]: Woven fabrics are difficult to stretch and knit fabrics are easy to stretch

was pressurized. When the leaks were left open, the pressure drop at the leak was estimated to be 11Psi (0.8 Bar) and the water was leaking at a rate of 0.8 gallon (3 Liters) per minute. The actual experiment setup is shown in Fig.4-22. The 4mm high obtrusions were in the back row, and the 8mm crack leak is in the middle of the central row where the hose is connected.

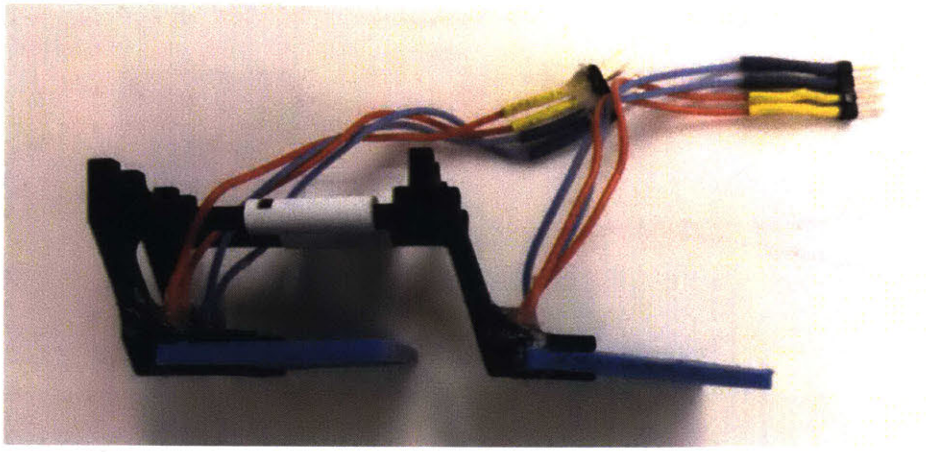


Figure 4-20: Prototype of the new leak sensor

Baseline: leaks and obtrusions are the same to rubber only sensor

The rubber-only membrane leak sensor developed in the last chapter was first experimented to establish a benchmark. It could not tell leaks apart from obtrusions. The rubber-only leak sensor, as illustrated in Fig. 4-17, was not designed as a proper bending angle sensor. It was expected to produce similar resistance change in response

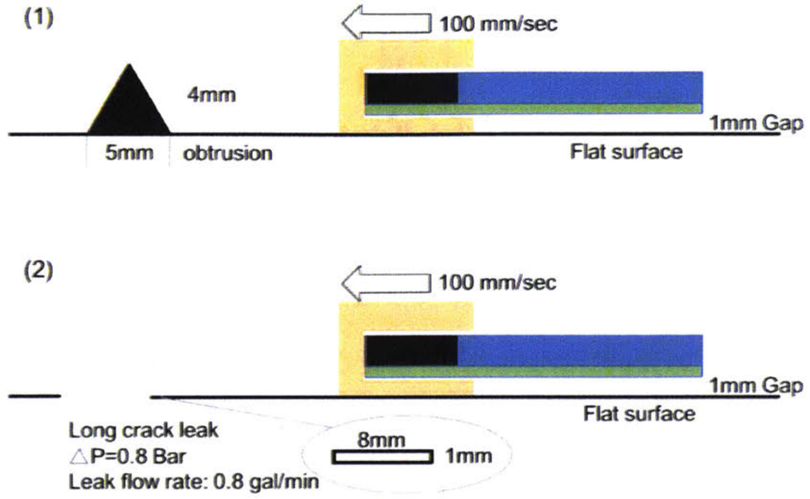


Figure 4-21: Illustration of the experiment setup for testing the new leak sensors' ability to tell leaks and obstacles apart

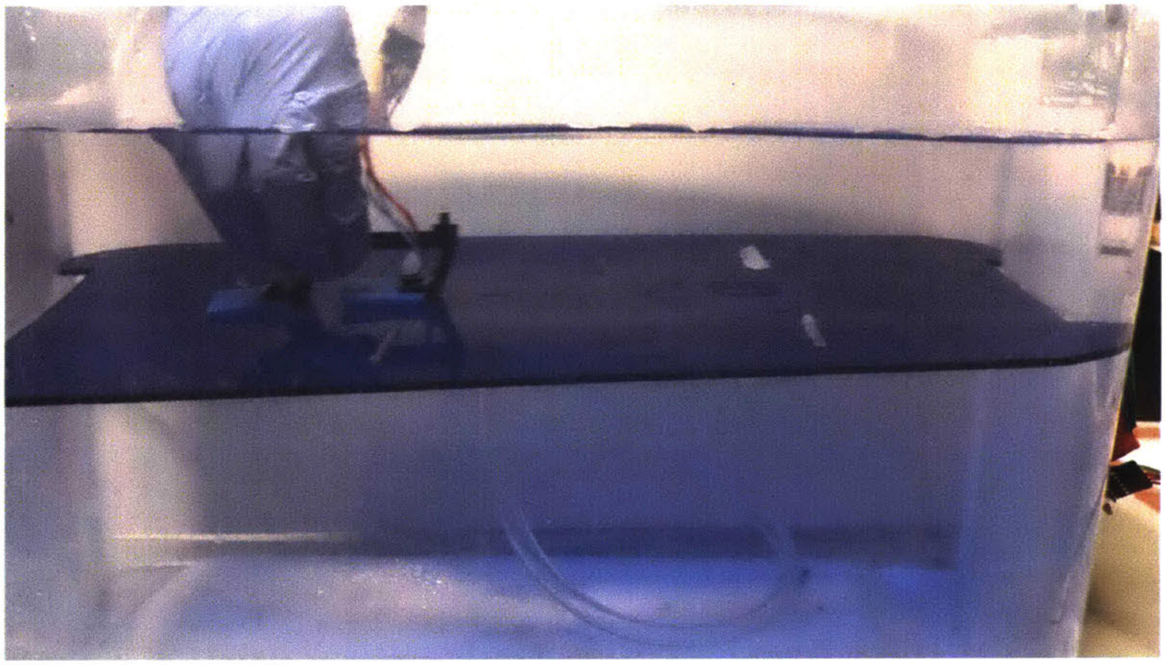


Figure 4-22: Actual experiment setup for testing the new leak sensors' ability to tell leaks and obstacles apart

to obtrusions and leaks. When the obtrusion bends the rubber-only leak sensor, the sensing element that crosses the neutral axis may experience elongation on the bottom side and compression on the top side. This mixed effect makes it difficult to observe the direction of motion from the resistance change alone. It depends on the different directional sensitivities of the sensing element to elongation and compression, and this particular sensing element is more sensitive to the elongation than compression. As the experimental results in Fig. 4-23 shows, in all three repeated tests on the obtrusion, the leak sensor outputs a positive peak for each obtrusion in its resistance measurement. At the same time, when the rubber-only leak sensor were experimented on the leaks, it also produced a positive resistance change for each leak as shown in Fig. 4-24. This agrees with the analysis that leaks pull and stretch the leak sensor, causing an increase in resistance. The response was weaker because the leak was small. When comparing the response to obtrusions in Fig. 4-23 to the response to leaks in Fig. 4-24, they were not significantly different. They are all increases in resistance. The obtrusions caused reactions of large magnitude, so could a bigger leak with a larger pressure drop. The rubber-only leak sensor is not an effective bending angle sensor and it cannot tell leaks apart from obtrusions.

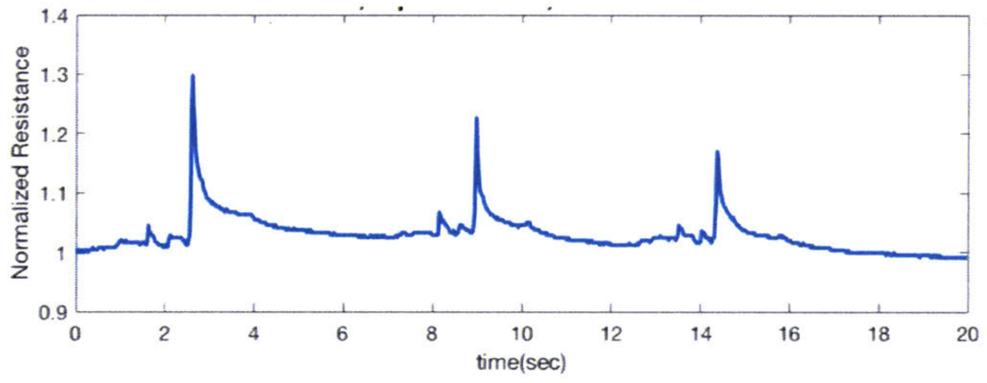


Figure 4-23: Lab experiment result, rubber-only sensor, measurement on three obtrusions (4mm high)

fabric-rubber sensor can measure leaks and obtrusions separately

In contrast, the new leak sensor, being an effective bending angle sensor, can clearly tell leaks apart from obtrusions. The new leak sensor has a stiff fabric layer in the bottom as shown in Fig. 4-17. Based on the neutral axis analysis, its sensing element should experience 100% compression when the leak sensor is bent upward by obtrusions. Thus it should output a decrease in resistance in response to obtrusions. In contrast, the sensing element should experience 100% elongation when the leak

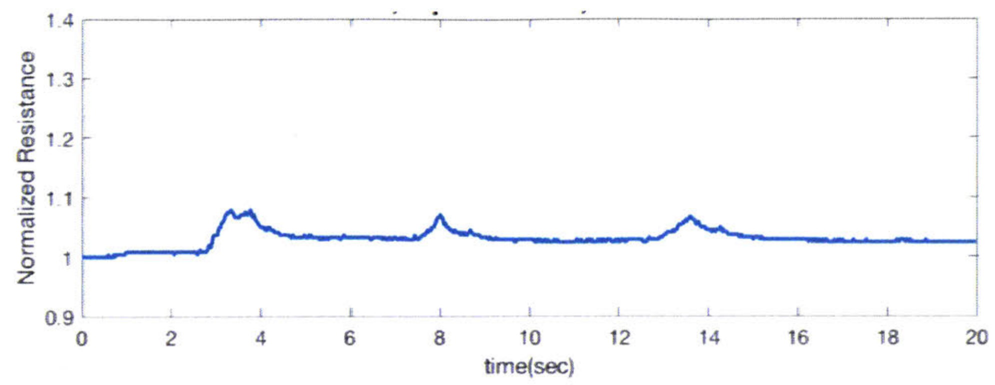


Figure 4-24: Lab experiment result, rubber-only sensor, measurement on three leaks (8mm cracks, 0.8 gal/min flow rate, 0.8 Bar pressure drop). The rubber-only sensor reacts similarly to leaks and obtrusions.

sensor is bent downward and pulled by the leak. Thus it should output an increase in resistance in response to leaks. This predicted difference was clearly visible in the experimental results. As shown in Fig.4-25, three obtrusions caused three drops in the resistance measurement. In comparison, three leaks caused three peaks in the resistance measurement as shown in Fig. 4-26. The difference in the obtrusion responses and the leak responses were distinctive; one positive and one negative. The soft bending angle sensor is an ideal implementation for differentiating leaks from obtrusions in pipes. Moreover, it can even be used to measure leaks and obtrusions at the same time.

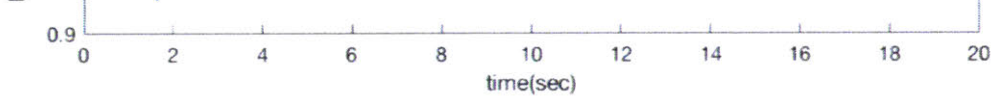


Figure 4-25: Lab experiment result, stretch fabric-rubber sensor, measurement on three obtrusions (4mm high)

One important note here is that stretch fabric is better than non-stretch fabrics

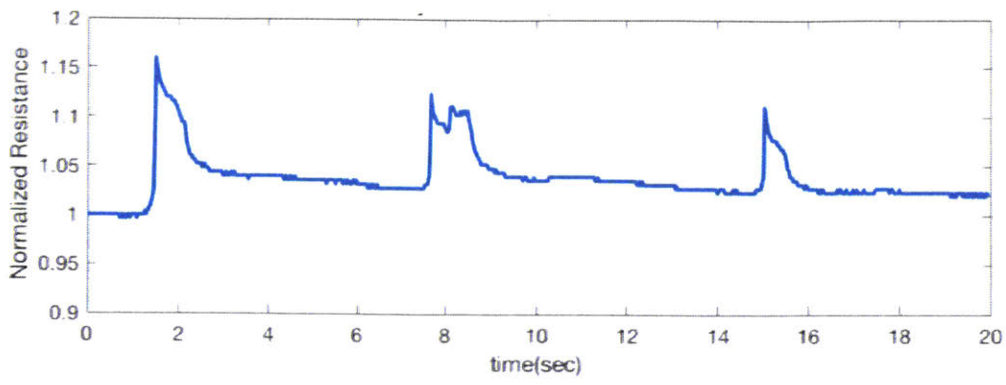


Figure 4-26: Lab experiment result, stretch fabric-rubber sensor, measurement on three leaks (8mm cracks, 0.8 gal/min flow rate, 0.8 Bar pressure drop)

for leak detection. The results above were produced by the leak sensor made with stretch, knit cloth but not non-stretch, woven cloth. Its response to leaks were much stronger than that with non-stretch fabric because it can also measure pulling force. Leaks affect the leak sensor in two ways: it bends the membrane down and also pulls on it. The pulling effect of a leak is much stronger than bending, since the maximum angular displacement is constrained by the the 1mm gap between the leak sensor and the leak surface as shown in Fig. 4-21. Moreover, given this constraint on bending angle, bending angle measurement alone does not reflect the size of the leak. The bending angle sensor made with non-stretch fabrics such as the woven cloth can measure bending angle fine, but it cannot measure pulling force. Given the high contrast in stiffness between the non-stretch fabric and the soft rubber, Almost all of the pulling force will be transferred through the non-stretch fabric to the rigid bracket. The sensing element will feel minimum input from the pulling effect of the leak. A leak sensor made with non-stretch fabric is not effectively measuring leaks; it is measuring the minor effect of bending while neglecting the major effect of pulling. This was observed in experimental result with the non-stretch fabric-rubber leak sensor, as shown in Fig. 4-27. In three repeated experiments on the same leak, the leak sensor made with non-stretch fabric produced almost no changes in resistance to the first two (at 4 second and 10 second), and very small resistance change at the third experiment. The maximum resistance change for this small, 0.8 gallon/minute leak was less than 5%. In comparison to this sensor made with non-stretch fabric, the one made with stretch fabric allowed the leak sensor to measure pulling force. It produced 2 to 3 times stronger response to the same leaks as shown in Fig. 4-26. With a lower contrast in stiffness between the fabric layer and the rubber layer, the sensing element shares a part of the pulling input with the fabric layer. Although the stiffness contrast is lower, the stretch fabric maintained the sensor's overall ability to sense bending direction. Since the fabric is placed in the bottom layer, both the pulling and downward bending input from the leaks elongate the sensing element and increase its resistance. The upward bending input from obtrusions compresses the sensing element and reduces its resistance. Leak sensors made with stretch fabric is well suited for both measuring leaks and obtrusions with high differentiability and high sensitivity.

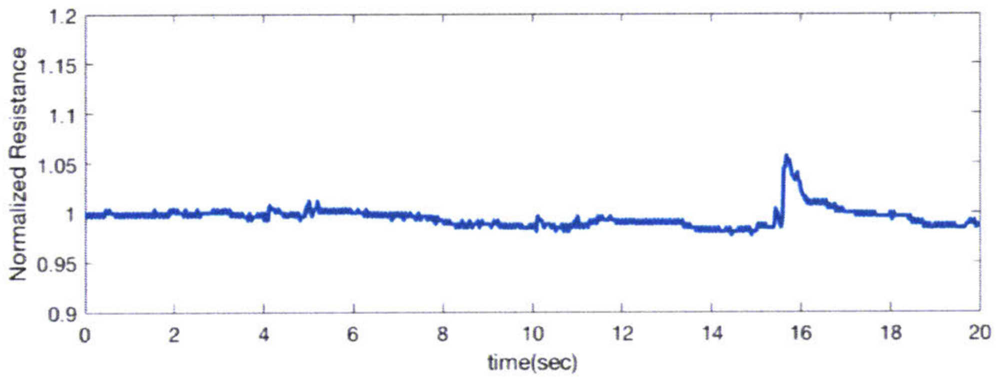


Figure 4-27: Lab experiment result, non-stretch fabric-rubber sensor, measurement on three leaks (8mm cracks, 0.8 gal/min flow rate, 0.8 Bar pressure drop). Non-stretch fabric-rubber sensor cannot detect the pulling effect from the leaks.

4.5.3 High Manufacturing Tolerance: Only Two Failure Modes

Manufacturing Process

All soft bending angle sensors presented in this chapter were fabricated in a simple manual process, and the devices worked well. In order to make a complete sensor as shown in Fig. 4-28-a, three components are prepared first.

Step 1: The U-shape sensing element are cut out from the commercially available

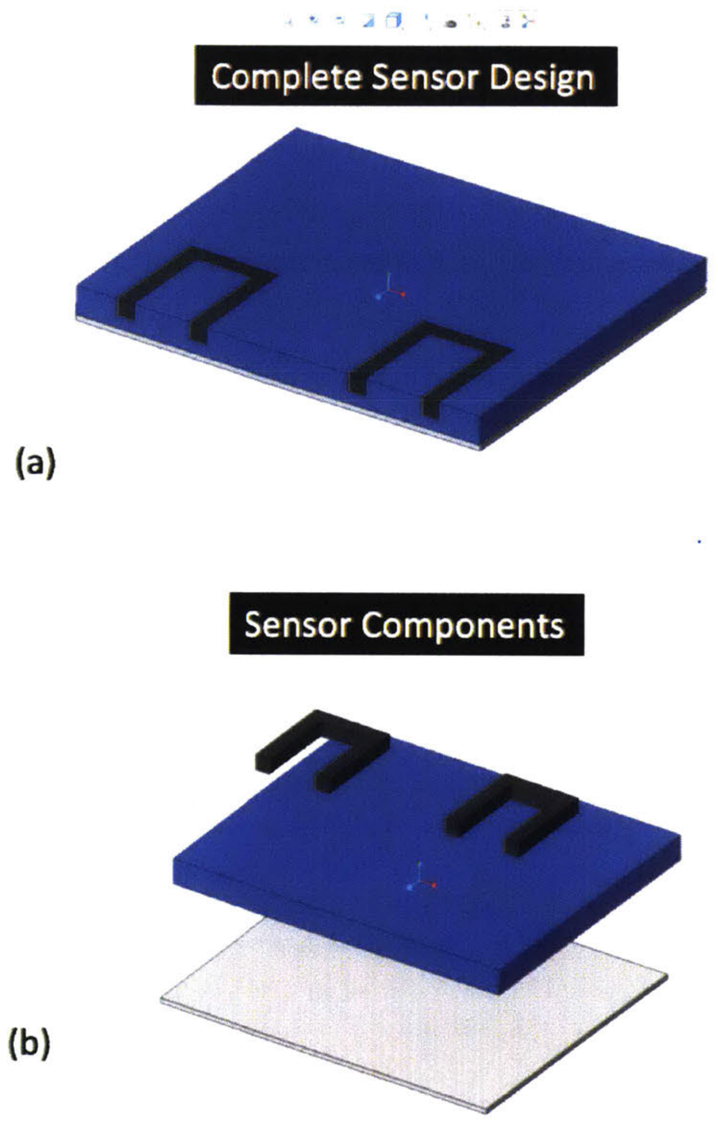


Figure 4-28: Fabrication Process of the Bending Angle Sensor

electrically conductive rubber.

They are 1.5mm thick, cut with ordinary scissors into the dimensions detailed in Fig. 4-17. The tolerance in its dimensions are $\pm$ 0.5mm; increasing the width or thickness of the sensing element will lead to an increase in its nominal electrical resistance, which is expected to be around 70 kΩ.

Step 2: Liquid rubber is prepared and poured into the mold as shown in Fig. 4-29. The blue non-conductive rubber are produced with Smooth-on Mold Star 30. It comes in two liquid-form compartments. After mixed in a 1 to 1 ratio and stirred

until reaching a uniform consistency, the liquid rubber is poured into the mold.
Step 3: The fabric is wet with the liquid rubber before pushing into the bottom of the mold.
The liquid rubber is highly viscous, and it is not absorbed by the fabric without external pressure. Thus the fabric is wet in the liquid rubber and massaged by hand to force absorption. Then the wet fabric is pushed into the liquid rubber until it lays flat on the bottom of the mold. The fabric cannot be placed in the mold before the liquid rubber is poured in; otherwise there will be little rubber and many air bubbles on the bottom side of the fabric. The fabric can be any kind of cloth. In this case, a regular knit cloth of 0.5mm thickness was used. It is about twice as stiff in the stretch direction as the the 2mm thick, 100% Mold Star 30 piece of thc same geometry.
Step 4: Once the fabric is in place, the sensing element is pressed into the liquid rubber at the location indicated in Fig. 4-28-a.

This process must be done within 20 minutes of the mixing of the liquid rubber, or the liquid rubber would become too viscous to work with. In practice, it took less than 2 minutes to wet each fabric and place the items into the liquid rubber. After leaving the mold still for 6 hours in room temperature, the liquid rubber solidifies completely, and this soft bending angle sensor is ready for wiring. In the wiring process, thin 30 AWG stranded wires are stitched with the help of a needle through the two ends of each U-shape sensing element. After the needle is removed, the wires are tied with a knot and fastened to the sensing element.

This soft angle sensor is already low-cost and even cheaper if manufactured in a industrial setting. With all off-shelf, commonly available material, each sensor costs 0.15 USD in material. In my manual process, it took about 15 minutes to do preparation and post-processing of four pieces of sensors, and 6 hours of waiting for the molding to be complete.

The time and cost to make each sensor can be further reduced when they are manufactured in bulk. The sensing elements can be stamped or machine cut instead of manual cut with scissor. The fabrics can be wet with liquid rubber with a paint

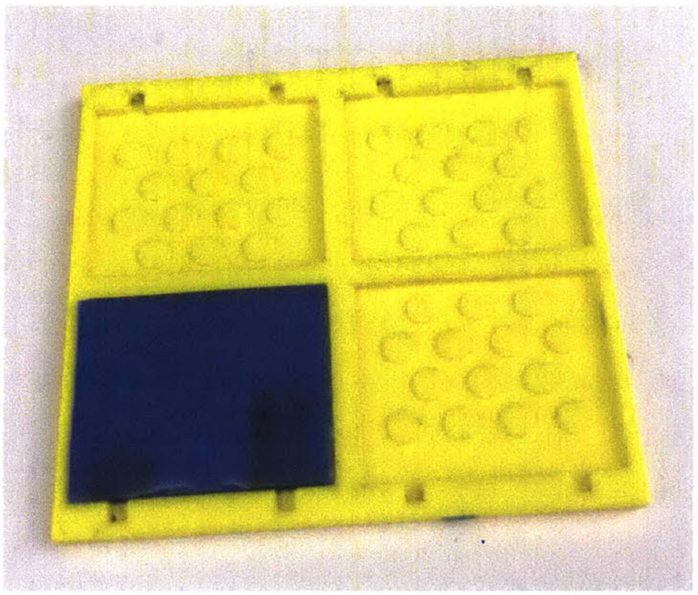


Figure 4-29: Mold for casting the soft bending angle sensor

roller rather than the manual wetting process. Instead of individual molds or four per batch as in Fig. 4-29, a large sheet that consists of 1000s of the sensors can be made at once, and then cut into the individual pieces. Instead of making a few sensors and waiting 6 hours for the liquid rubber to dry, one can make 1000s of the sensors with a bigger mold in the same 6 hours.

Item	Cost
1.5mm thickness U Shape Conductive Rubber (0.8 cm^2)	0.12 USD
Mold Star 30 Silicone Rubber (2 $gram$)	0.02 USD
0.5mm knit fabric (11 cm^2)	0.01 USD
Total	0.15 USD

High Manufacturing Tolerance: fabric layer is not necessarily flat

The manufacturing of this soft bending angle sensors has high tolerance to errors. For example, the liquid rubber can be mixed at 0.95 to 1 ratio and the outcome is still solid piece of sensor. Within this range of mixture ratio, no significant changes in the material or the effect on the sensor are observed. Neither of the sensing element or the soft rubber have to be of uniform thickness. The high contrast in stiffness between the soft rubber and the fabric ensures the soft bending angle sensor is effect regardless of millimeter thickness errors. The fabric does not even need to be flat in the sensor. In multiple prototypes I made, the fabric inside the soft bending angle sensor is curved as indicated in Fig. 4-30. There were air bubbles trapped underneath the fabric during the molding process and they produced enough buoyancy to push the fabric upward. However, the prototypes still worked. They were measuring bending directions and pulling forces equally well when compared the perfect sensors.

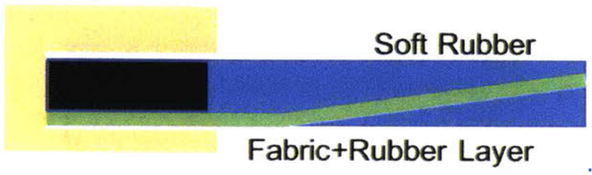


Figure 4-30: Manufacturing tolerance: the fabric(green) does not need to be flat

There are though two manufacturing errors that leads to failures. The first one is when the fabric layer falls short of covering the entire sensor. It disables the sensor's ability to differentiate bending directions. The other failure mode is when the sensing element is excessively long and come into contact with the edge of the brackets. This interaction adds significant noise to the sensor's measurement. Both failure modes are explained in detail below.

Failure Mode 1: Gaps in the fabric-rubber sensor disable bending angle sensing

When the fabric layer falls short of reaching the mounting end of the bending angle sensor, the sensor can no longer tell bending directions. As indicated in Fig. 4-31-1, when the fabric is cut short, there can be a a gap between the left end of the fabric and the mounting location in the bracket. Within this gap d_1, there is no fabrics, and the neutral axis is no longer along the interface between the fabric and the soft rubber but reset to the centerline of the device. Thus portion of the sensing element in this no-fabric zone can not tell the bending directions. Moreover, the part of the sensing element in this gap is more sensitive than the other part in the fabric zone. The device has much higher stretch stiffness in the fabric zone than that in the no-fabric zone. Since the less stiff region stretches more, given any kind of input, the response of the sensing element in the no-fabric zone will be dominant. Thus, to a bending angle sensor with this kind of defect, bending up and down will be indistinguishable.

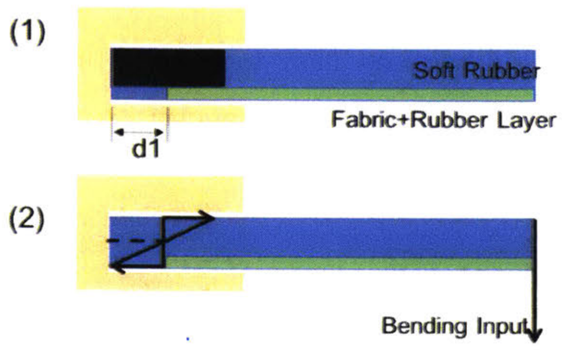


Figure 4-31: A gap in the fabric-rubber layer resets neutral axis

For a leak sensor with this defect, it cannot tell leaks apart from obtrusions as expected. Even when the gap size (d_1 in Fig. 4-31) is only 1mm, the sensor's failure to differentiate leaks from obtrusions was visible in experimental results. As shown in the leak measurements in Fig. 4-33 and the obtrusion measurements in Fig. 4-32. While the normal leak sensor was expected to be compressed by the upward bending input imposed by the obtrusion and output a resistance drop, the defected sensor output a resistance increase. This indicated that the part of the sensing element within the 1mm gap was elongated significantly by the obtrusion input. With d_1 = 1mm, the defected leak sensor behaved similarly to the rubber-only leak sensor as displayed in Fig. 4-23 and Fig. 4-24. Leaks and obtrusions all result in increases in resistance. Instead of moving the neutral axis and enable new functionalities, a short fabric only make the leak sensor stiffer without adding new functions.

Failure Mode 2: Excessively long sensing elements add noise

Similar to what we observed in the fish tail experiments, any unnecessary physical contact between the sensing element and the mounting tool causes undesired noise in the measurement. In the fish tail case, the mounting tool introduced a concentrated normal pressure on the sensing element and thus reduced its resistance. The leak

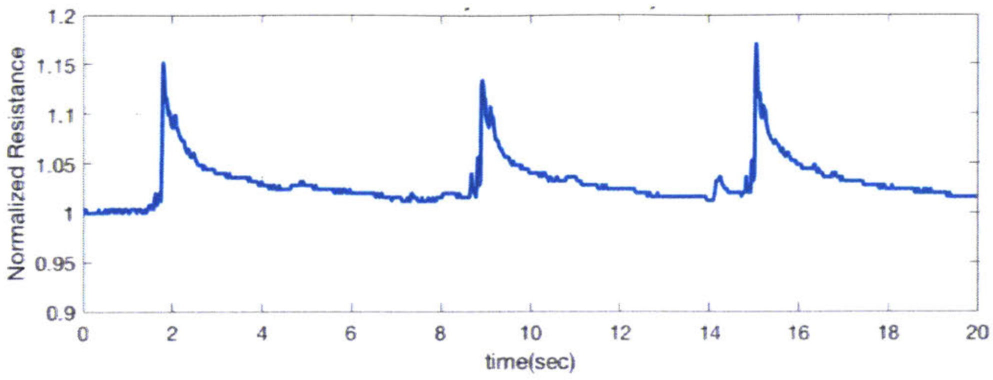


Figure 4-32: Lab Experiment result, fabric-rubber sensor, gap d1=1mm, measurement on three obtrusions (4mm high)

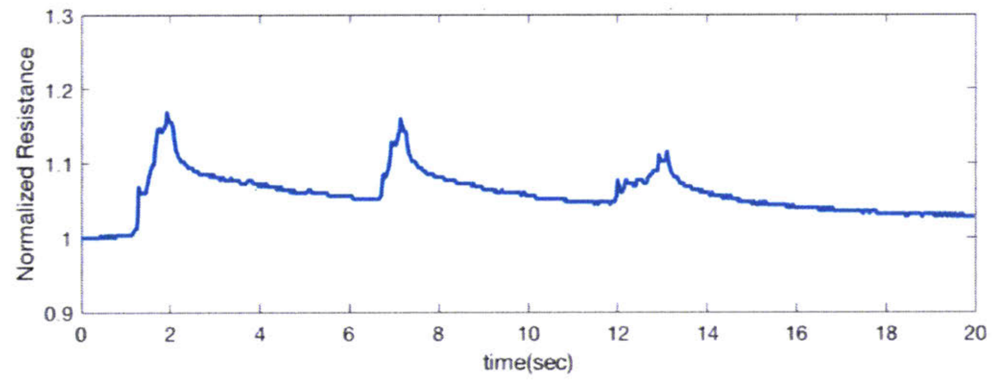


Figure 4-33: Lab Experiment result, fabric-rubber sensor, gap d1=1mm, measurement on three leaks (8mm cracks, 0.8 gal/min flow rate, 0.8 Bar pressure drop). The sensor can no longer differentiate leaks from obtrusions

sensor is designed with the intention to avoid this unnecessary single-point normal pressure. In its design as shown in Fig. 4-17, the sensing element is shorter than the bracket and fully enclosed in the bracket. In this way, the edge of the bracket will never press against the sensing element and affect its reading. A sensor with excessively long sensing element is considered a defect. An example of such is depicted in Fig. 4-34. When the sensing element extends a length of d2 beyond the yellow bracket, the edge of the bracket can press against the sensing element when the device is bent. This normal pressure would compress the sensing element; in the case of a conductive rubber, this compression reduces its electrical resistance. Therefore, when this defected leak sensor passes by a leak, there will be two competing effects on the sensing element: the elongation from the bending downward and pulling which increases the resistance, and the compression from the edge of the bracket(Fig. 4-34) that reduces resistance. This agrees with the observations in the experiments. In the experiments, a stretch fabric-rubber sensor, with a extra-long sensing element and d2=3mm, was tested on the same leak three times as all other leak sensors earlier. As shown in Fig.4-35, this sensor outputs a drop (compression) and then an increase (elongation) in resistance for all three times it passed by the leak. The compression from the edge of the bracket was dominating first and then outmatched by the elongation from the leak's downward bending and pulling effect. In comparison to the simple, clean, easy-to-interpret measurement from a normal leak sensor as shown in Fig. 4-26, the extra length of the sensing element add some unnecessary difficulty to interpreting the measurement.

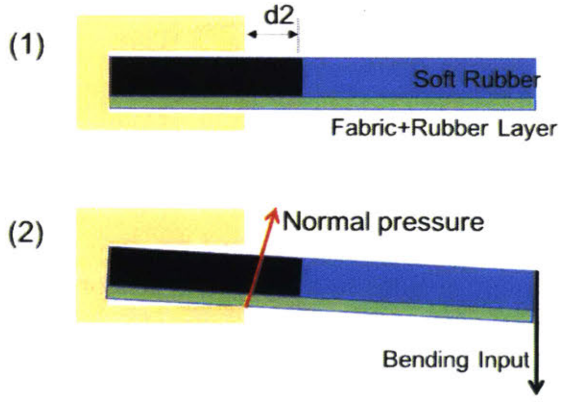


Figure 4-34: Excessively long sensing element interacts with the edge of the bracket and causes noise to the measurement

These two failure modes can easily be avoided. To avoid having a short fabric in the device, we can first make a longer device with longer fabrics and then trim it to the right length. During the trimming process, any excessive length of the sensing element can be reduced as well. In addition, there is enough tolerance built into the sensor design. The sensing element is intended to be 8mm long. If it is made 9mm long, it is still shorter than the 10mm long bracket and it would not touch the edge of the bracket. Those precautions are easy to implement, making the manufacturing process of this soft bending angle sensors even more tolerant to errors. It is a robust

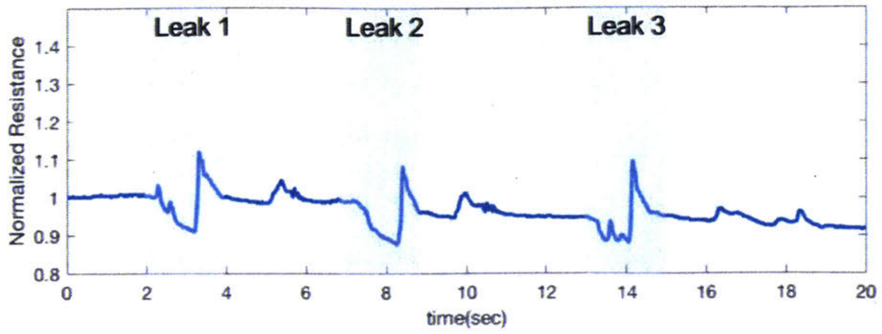


Figure 4-35: Lab Experiment result, stretch fabric-rubber sensor, long sensor d2=3mm, measurement on three leaks (8mm cracks, 0.8 gal/min flow rate, 0.8 Bar pressure drop). The leak sensor now sees a mixed effect of pulling and compression.

way to produce effective soft bending angle sensors.

4.5.4 Summary: A Low Cost Leak Sensor That Can Measure Both Leaks and Obtrusions, Separately

The soft bending angle sensor was successfully implemented in leak detection, in a low cost and robust way. It was capable of differentiating leaks and obtrusions. The principle of this sensor is about engineering the neutral axis of the device. This can be done by embedding in the soft device a piece of simple, ordinary fabric. Fabrics, being thin while extremely stiff in the elongation direction, allow us to use thick but commercially available conductive rubber to build previously impossibly thin soft bending angle sensors. This design allows us to build good performance sensors with low cost ordinary material. This design also has high tolerance to manufacturing errors. Thus the sensor can be produced without high precision machineries and even simply by hand. It is a low cost but effective solution to a complex problem.

4.6 Conclusion

A 1 USD soft bending angle sensor has been developed to measure leaks and obtrusions in pipes at the same time. It is a composite material sensor, made of low cost silicone rubber and ordinary fabrics. It is designed for low-cost manufacturing. It has minimum precision requirements even when they are produced on millimeter scale. Although being low cost, this sensor has great performance and many applications. When made into a leak sensor, it can differentiate leaks from other disturbances in the pipe such as obtrusions. It can even measure obtrusions at the same time. When made into motion sensors, it can track the tail swing of robotic fish. There are definitely more applications worth exploring with this soft bending angle sensor.

Chapter 5

A Practical Minimalism Approach to In-pipe Localization for A Soft Robot

5.1 Overview

In this chapter, I propose solutions to the localization challenge specific to this in-pipe soft passive robots. The challenges are two folded: being in-pipe and being passive. Being inside an underground water pipe, the robot has little to none connectivity with GPS or remote sensors. Locating the in-pipe robot with outside-the-pipe sensors are difficult, power intensive and costly. As a passive robot, it is propelled by the water flow and its speed changes with the water flow. This passive robot has no control of its speed; given the fluctuation and changes in the water flow, it is impossible to estimate the robot's trajectory based on vehicle dynamics and known actuation. Measurement of the robot's motion with on-board sensors is then key to the localization. Given the design of this soft robot and its leak sensors, it is possible to obtain speed measurement in an unconventional way. The robot, for most of the time it is in the water pipe, touches the pipe inner surface. This allows us to measure certain in-pipe features, such as pipe joints, to obtain speed reference. I propose two methods to utilize these measurements to estimate the location of the robot, and do so with a minimum number of on-board sensors only.

5.2 Sensors Are Not The Only Source of Information

In literature and in practice, in-pipe robots are commonly localized with one of two approaches: remote sensing and on-board sensing. The first one is remote sensing. The only other passive, flow-propelled in-pipe device on the market, Smartball by Pure Technology [20] is located with remote sensing. It is similar to GPS, just on a local scale. An extensive network of wireless relays are attached to the underground water pipe from the outside, and establish communication with the in-pipe device when it is within a certain range. Typically, the localization is done with the measurement of received signal strength from the relay to the in-pipe device, or from the device to the relay. In case there are regions where the wireless signal from the relays fail to reach the in-pipe device, the localization can still be performed with on-board sensors first, and then corrected via loop closure when the device is reconnected to the relays [43]. In practice, this remote sensing approach is expensive. Any GPS or wireless signals or are easily attenuated by the soil, the pipe material and the water before they reach the robot. If there are not enough density of the relays, there will be multiple no-signal zones where the robot can estimate its location with only low confidence5-1. To ensure the quality of localization results, the relay network are usually high in density and high in power consumption. The price to implement remote sensing effectively can be on the order of tens of thousand USD per mile. The overall goal of this thesis is to develop low cost solutions to solve the water leak problem. The robot so far costs less than 500 USD. It needs a localization solution much cheaper than the remote sensing network.

In comparison to the high cost wireless sensor network, on-board sensors are usually lower cost but also less effective solutions for in-pipe applications. On-board sensors can be used with the remote sensing to achieve greater accuracy, or used independently. Wheel encoders and inertia measurement units(IMUs) are used in many mobile robots, but they are commonly reported inaccurate [44]. We also attempted to use wheel encoders on the robot, and there was significant slip between the wheel and the pipe surface even when the pipe is empty. When the pipe was filled with

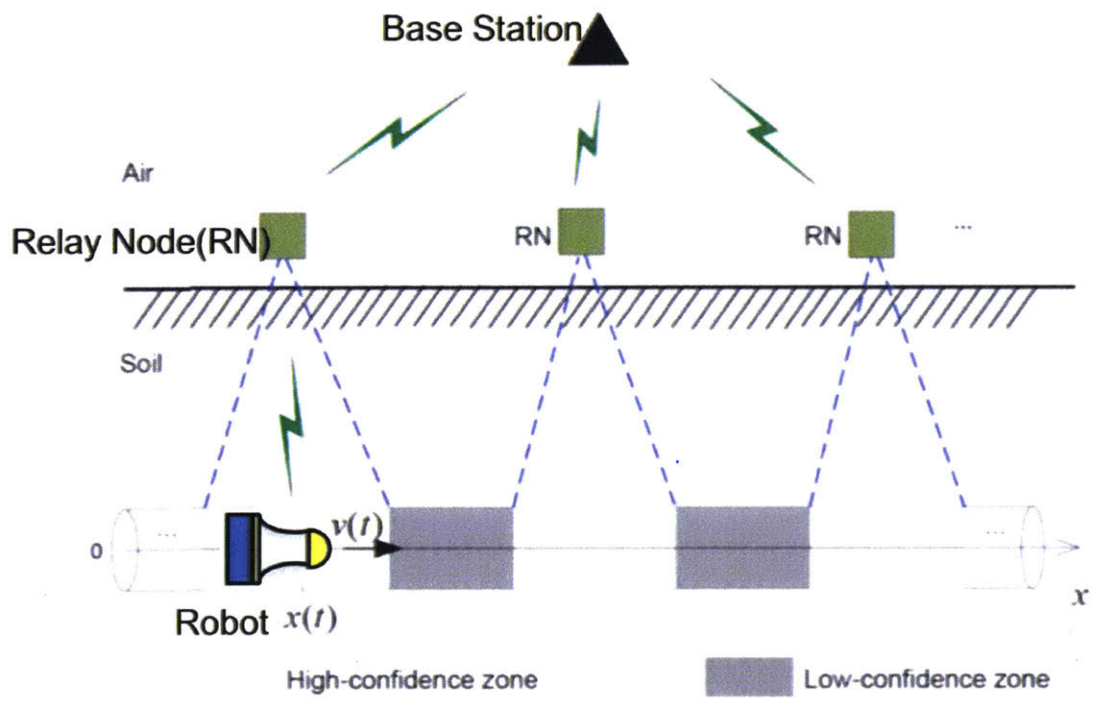


Figure 5-1: Typical remote sensing setup: the robot can estimate its location with high confidence when connectivity is available, and low confidence when the connectivity is not available.

water, the slip issue was worse. Wheel encoders are also difficult to install in soft robots; shafts, wheels and electronics are among the many components a wheel encoder requires. In comparison, IMUs are much easier to install in the soft robots. It is just one integrated chip. In the soft leak detection robots presented in previous chapters, there is already a Pololu MinIMU-9 v5 embedded. As the results in Chapter 3 indicate, IMUs are helpful in identifying if the robot is passing a pipe elbow, as well as heading directions. In many mobile robots and aerial drones, the encoders or IMUs are used together with other sensors such as cameras, acoustic and laser range finders. The low precision but high sampling rate data from encoders and IMUs are combined with the high precision but low sampling rate data from the other sensors through data fusion algorithms in order to produce better motion estimations. The available algorithms include Kalman Filter [45], SLAM [46] and others. When applied to in-pipe robots, cameras can be used for optical flow or visual odometry [47], or recognition of certain in-pipe features [48]. Cameras can lose its effectiveness due to low visibility when the pipe is filled with water and the water is moving. This is, however, the condition our leak detection robots operates in. Moreover, those additional sensors require space on the robot. It is possible to install them on the 4, 6-inch robots, but the space on the 2-inch robot is a huge constraint. Considering the cost, space and effectiveness, none of those additional sensors are chosen for this soft robot.

Sensors are sources of information but not the only one; all constraints imposed by the pipe system are also sources of valuable information. Water pipe systems impose many constraints on soft in-pipe robots. First there are the geometric constraints. the robot can only go forward inside each straight pipe section; since it matches the pipe in size, the pipe wall prohibits the robot from translating up and down or left and right. The design of this robot does not allow it to pitch or tilt in the pipe either. The only places the robot can turn are pipe elbows and T junctions. Then there are the dynamic constraints. In a typical water system, there are major fluctuations in the flow rate from one hour to the next through out the day. This fluctuation is slow and it follows regular patterns as an result of human activities. Many people are home using water in the evening while few people are at home using water during the day. However, within a short period of time, half an hour for example, the flow rate through any particular section of the pipe is more or less constant. Since the robot is passive and propelled by the water flow, knowing the flow speed is the same as knowing the robot's speed. Last but not the least, there are standards in pipes. All pipe systems are built with straight, fixed length pipe segments. In the US, the standard pipe segment length in the water systems are 6 meters (20 ft). This basically means every 20 ft there will be a pipe joint. If there is an accurate map of the pipe construction, the robot can count the joints and know exactly where it is. This is the case in the Saudi Arabia field test. If there is not an accurate map of the pipe construction or no map at all, the robot can still count the joints to estimate its average speed and location. This is the majority of the cases; water companies rarely keep a record of their pipe network that accurately shows where every pipe segment is.

Instead of adding more sensors to the robot, one can add information such as those geometric, dynamic and standard constraints from the pipe system to the robot's localization process. Given the design of this soft robot and its leak sensors, two kinds of on-board sensors are enough to estimate the robot's speed and heading direction inside a water pipe system: leak sensors and IMUs. The leak sensors are essentially tactile sensors. They measure features on the pipe inner surface. As we have discussed in Chapter 3 and Chapter 4, these tactile sensors can be used to measure obtrusions in pipes such as pipe joints. The tactile sensor enables the robot to count joints in pipes. Then the robot can estimate how far it traveled and its speed. At the same time, the IMU can tell precisely when the robot turned at a pipe elbow, which is the only place it can change heading direction, in addition to the rough measurements of the robots' acceleration, rotational speed and heading direction. Through data fusion, observations from the two sensors and the other available information about the pipe system are utilized together to create the best estimation of the robot's trajectory underground.

5.3 Field Test Observations

How Joint Presence Can Be Detected

Pipe joints compress the robot and this compression is measurable. As indicated in Fig. 5-2, at the location where two standard pipe segments are joint together, there is commonly an obtrusion pointing radially inward. This obtrusion can be an o-ring that seals the connection in PVC pipes, or the backside of a groove in some cast iron pipes. This obtrusion is commonly all around the circumference. As indicated in Fig. 5-2, in the case of a 52.5mm (2 inches) inner diameter cast iron pipe joint, the obtrusion is 1.6mm thick in radial direction. This drawing was for the same pipe and joints in the 2017 Saudi Arabia field tests described in Chapter 2 and 3. When the robot passes such a joint, this obtrusion will compress the robot on all sides.

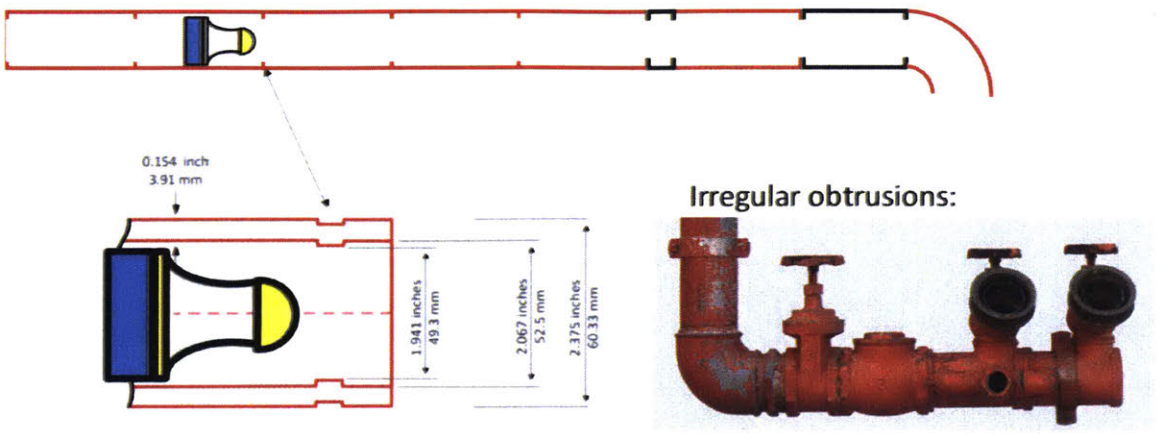


Figure 5-2: Illustration: Pipe joints, either between standard pipe segments or between other pipe elements, can compress the robot.

Two indicators of pipe joints were observed in the field tests. In one case, joints can disturb the leak sensors. obtrusions at joints can compress the robot on its leak membrane sensors and bend the sensors. Since the compression is from all sides, leak sensors all around the robot will output resistance changes. This simultaneous reaction on all leak sensors was observed in the field test result from Saudi Arabia, as shown in Fig. 5-3. It was a 2inch diameter cast iron pipe loop. There were 36 joints in the 221-meter long pipe loop, every one of them triggered readings on all four leak sensors. Therefore, one criteria to identify a joint is if there were simultaneous changes in all leak sensors on the same circumference.

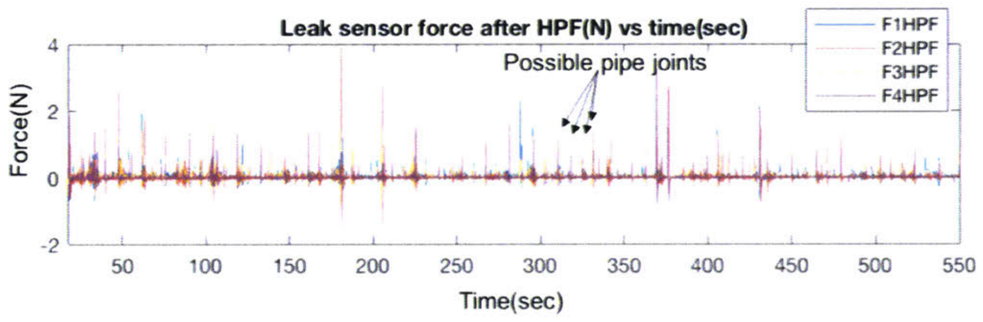


Figure 5-3: Field test results from 2017 Saudi Arabia test: every peak in the leak sensor output history may indicate the detection of a pipe joint.

In the case of larger robots, a joint can momentarily disturb the robot's radial stability. While the robot is in the pipe, its umbrella shaped back-end is fully expanded and maintains its contact with the pipe surface on all sides. This contact is keeping the robot centered in the pipe and aligned with the pipe's centerline. When the robot goes through a pipe joint, this umbrella part of the robot is squeezed and now smaller than the nominal pipe diameter. For a brief moment after the robot squeezed through the pipe joint, the robot is still smaller than the pipe diameter. Part or all of its contact with the pipe surface may be lost temporarily until the robot is expanded again. In this process, the robot may pitch or yaw slightly due to disturbance in the water flow or unevenly distributed contact and friction with the pipe surface. Then the angular position of the robot resets as the robot expands and re-establishes contact with the pipe surface. This angular deviation and reset is small in magnitude but high in frequency. It can be measured by the gyroscope inside the IMU on the robot. This angular deviation was observed in the 2018 Virginia field test. As shown in Fig. 5-4, in one segment of the field test, the robot recorded a sequence of pulses in the measurement for its angular velocity in the radial direction, each representing the arrival at a joint. Meanwhile, the magnetic sensor or compass in the IMU reported that the heading direction did not change much in the process. This indicated that after the momentary angular deviation, the robot returned to its normal orientation. One may note that this radial instability was not observed in the 2017 Saudi Arabia test results (Fig. 5-3). The reason is the difference in the robot size and design. In the Saudi test, the robot was 2 inch in diameter and had little compressibility. It was always in contact with the pipe surface and thus always stable in the radial direction. In this Virginia test, the robot used was 6 inches in diameter and can be compressed down to 4 inches. It can easily be compressed and destabilized in the radial direction. As the bigger robots aimed for more compressibility and adaptability, the price in this design trade-off is radial stability.

Pipe mapping can be done based on the identification of the joints. A pipeline is made of connected, straight, standard pipe segments. In each country there are standard length for the pipe segments. In both field tests in Saudi Arabia and Virgina, the pipe segments are 6 meters (20 feet) in length. By counting the joints, the robot estimated how many 6 meter long segments it passed. For example, as shown in Fig.

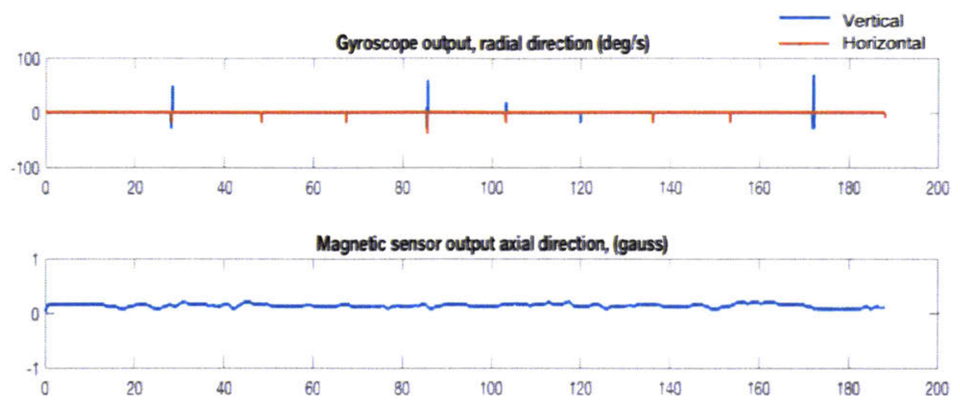


Figure 5-4: Field test results from 2018 Virginia test: (1) every peak in angular velocity may indicate the detection of a pipe joint; (2) the magnetic sensor monitors the robot's heading direction change. In this case the robot's heading direction did not change much when the pipe joints disturbed the robot's radial stability.

5-5, a portion of the pipeline in the field test could be mapped by the robot. The white line indicates the trajectory of the robot which is the same as the pipe. The white dots on the line represent pipe joints. With the ability to identify joints, this robot can create pipe maps with details such as location of pipe joints. While most of the existing pipe geographic information system (GIS) provide coordinates of pipes, the map this robot created adds the locations of pipe joints. This kind of resolution of pipe GIS has not been demonstrated by any commercial products before. On top of that, with good speed measurement, the position of all features in each pipe segment can also be estimated.

Challenges In Joints Identification

Observations from the field tests also indicated three potential challenges. From the two field tests, it is observed that joints can be identified from measurements of either the leak sensor, or the gyroscope. However, either those measurements are designated for joint identification, nor can they detect joints with 100 % accuracy. The three challenges are summarized in Table. 5.1

The first challenge is false positives. Joints are obtrusions to the robot, but not all obtrusions are joints. Other features in the pipes, such as ball valves, gate valves and sometimes tuberculations, can cause similar readings on the robot's measurements

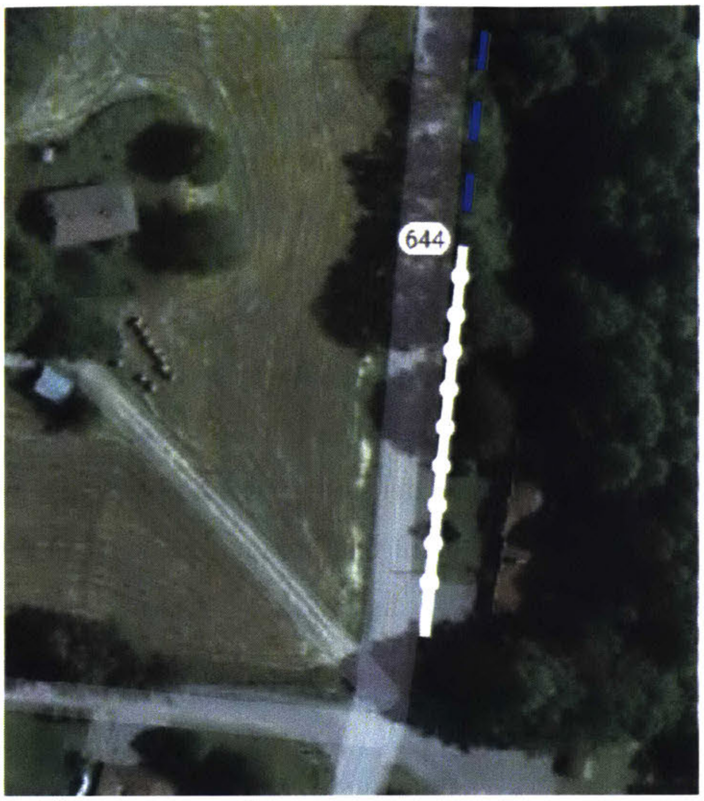


Figure 5-5: Field test results from 2018 Virginia test: plot pipe maps from the joint measurements

as pipe joints do. These are the false positives. Those false positives are common; as shown in the Saudi field test result in Fig. 5-3, there were many minor spikes in the leak sensor measurement. Many of those false positives were caused by the rust buildup in the water pipe. The Saudi field test was conducted in a 6 year old cast iron pipe that was mildly rusted. Just like we were trying to differentiate leaks from obtrusions in Chapter 4, now we need to differentiate different types of obtrusions. Many of the false positives can be screened with a threshold on the signal

Table 5.1: Summary of Practical Challenges in Joint Identification

Challenges	Examples	Impact
False Positives	Valves and tuberculations that compresses the robot similarly as joints	Overestimate Distance
False Negatives	Worn out joints that the robot fails to notice	Underestimate Distance
Other Pipe Elements	Tee junctions, pipe elbows, valves, shorter pipe segments that add lengths	Overestimate Distance

magnitude and a threshold on the correlation among all leak sensors. The obtrusion at a joint is all around the circumference so it compresses the leak sensors on all sides. In comparison, a rust buildup are rarely even on all sides of the pipe; they are more likely to be on one side of the pipe and compress one or two leak sensors. Through calculating the correlation of the outputs of all leak sensors, we can filter out those responses due to single-side rust buildups. Example of this correlation method can be seen in Appendix B. However, there may still be false positives due to rust buildups that are all around the pipe circumference, as well as all the valves. Counting obtrusions is easy for the robot, but identifying which obtrusions are joints can be difficult. With false positives in the joint measurements, the robot may overestimate the number of joints it has passed and the distance it has traveled.

The second challenge is false negatives. The robot may pass some pipe joints without recognizing them. These are the false negatives. Pipes are manually connected, and joints are assembled at different quality. In some cases, the obtrusion at a joint may be very shallow. This could be an inappropriately placed o-ring in a PVC pipe joint. The obtrusion can also be worn out. In those cases, the robot will not measure anything at this joint. The leaks sensors may not pick up any signals, and the gyroscope may not notice any angular displacements. With false negatives in the joint measurements, the robot may underestimate the number of joints it has passed and the distance it has traveled.

The third challenge is the length of other pipe elements. Tee junctions, pipe elbows and valves are common elements in a water pipe network. They all have their own dimensions which are much shorter than the standard pipe segments. There could also be shorter pipe segments. Those shorter ones are cut off from the standard ones. They are used in place where the space may not permit a full 6 meter pipe segment. All those non-6-meter elements are connected in series to the standard 6-meter-long pipe segments. This means more joints. This also means the distance between joints are not always 6 meters. If the robot assumes the distance between each adjacent joints are 6 meters constant, it may overestimate the distance it has traveled. The good thing is that in any water pipe system, most part of them are the standard pipe segments. Shorter pipe elements are the minority.

Challenges In General In-Pipe Localization

Other than the challenge in identifying joints, there are three more practical constraints specific to the localization problem in municipal water pipes. From the field tests and interviews with water authorities, I find that pipe map is commonly not accurate, and neither is the pipe flow speed in real time. The flow rate along the water pipe is not consistent but it diminishes due to active water usage. Given those constraints, we cannot make certain assumptions such as the availability of accurate pipe maps or access to real time flow speed knowledge. Instead, the robot must be able to estimate its speed and location on its own.

Existing pipe maps are not accurate. During the field tests and in my interviews with municipal water authorities, this is commonly the case. Pipes were installed many decades ago, and the construction map was handcrafted without exact dimensions. As the cities are rebuilt over time, the pipe maps are usually not updated. The old map may make reference to a building or a street that no longer exist. Connections added to the water pipe network may not be recorded. Moreover, locations of pipe joints were never on the map. It is impossible to know the absolute location of the joints without digging up the pipes. Frequently, the location of pipe valves were not even marked at all. Sometime even pipe elbow locations are not marked. The water authorities know the pipe is roughly along this street, but there can be a couple 90 degree bends on it that they couldn't find in the record. The robot cannot rely on the existing pipe map.

Pipe flow speed is generally not available. During the field tests and in my interviews with municipal water authorities, no-one could give the exact flow rate through every pipe sections. When asked for flow rate, the water authorities can at most provide guesses. Rarely they could find a nearby district flow meter and read the flow rate from it. If the robot needs to know its speed, it has to measure it on its own.

The inlet flow rate may be constant but the flow rate through out the water pipe is not. There is large, slow, periodic variations in the pipe flow rate throughout the day, and it is caused by human activity patterns. The duration of the robotic pipe inspection is short when comparing to the period of water usage fluctuation. Thus the flow rate at the inlet during the inspection may be constant, with minor fluctuation. The flow rate through out the tested water pipeline may be not consistent due to water usage. The flow rate decreases after every active service line connection. The service line connections are where water is extracted from the main pipeline into households, restaurants and factories. With some amount of water leaving the pipe at each service connection, the flow rate forward is reduced by the same amount. This reduction is a step function; it is not a gradual or linear decrease of flow rate over a distance.

5.4 Proposed Methods and Key Assumptions

In this chapter, I present two methods to overcome those challenges to perform robust in-pipe robot localization. Both methods are designed specifically for soft in-pipe robots. They are low cost as they require a minimum number of sensors on-board and zero remote sensors. Their performance are sufficient at the minimum requirements, but can surely be improved further with more sensors. Both methods integrate the motion estimation provided by an IMU and the information gathered from identified joints. The first method gathers relative distance and average speed from counting joints. The IMU data and the relative distance data compensates each other. The IMU provides good estimation of high frequency changes in the motion, such as when the robot start moving after being stationary for a while. This is the case in launching the robot. It is also the case when the flow is stagnant for a while and starts moving again. At the same time, the relative distance measurement tracks the low frequency, steady-stage shift in the robot's motion. The second method includes further information such as the instantaneous speed at each obtrusion. Both methods are robust when faced with the three general challenges of in-pipe localization. None of them requires prior knowledge of the pipe map or flow speed. When implemented on a passive, flow-driven robot, those methods enable the tracking of the flow speed change throughout the pipeline. This information is previously extremely difficult to capture without expensive and complex sensing systems.

Both methods rely on two key assumptions about a typical water pipe system. The first assumption is that flow rate is varying slowly during the short period of robotic inspection. This is usually the case in practice. Given the mass of the water in the pipe, the flow fluctuation is small in a short period of time. However, the flow rate varies in space, as every active household water usage reduces the flow rate in the main water pipe. The proposed methods, without relying on external measurement or prior knowledge, will be able to capture the flow speed variation in time and space.

The second assumption is that majority of the water pipeline are made with consecutive standard pipe segments. There are both standard pipe segments and other non-uniform length pipe elements in any water system, but it is reasonable to claim that more than 90% of the length consists of consecutive, standard, 6-meter long straight pipe segments. In contrast, the distribution of the other pipe elements, such as valves and pipe elbows, are sparse and random. It is unlikely that there are multiple valves evenly distributed with exactly the same distance between them. One can imagine the pipeline as music. The standard pipe segments are the repeating notes with a rhythm. The other pipe elements are occasional, and they do not have a rhythm. Within those consecutive standard pipe segments, the robot can count the joints to estimate its positions and speed with high confidence. In practice, two problems stand out. The first problem is how to recognize consecutive standard pipe segments from all other obtrusions in the pipe. The second problem is how the robot estimate its position in the rest 10% pipes where there may be no consecutive standard pipe segments. Both problems are addressed in my proposed methods.

5.5 Simulation Study Setup

The proposed localization algorithms were intensively studied in simulations. There were limited field test opportunities for us to try the robots with the algorithms in real pipe conditions. To refine the algorithms and test their robustness, simulation studies were performed in Matlab. In the simulations, the pipelines and their flow conditions were first generated to be similar to what was observed in the 2017 Saudi Arabia field test and 2018 Virginia field test. Then their parameters were adjusted to further test the robustness of the localization algorithms.

The simulator consists of three modules: a generator, a sensor and an estimator. First, the pipe map and robot's motion is simulated in a generator program. In the generator, a virtual robot flows through a randomly generated pipeline, and the output is the robot's true motion data. The robot is passively propelled by the water flow. The water flow enters the pipeline at a constant average rate with some random fluctuations. The flow rate reduces throughout the pipeline due to active water consumptions.

Second, the measurement data is simulated in a sensor program. The sensor simulates the robot's onboard sensors such as the IMU. The measurements include linear acceleration, angular rotational speed, and compass direction reading. The robot also has the tactile sensor that can measure both out-flux of water at leaks or service connections, as well as any obtrusions in the pipe. Those measurements are super-positioned with noise. The water flow rate is unknown to the robot. The robot can not measure location or speed directly.

Third, an estimator program is tasked to give the best guess of where the robot is at every point of time. The robot has to estimate its speed and location from only the IMU readings and tactile sensor outputs. The proposed algorithms are implemented in the estimator.

Generator

The simulated pipe map is generated with many features. The top view of an example pipe map is shown in Fig. 5-7. There are standard 6-meter-long pipe straight segments, pipe joints, 90 degree elbows. There are several minor Tee junctions on the pipeline, and they represent service connections to homes and businesses. They can point to the left of the pipe, or to the right. The mean density of these service connections are 6 per 100 meter of pipes. This is in-line with the reported national

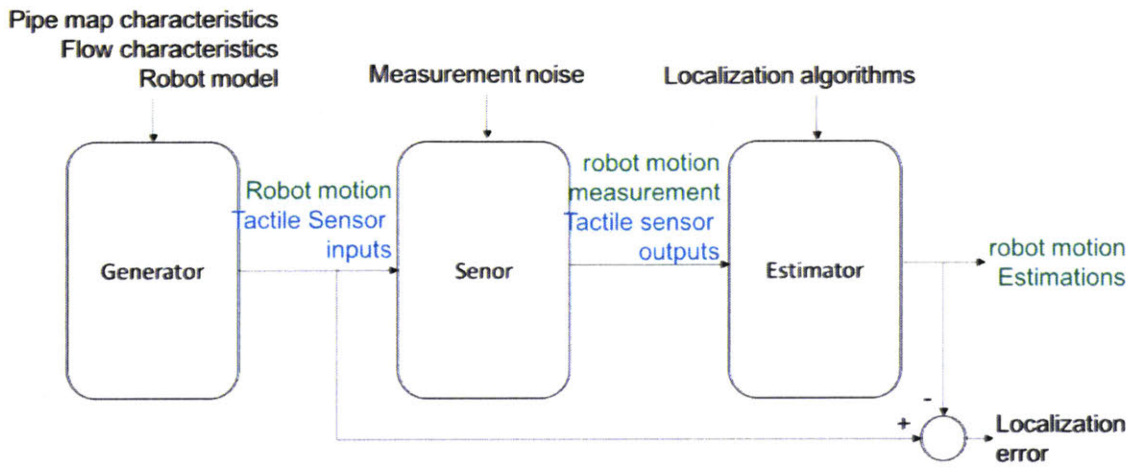


Figure 5-6: Three modules in the robot localization simulator

average of 191 people or about 63 families per kilometer of pipes [16]. There are also a few valves placed in the pipeline. There are also other obtrusions on the pipe representing tuberculations.

For simplicity, the following assumptions are applied to the simulated maps:

1. All pipes are in horizontal plane
2. 90 % Pipe segments are straight and of standard length L_{seg}
3. Pipe segments are aligned (0 degree offset)
4. Pipe only change heading at 90 degree bends
5. All pipe segments are of the same diameter d
6. All Tee junctions are service connections; they are too small in diameter for the robot to enter
7. There is one inlet and one outlet for this pipeline, no branches.

This scope of this simulation study is thus limited to 2D pipelines. Change in the depth of the pipes, or a network of pipes are out of the scope of this study, but the simulation algorithms can be augmented for those conditions.

The robot's motion is confined within the simulated pipeline, and a few more assumptions are applied to it. The robot has no actuation, and it is not self-propelled. Its motion is driven by the fluid drag force $F_d(V_f(t,s), V_r(t))$.

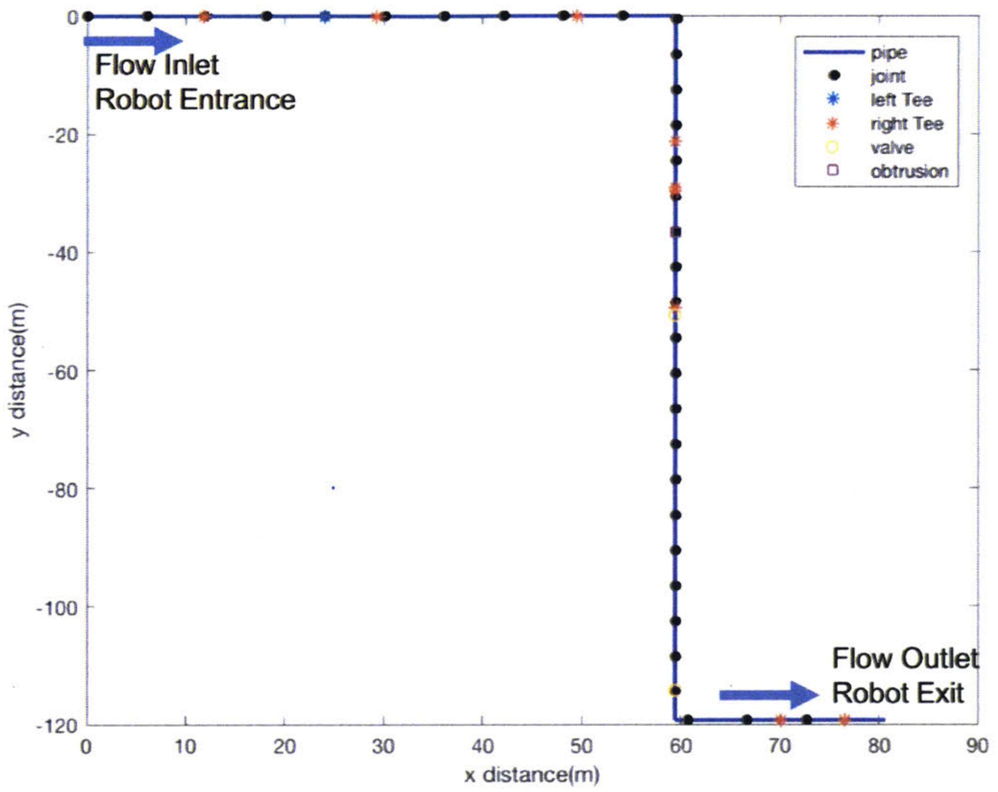


Figure 5-7: Example Simulated Pipe Map

$$F_d(V_f(t,s), V_r(t)) = \frac{c_d \rho A}{2}(V_f(t,s) - \dot{s})^2 \tag{5.1}$$

Here $V_f(t,s)$ is the flow speed at time t and distance s from the inlet. The flow speed is dependent on distance because it reduces after every active household connection. $V_r(t) = \dot{s}(t)$ is the robot's in-line speed at time t. Since the robot's dimension is almost the same as the water pipe diameter, the robot is like a piston in the cylinder with very high drag coefficient c_d. c_d can be determined through computational fluid dynamic(CFD) simulation. ρ is the density of water and A is the back-side area of the robot in the direction of the pipe flow.

At the same time, there are a small amount of friction on the robot as it slides along the pipe surface. This friction is denoted as

$$F_f(g(s)) = c_f * F_N(g(s)) \tag{5.2}$$

$F_f(g(s))$ is dependent on the location of the robot s, friction coefficient c_f, and normal force F_N. F_N is dependent on if there is a change of pipe feature, $g(s)$,at location s. The value of $g(s)$ is binary; it is zero in normal pipe segments, and one at ball valves, gate valves, pipe joints, Tee junctions, pipe elbows as well as any tuberculations. Those features have their own lengths. At those features, the normal force between the robot and the pipe $F_N(g(s))$ will be higher, and thus friction $F_f(g(s))$ will be higher.

For simplicity, four assumptions were made in the simulated flow in the pipe:
1. The flow speed at the inlet has a mean value of V_{f0} and Gaussian white noise of standard deviation σ_{f0}
2. The mass of the water in the pipe acts like a 1st order low pass filter with a cut-off frequency at f_l
3. Active household connection reduces by the same amount ΔV_f
4. Any active household connection is assumed to remain on for the entire duration of the robotic pipe inspection.

For simplicity, six assumptions were made in the simulated robot's motion:
1. When the robot first enters the pipeline through the insertion point at $s = 0$, it is stationary: $V_r(s = 0, t = 0) = 0$
2. The robot is never stuck or moves backward $V_r(t > 0) > 0$
3. The robot, given its size is almost the same as the pipe diameter, is constrained to move only in the axial direction of the pipe but it cannot translate in the radial direction of the pipe
4. Friction coefficient c_f is treated as constant throughout the pipe
5. Within normal pipe segments, normal force $F_N(g(s) = 0) = c_1(s)mg$ where mg is the weight of the robot, $c_1(s)$ has a mean value of c_1 with Gaussian white noise of

standard deviation σ_{c1}

6. At other pipe features, normal force $F_N(g(s) = 1) = c_2 mg$, and c_2 is a lot higher than c_1, $c_2(s)$ has a mean value of c_2 with Gaussian white noise of standard deviation σ_{c2}

7. Friction is much smaller than the drag force, so for most of the time $V_f(t, s) \approx V_r(t)$).

Then the robot's motion through out the pipeline can be generated with the following dynamic equation

$$(m + m_a)\ddot{s} = F_d(V_f(t, s), V_r(t)) - F_f(g(s)) \tag{5.3}$$

where m is the mass of the robot and m_a represents the added mass [40] of this robot. In this setup, the robot's motion is confined to within the pipeline and it cannot move sideways. The input variables are the flow speed $V_f(t, s)$ and pipe features $g(s)$. The output variables are the in-line distance s, in-line speed $\dot{s}$ (same as V_r) and in-line acceleration $\ddot{s}$ of the robot. Substitute in the representations of the drag and friction forces, this detailed dynamic equation governs the robot motion.

$$(m + m_a)\ddot{s} = \frac{c_d \rho A}{2}(V_f(t, s) - \dot{s})^2 - c_f F_N(g(s)) \tag{5.4}$$

Analysis in the rest of this chapter will be shown using the example pipe shown in Fig. 5-7 and the parameters summarized in Table 5.2. Some of the parameters are determined by the design of the actual robot prototype. The other parameters are manually chosen to represent certain set of flow and friction conditions.

This simulation generator outputs the true, discretized motion trajectory of the robot, and it captures several key features in the real water system. An example motion is shown in Fig. 5-9. In this example, the discretized motion trajectory is sampled at 5,000 Hz. The robot's velocity has an overall decreasing trend. It reflects that water is being drawn by households, and the pipe flow diminishes toward the end of the pipeline. The in-line acceleration is high at the beginning, as the stationary robot is catching up to speed with the moving water flow. For the remaining part of

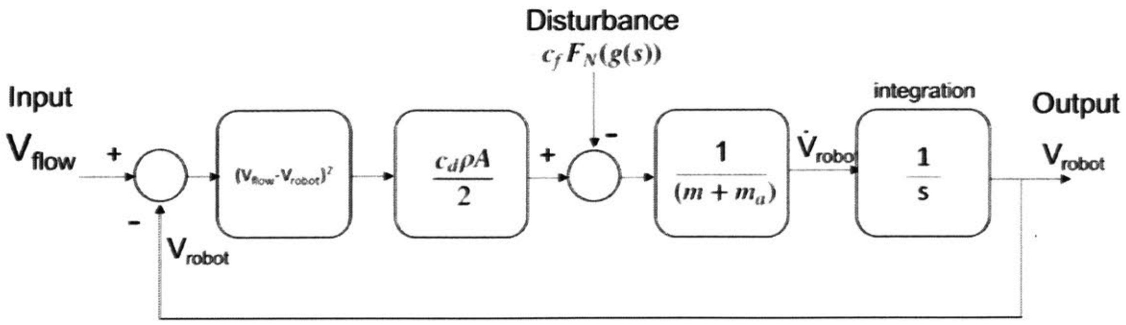


Figure 5-8: Block Diagram of the robot system dynamics

Table 5.2: Parameters used to generate an example simulated in-pipe robot mission

Parameter	value	unit
L_{seg}	6	meters
d	150	millimeters
V_{f0}	0.5	m/s
σ_{f0}	0.5	m/s
f_l	10	Hz
ΔV_f	3%	
c_d	2	
rho	1000	kg/m^3
A	0.0177	m^2
m	2.35	kg
m_a	0.88	kg
g	9.81	m/s^2
c_f	0.01	
c_1	0.2	
σ_{c1}	0.02	
c_2	2	
σ_{c2}	0.4	

the trajectory, the acceleration has a mean value around zero and minor deviations due to fluctuation in the flow speed, and friction changes at every in pipe features.

The Matlab code for the generator can be found in Appendix C.

Sensor

The sensor simulates the sensors on-board the robot, generating noisy measurements about the robot's interaction with the simulated environment. It includes simulated accelerometer, gyroscope and tactile sensors. None of these sensors provide direct

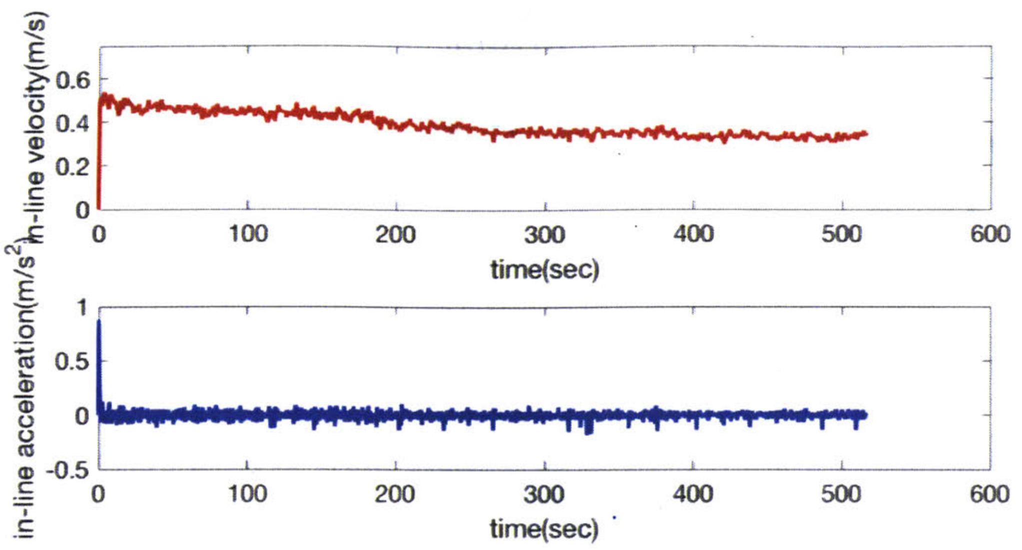


Figure 5-9: true motion trajectory of the robot in simulation. The red trace is velocity and the blue trace is acceleration.

measurement of the robot's speed. The sensor samples at $f = 50$Hz; it is a low but reasonable sampling rate for implementation in robots powered by regular arduino based micro-controllers. The accelerometer is assumed to be of a single axis and it is aligned with the axis of the pipe. It captures the robot's in-line acceleration as in Fig. 5-9, but it is also affected by an addictive Gaussian white noise of zero mean and standard deviation σ_{acc}. The gyroscope is also assumed to be of a single axis and positioned to capture the robot's yaw rate in the horizontal plane. The gyroscope is affected by an addictive Gaussian white noise of standard deviation σ_{gyro}, in addition to a bias term governed by b_{gyro}. The parameter of b_{gyro} defines the expected drift in the gyroscope reading per minute. Both the noise and drifts in the accelerometer and the gyroscope is modeled following Kalibr [49, 50], a conventional off-line IMU calibration approach. In this model, the noise in the measurements is added using the following equations:

$$\ddot{s}_{output}(k) = \ddot{s}(k) + \sigma_{acc}n(k) \tag{5.5}$$

$$w_{output}(k) = w(k) + \sigma_{gyro}n(k) + bias(k) \tag{5.6}$$

Table 5.3: Noise Parameters in the simulated accelerometer and gyroscope

Parameter	value	units
σ_{acc}	0.5	m/s^2
σ_{gyro}	2	deg/s
b_{gyro}	1	$deg/s/min$

where $\ddot{s}$ and w are the true acceleration and true rotational speed of the robot. They are also outputs of the generator. $\ddot{s}_{output}$ and w_{output} are the measured acceleration and rotational speed. The measurement noise are characterized by a unit strength white Gaussian noise n multiplied by the standard deviation of the noise, σ_{acc} and σ_{gyro}, respectively. The gyroscope measurement has a bias component that increases overtime. It is defined as

$$bias(k) = bias(k-1) + b_{gyro}\sqrt{\frac{1}{f}}n(k) \tag{5.7}$$

The parameters of the accelerometers and the Gyros are chosen based on the specifications of the IMU used inside the prototype robot, which is a miniIMU V5 by Pololu. A sample output from this part of the sensor is shown in Fig. 5-10.

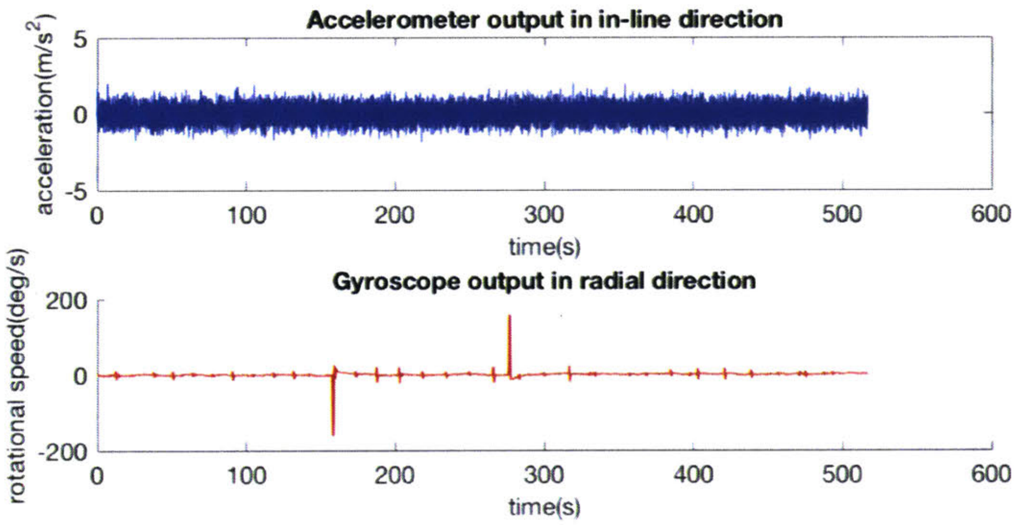


Figure 5-10: Sample motion data output from the simulated in-pipe robot. The acceleration measurement is very noisy. The gyroscope measurement contains large peaks that corresponding to pipe bends, and small peaks that corresponding to pipe joints.

Table 5.4: Parameters in the simulated tactile sensors

Feature	expected nominal magnitude
joints	5
household connections	20
pipe bends	20
obtrusions	5

The other part of the sensor simulates the tactile sensor outputs. An example output is shown in Fig. 5-11. The tactile sensor can measure both out-flux of water at leaks or service connections, as well as any obtrusions in the pipe. Fig. 5-11 shows the output of a simulated first version of the tactile leak sensor, which cannot differentiate leaks from obtrusions. There are four channels of the tactile sensors, each monitoring the pipe surface condition in their respective quarter of the pipe circumference. Leaks, service connections are assumed to be contained within one quarter of the pipe circumference, so only one tactile sensor will be triggered. Pipe features such as pipe joints, valves and bends, will trigger similar reading on all four channels. The other obtrusions such as tuberculations can trigger one or multiple tactile sensors. The probability they trigger any number of tactile sensors is a uniform distribution. In the simulation, the tactile sensor output, denoted as tactile factor, has a nominal magnitude and length for every kind of in-pipe feature. Those parameters are summarized in the table below. There are also variations in the magnitude of the tactile signal for each feature, and this variation is assumed to be 20% of the nominal magnitude. Although in reality pipe features usually have different dimensions, in this simulation all features are assumed to be uniformly 0.04 meters long for simplicity.

The Matlab Code and other information pertaining to the sensor can be found in Appendix C.

Estimator

The estimator performs data fusion on multiple sources of information to best approximate the robot's true motion. The goal is to estimate the robot's position at every point of time. The available sources of information include:

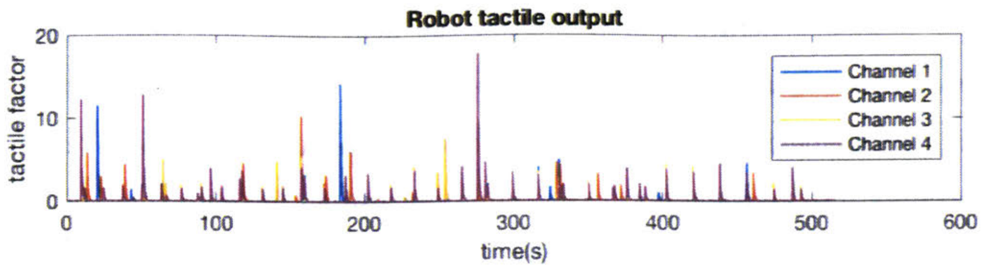


Figure 5-11: Sample tactile data output from the simulated in-pipe robot. Each peak in the data corresponds to either a pipe joint, a household connection, a pipe bend or an obtrusion.

Table 5.5: Four sources of information and constraints for the in-pipe robot localization problem

1. In-line acceleration, yaw rate, and tactile measurements of pipe features from the sensor
2. Understanding of the dynamics of the robot in a water pipe, as in Equation. (5.3)
3. Assumption that the in-pipe flow rate is varying slowly during the short period of robotic inspection
4. Assumption that majority of the water pipeline are made with consecutive, same-length, standard pipe segments

It is clear that the robot's speed information is missing, while the speed is critical for calculating position. In addition, the flow speed is also missing, while the flow speed is important for determining the robot's speed, according to Equation. (5.3). Thus this estimator is tasked to estimate the robot's speed, the flow speed at every point of time. From the speeds, we can then derive the in-pipe robot's trajectory and thus the pipe map.

The baseline estimator algorithm is an Extended Kalman Filter(EKF). EKF allows the data fusion between the noisy measurement (Source 1 in Table 5.5) and knowledge of the robot-pipe dynamic interactions (Source 2 and 3 in Table 5.5). Given the in-pipe robot's dynamics as in Equation. (5.3) is nonlinear, an Extended Kalman Filter is selected rather than regular Kalman Filter. In the framework of EKF, the dynamics of the robot and the assumption that the in pipe flow is changing slowly become the model. A model is the robot's understanding and belief of how things work. Once the robot has a belief of its current state, it can propagate its belief through the model, generate a prediction of the next state and thus a prediction of the next measurement. The model can be treated as a form of constraint, and the actual measurement, although noisy, is another form of constraint. By taking both constraints into consideration, the robot can use an EKF to find the optimal estimation of the next state. In this case, an optimal estimation is defined as the approximation with minimal expected error variance and at the same time it satisfies all constraints.

The robot's motion throughout the pipeline is tracked with four states. They are the in-line distance robot traveled s, the in-line velocity of the robot $\dot{s}$, the in-line linear acceleration $\ddot{s}$, and the flow velocity in the pipe V_{flow}. The goal is to know $s(k)$ for all time step k.

$$X(k) = \begin{bmatrix} s(k) \\ \dot{s}(k) \\ \ddot{s}(k) \\ V_{flow}(k) \end{bmatrix} \tag{5.8}$$

$$Y(k) = HX(k) = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} X(k) \tag{5.9}$$

What is measurable is the acceleration $\ddot{s}$. The robot's velocity $\dot{s}(k)$ and the flow velocity $V_{flow(k)}$ are both unknown. Together they determine the drag force on the robot. With the assumption that the friction force on the robot can be treated as Gaussian white noise, the dynamics equation (5.4)can be rewritten as

$$\ddot{s}(k) = \frac{c_d \rho A(m + m_a)}{2}(V_{flow}(k) - \dot{s}(k))^2 + W_f(k) \tag{5.10}$$

where $W_f(k)$ is the effect of the friction force on the robot, modeled as a Gaussian white noise of zero mean and variance Q_{aa}.

The estimator based on Extended Kalman Filter is then setup to estimate all four states. At every time step k, the inputs are the aposteriori estimate $\hat{X}(k-1|k-1)$, the belief about the state at the prior time step k-1, and the aposteriori error covariance estimate $P(k-1|k-1)$, the believed error in the prior state's estimate.

Step 1: update the apriori estimate of the current state $\hat{X}(k|k-1)$, with the system dynamics and the inputs. The system dynamics equation here is the linearized version of Equation (5.10)

$$\hat{X}(k|k-1) = \begin{bmatrix} \hat{s}(k|k-1) \\ \hat{\dot{s}}(k|k-1) \\ \hat{\ddot{s}}(k|k-1) \\ \hat{V_{flow}}(k|k-1) \end{bmatrix} \tag{5.11}$$

$$\hat{X}(k|k-1) = A_{est}(k-1)\hat{X}(k-1|k-1) \tag{5.12}$$

With the linearized model which is update at every step

$$A_{est}(k-1) = \begin{bmatrix} 1 & \Delta t & (\Delta t)^2 & 0 \\ 0 & 1 & \Delta t & 0 \\ 0 & -\frac{c_d \rho A(\hat{V_{flow}}(k-1|k-1)-\hat{\dot{s}}(k-1|k-1))}{(m+m_a)} & 0 & \frac{c_d \rho A(V_{flow}(\hat{k-1}|k-1)-\hat{\dot{s}}(k-1|k-1))}{(m+m_a)} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.13}$$

Step 2: update apriori error covariance estimate at time step k by propagating the aposteriori error covariance estimate $P(k-1|k-1)$ through the system model.

$$P(k|k-1) = A_{est}(k-1)P(k-1|k-1)A_{est}(k-1)^T + Q^T \tag{5.14}$$

where Q is the error covariance for the process noise. It includes information such as the flow fluctuation and friction. Q is assumed to be static and time-invariant. The choice of initial value of P, X and Q will be explained in the example in the next section.

Step 3: the Kalman gain for time step k is determined from the apriori error covariance estimate $P(k|k-1)$ and the measurement error covariance R.

$$K(k) = P(k|k-1)H^T/(HP(k|k-1)H^T + R) \tag{5.15}$$

Step 4: update aposteriori state estimate $\hat{X}(k|k)$, which is the weighted average of the apriori state estimate and the measurement.

$$\hat{X}(k|k) = \hat{X}(k|k-1) + K(k)(Y(k) - H\hat{X}(k|k-1)) \tag{5.16}$$

Step 5: Update aposteriori error covariance estimate $P(k|k)$ to reflect that the expected error in the state estimate increased.

$$P(k|k) = (I - K(k)H)P(k|k-1) \tag{5.17}$$

where I is the identity matrix of rank 4.

The Matlab code for the estimator can be found in Appendix C.

5.6 Benchmark: Simple Dead Reckoning with EKF

Integration of Acceleration

Using EKF as the estimator is better than simple integrating acceleration data. EKF is a data fusion technique, and it uses the system dynamics and the measurement as constraints to generate state estimations. If the robot does not used EKF but simply estimates its speed through integration on its acceleration measurement, it is ignoring the dynamic from the in-pipe water flow and generating unbounded error in its speed estimation. As shown in Fig. 5-12, the speed estimation from simply integrating accelerometer data diverges from the actual velocity of the robot, and its error grows over time. A method to generate bounded speed estimation is necessary, and EKF is the right algorithm for this task.

The Matlab code for this result can be found in Appendix C.

EKF and Assumed Average Speed

A method to generate bounded speed estimation is necessary, and EKF is the right algorithm for this task. EKF allows the robot to constraint its motion estimation with both the IMU measurement and the assumed system dynamics model. However,

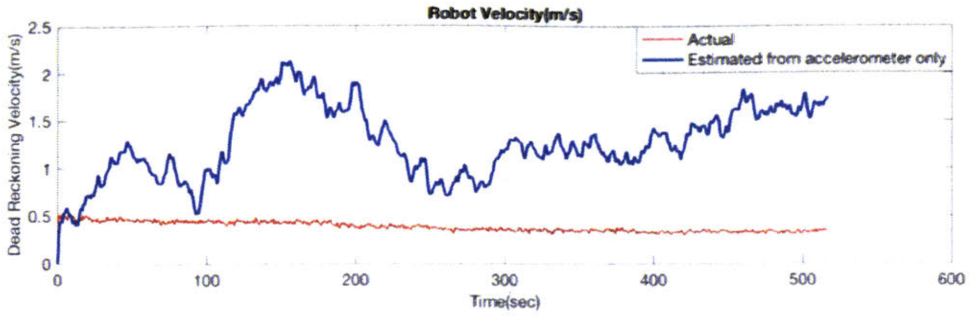


Figure 5-12: Challenge with estimating velocity with accelerometer data: error build up over time and diverge from real speed

there are two difficulties in implementing EKF on this robot: the initial flow speed is unknown and the flow speed variation throughout the pipeline is unknown.

The first difficulty is to know the initial flow speed. At k=0, the robot is placed inside the water pipe at the starting point and it is momentarily stationary. Thus in the initial condition, $s(0) = 0, \dot{s}(0) = 0$. However, $\ddot{s}(0)$ and $V_{flow}(0)$ are unknown. The flow speed in the water main is commonly not monitored in practice. The water authority generally can only give an estimation of the flow rate and it is usually not accurate. There is only one feasible way left to estimate the initial flow speed. The water authority can give an estimation of the length of the pipeline, and the robot can measure the time it takes to go from one end of the pipeline to the other. Dividing the total distance by total time, the robot can have an estimate of its average speed. Since this is a passive, flow driven robot, its average speed is thus very close to the average flow speed in the pipe. The robot can assume the flow speed is constant and take this average flow speed as the initial flow speed. Now three out of four states in the initial condition is determined, the only remaining state, the acceleration, can be calculated with Equation. (5.4) given the robot's initial speed and the initial flow speed.

Here is an example on initial conditions. While the pipeline is actually 200 meters long, and the water authority's estimation can have a 5% error due to inaccurate map. Thus the pipeline length estimation is 210 meters. It takes the robot 516 seconds to travel through this pipeline, so the average robot speed is $0.4m/s$. The average flow speed is assumed to be the same as the average robot speed, $0.4m/s$. From the system dynamics in Equation. (5.4), the expected initial acceleration is $1.8m/s^2$. The initial state estimate is then

$$\hat{X}(0|0) = \begin{bmatrix} \hat{s}(0|0) \\ \hat{\dot{s}}(0|0) \\ \hat{\ddot{s}}(0|0) \\ \hat{V_{flow}}(0|0) \end{bmatrix} = \begin{bmatrix} 0m \\ 0m/s \\ 1.8m/s^2 \\ 0.4m/s \end{bmatrix} \tag{5.18}$$

The initial aposteriori error covariance estimate should account for the expected errors in these estimations. The expected error ε_s in distance $s(0|0)$ and $\varepsilon_v$robot speed $\dot{s}(0|0)$ should be very small since they are known. The expected error ε_{vf} in the flow speed $V_{flow}(0|0)$ should be fairly large since the average flow speed is used in this place. It is assumed to be 25% of the average value to account for possible variations in the flow speed throughout time and distance. Given the inaccuracy in the flow speed estimation and the noise due to friction, the expected error ε_a in the initial acceleration estimation is assumed to be 100% of the acceleration value. Given those assumptions, the initial aposteriori error covariance estimate is

$$P(0|0) = \begin{bmatrix} (\varepsilon_s)^2 & 0 & 0 & 0 \\ 0 & (\varepsilon_v)^2 & 0 & 0 \\ 0 & 0 & (\varepsilon_a)^2 & 0 \\ 0 & 0 & 0 & (\varepsilon_{vf})^2 \end{bmatrix} \tag{5.19}$$

$$P(0|0) = \begin{bmatrix} (0.01m)^2 & 0 & 0 & 0 \\ 0 & (0.01m/s)^2 & 0 & 0 \\ 0 & 0 & (1.8m/s^2)^2 & 0 \\ 0 & 0 & 0 & (0.1m/s)^2 \end{bmatrix} \tag{5.20}$$

To start the EKF process, estimations of the system process noise and measurement noise must be defined through system identification. The system process error covariance matrix Q accounts for the expected variations in the flow rate and the friction. The overall structure of the process noise covariance matrix is

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & Q_{rr} & 0 & Q_{fr} \\ 0 & 0 & Q_{aa} & 0 \\ 0 & Q_{fr} & 0 & Q_{ff} \end{bmatrix} \tag{5.21}$$

The first row is the error covariance on the distance. There is no process noise that affect the distance estimate directly. Noise from friction and flow speed variations directly affects the robot's acceleration instead of distance, and thus these errors are propagated through the system model to the distance estimate.

The second and the fourth row are the error covariances related to the speed of the robot and the speed of the flow, which are highly correlated. The expected flow speed fluctuation can be reasonably assumed to be $0.1m/s$ for every 10 meters of distance. Taking into account that the system is updated at 50 Hz, the variance in the flow speed is

$$Q_{ff} = [\frac{0.1m/s}{10m * 50Hz} m/s]^2 = (0.0002m/s)^2 \tag{5.22}$$

Under the assumption that this passive, flow driven robot moves at almost the same speed as the water flow, the robot's speed is highly correlated to the flow speed. Thus, it is further assumed that

$$Q_{rr} = Q_{fr} = Q_{ff} \tag{5.23}$$

The third row is the error covariance related to the robot's acceleration. The friction is treated as a process noise to the acceleration only. The effect is assumed to be very small when compared to the effect of drag force. The process noise due to friction is assumed to be $Q_{aa} = (0.2m/s^2)^2$. The measurement covariance matrix R account for the errors in the accelerometer readings. This value can usually be looked up from the specifications of the accelerometer. The accelerometer on the robot is part of the Pololu minIMU v5, and its error variance $R = (0.5m/s^2)^2$.

When EKF is implemented with the above conditions, the result speed estimation is much better when compared to simple acceleration integration. As shown in Fig. 5-13, the robot speed estimation is bounded and close to the actual speed. However, there is still a large error in the distance estimation for two reasons. The first source of the error is due to inaccurate estimation of the total length of the pipeline, and thus inaccurate average speed. The other source of the error is that EKF over-constraints the speed estimation and does not capture the variation in the flow speed.

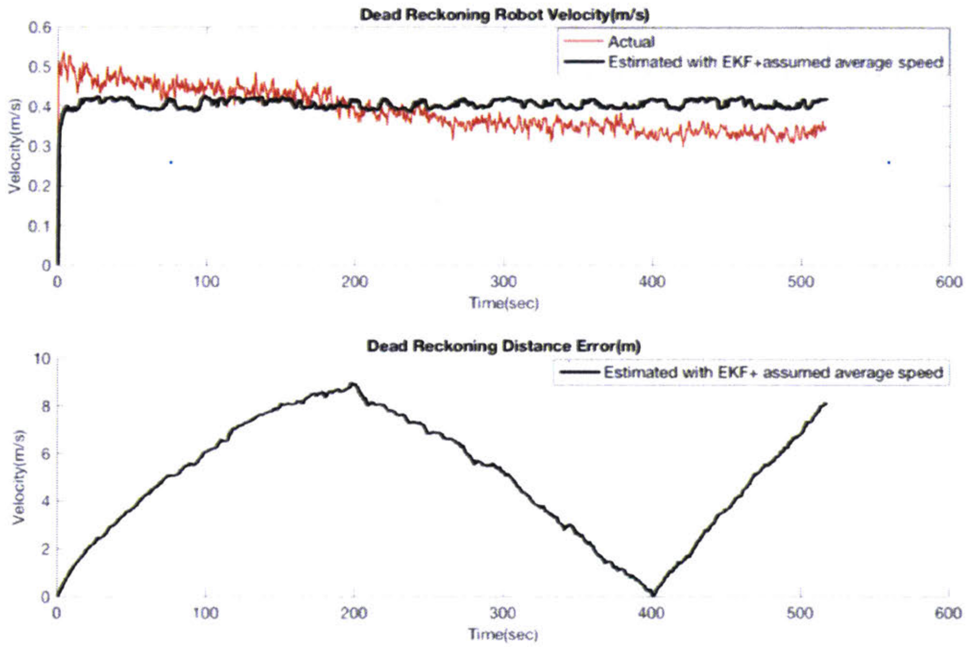


Figure 5-13: Challenge with conventional deadreckoning: fail to capture flow speed change over time and distance

The Matlab code for this result can be found in Appendix C.

Necessity to Obtain Speed Information

To capture the flow speed variation is the second difficulty for this EKF estimator. As Fig. 5-13 indicates, the assumption that the in-pipe flow rate is varying slowly is only valid for short period of time and distance. The active household connections along the pipeline reduce the in-pipe flow rate over distance, especially toward the end of pipeline. With the accelerometer, the robot can capture the fast fluctuations in its speed and the flow speed. Without actually measuring the speed, the robot cannot capture the slow variations in the flow speed due to active water usage.

There are additional information that can help the robot measure speed without adding more sensors the the robot. So far in the EKF setup, three out of the four available sources of information are used. They are

1. In-line acceleration, yaw rate, and tactile measurements of pipe features from the sensor
2. Understanding of the dynamics of the robot in a water pipe, as in Equation. (5.3)
3. Assumption that the in-pipe flow rate is varying slowly during the short period of robotic inspection

While the fourth source, the assumption that majority of the water pipeline are made with consecutive, same-length, standard pipe segments, has not been used at all. The concept is to identify those consecutive, same-length, standard pipe segments, and use their known length and measured passage time to obtain speed measurements. Inspired by the observations in the field tests, I propose to identify the joints between those consecutive, same-length, standard pipe segments from either the gyroscope data as in Fig. 5-10, or the tactile sensor data as in Fig. 5-11.

5.7 Method 1: Estimate Robot Speed From Consecutive Pipe Joints

In this method, the goal is to populate the blank robot speed measurement data space with actual data. This concept of populating the measurement data space to get more accurate estimation can be visualized in Fig. 5-14. If nothing is known about the actual process (the red curve), then the estimation about the process may be as inaccurate as the blue dash curve. If two or more measurements(black dots) are available, then the estimation of the process becomes the black curve and it is much more accurate. In the in-pipe localization case, the robot needs at least a few measurements of its speed in order to capture how the actual speed varies in time. In particular, the robot is obtaining speed measurements via searching for indicators that it passes consecutive, standard length pipe segments. The indicators are the pipe joints. The robot can obtain more accurate speed estimation for the short duration it travels from one pipe joint to the next one. Based on information source No. 4 that majority of the water pipeline are made with consecutive, same-length, standard pipe segments, the robot expects to find many of these pipe joints, and thus obtain speed estimation in many short periods. Those periods may be segmented and disconnected. Then with an algorithm, the robot can connect those data segments and create a smooth estimation of how the robot speed varies throughout the entire inspection. In comparison to the benchmark method, the robot now has more information to constraint its trajectory estimation. It is then more likely to generate an accurate estimation of its speed and distance.

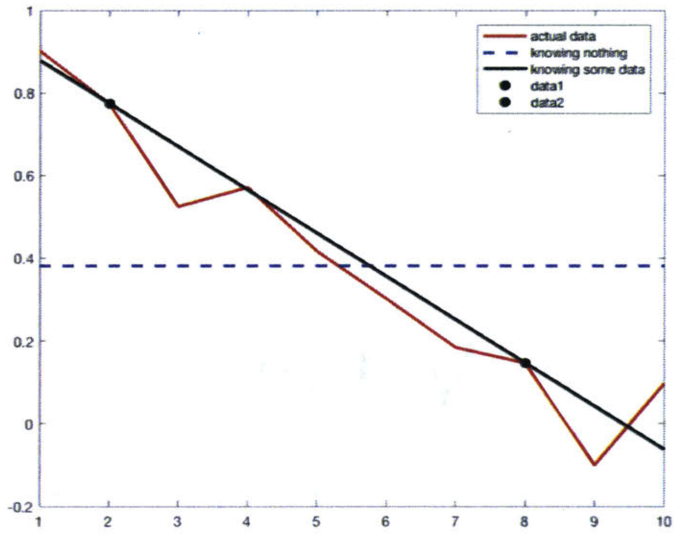


Figure 5-14: The Concept of Populating the Measurement Data Space

This method is performed in four steps. First, time stamps of all possible candidates of the pipe joints are identified. They may include the joints between standard, fix-length pipe segments, and also valves and other obtrusions in the pipeline. Then from these candidates, groups of pipe joints that are most likely to generate valid speed references are selected. These groups are denoted as High Confidence Zones. The other candidates that are not selected are grouped into the Low Confidence

Zones. The robot has low confidence in determining if they are joints, valves or other obtrusions in the pipeline. In an ideal pipeline with no valves, no bends and no tuberculations, all candidates should be identified as one High Confidence Zone. In the third step, the robot's speed trajectory within the High Confidence Zones are estimated using all four sources of information. This step actually also generated the boundary conditions for the Low Confidence Zones. In the last step, the robot's speed trajectory in the Low Confidence Zones are estimated with all available information and constraints.

Step 1: identify potential candidates for pipe joints with peak searching

All potential candidates of the pipe joints are extracted from the tactile measurement of the robot. From the field tests, it is observed that pipe joints have the tendencies to compress the robot radially from all sides and trigger impulses in all tactile sensors readings. Therefore, the occurrences of possible pipe joints can be identified from the tactile sensor measurements in three steps.

Step 1-1: Apply to the tactile sensor measurement a high pass filter with a very low cut-off frequency, e.g. 1Hz. It filters out the steady state value and its slow variations in the tactile sensor readings. The output of the filter highlights many peaks as shown in Fig. 5-11. This filter is performed on all channels of the tactile sensors.

Step 1-2: Search for all peaks in the filtered tactile sensor output. All time stamps of the peaks in filtered tactile sensor output are identified. The peaks can often be defined by magnitude, raising edge, falling edge or a combination of them. In my work, the peak is defined as a single data point that has the highest magnitude in its vicinity of a certain duration, e.g. 1 second. The peaks must exceed a magnitude threshold, e.g. 10% of maximum value in the filtered tactile output. This step is performed on all channels of the tactile sensors. For each peak, only the time stamp value is important, while its magnitude or width are dropped in the further analysis. This avoids the bias toward obtrusions that cause higher tactile sensor readings. A higher tactile sensor reading may indicate a larger obtrusion, but a larger obtrusion is not necessarily a pipe joint.

Step 1-3: Remove unlikely candidates via data correlation. In the example observer output, there are four channels of the tactile sensors. From the last step, the time stamps of peaks in every channel are obtained. From the field tests, it is observed that pipe joints are most likely to trigger impulses in all channels of tactile sensors. Therefore, by performing a correlation among all channels, one can find the candidates that are more likely to be pipe joints. In my work, this correlation is simple and binary; if there are peaks in all channels at the same time stamp. For every time stamp, if the correlation is one, then this time stamp is added to the queue of candidates. Each candidate represents the occurrence of a possible pipe joint. If the correlation is zero, then this time stamp is not added to the queue of candidates.

By performing step 1-1 to 1-3 on the tactile sensor output, a candidate queue is obtained. This queue is shown in Fig. 5-15. It can be told from Fig. 5-15 that in some part of the queue, the candidates are evenly spaced, such as those between t = 50 sec and t = 100 sec. There is a regular rhythm in them. Information source No. 3 in Table 5.5 states that flow speed and robot speed varies slowly and they can be treated as constant in a short period of time. Therefore, those candidates, evenly spaced in time, are very likely to be evenly spaced in distance. Those candidates may be a sequence of consecutive pipe joints, and the interval between each adjacent pair is a standard-length pipe segment. Meanwhile, in some other part of the queue, the candidates seem to be randomly distributed, such as those between t = 300 sec and t = 350 sec. There is not an obvious rhythm in them. Among these candidates there may be tuberculations, valves and actual pipe joints. It is difficult to tell them apart. Which are the useful candidates? Step 2 and 3 are designed to extract such information with high confidence out of this queue.

The Matlab code for this result can be found in Appendix C.

Step 2: identify High Confidence Zones with autocorrelation

The useful parts of the candidate queue are those with rhythms in them. Those are consecutive pipe joints, and in-between each adjacent pair of them is a standard

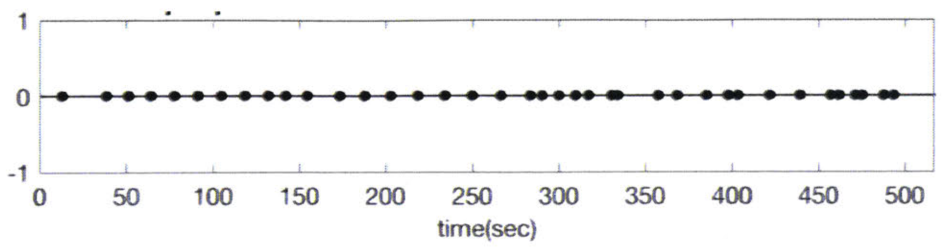


Figure 5-15: Step 1: identify potential candidates for pipe joints and record their time-stamp

pipe segment of length L_{seg}. In the US, the pipe standard defines $L_{seg} = 6$ meter. Knowing this distance expectation is critical. The candidate queue gives the time difference between each pair of the adjacent candidates. With time and expected distance known, the robot can obtain the average speed between two adjacent pipe joints. When combined with the acceleration measurement, the robot can accurately estimate its speed variation between each pair of adjacent pipe joints.

To identify the useful part of the candidate queue, a piece-wise autocorrelation algorithm is used. Autocorrelation is the standard method for identifying dominant periodic patterns in signals. When used on a piece of the candidate queue, e.g. $t = 50$ sec to $t = 100$ sec, it can identify the dominant period in this part of the queue, identify irregularities and predict the most likely next candidate in the queue. After repeating this procedure for all pieces in the candidate queue, all groups of consecutive pipe joints, denoted as a High Confidence Zones(HCZs), are identified. For example, after the candidate queue in Fig. 5-15 is processed, three HCZs are identified and highlighted in Fig. 5-16.

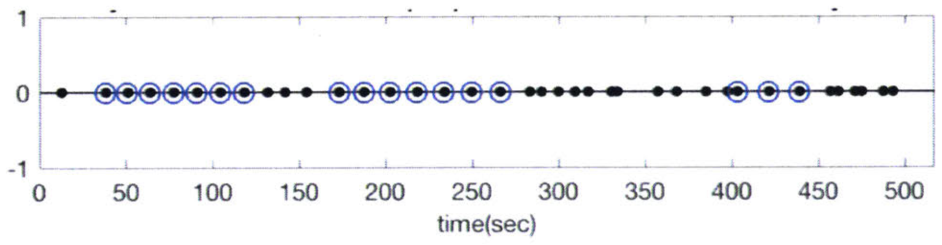


Figure 5-16: Step 2: identify high confident zones where there are consecutive, almost evenly spaced pipe joints

The details of the piece-wise autocorrelation algorithm is as follows. The Matlab

Table 5.6: Algorithm: Identify High Confidence Zone(HCZ) Through Piece-wise Autocorrelation

```
0  Given at least 6 candidates in the queue $t(1,2,3,..,k,..,k_{end})$
1  Initialize k=6, empty arrays $HCZ_{temp} = []$ and $HCZ = []$
2  Initialize an array y(t)=1 for t= for t=t(k-5), t(k-4), t(k-3), t(k-2), t(k-1), t(k),
   y(t)=0 everywhere else
3  Y(t(k-5)-1:t(k)+1)= Convolve y(t(k-5):t(k)) with a triangular wave
   h(t)=1+t when -1 $\leq$ t $\leq$ 0, h(t)=1-t when 0 $\leq$ t $\leq$ 1, and h(t)=0 everywhere else
4  Perform autocorrelation on Y(t(k-5)-1:t(k)+1), and find the non-zero delay,
   $\Delta t_{corr}$ with maximum correlation value
5  Predict the expected arrival time of the next candidate $\hat{t}(k+1) = t(k) + \Delta t_{corr}$
6  If $|\hat{t}(k+1) - t(k+1)| \leq 1|$, add k to the $HCZ_{temp}$; otherwise terminate $HCZ_{temp}$
7  If $HCZ_{temp}$ is terminated, and $HCZ_{temp}$ has at least 3 elements, then add
   $HCZ_{temp}$ as an element to $HCZ$
8  If k is not the end of the candidate queue, increment k=k+1, return to step 2
   otherwise terminate
```

implementation code can be found in Appendix C.

The algorithm can be visualized in the following two examples in action. The first example shows when a new pipe joint is recognized and added to the current High Confidence Zone queue. When the robot just records a new candidate at $t = 105$ sec, it has a non-empty array HCZ_{temp} that records the most recent sequence of believed pipe joints. Now it is trying to determine if it should believe this candidate $t(k) = 105$ sec as a pipe joint. So the robot performs autocorrelation on the latest 6 candidates, from $t(k-5) = 38$ sec to $t(k) = 105$ sec, as shown in Fig. 5-17-(1) and (2). The autocorrelation result indicate that the non-zero delay with maximum correlation value is about $\Delta t_{corr} = 13$ seconds. Thus the robot predicts that if $t(k) = 105$ sec is a pipe joint, then it should take another 13 seconds to pass the following standard pipe segment and arrive at the next pipe joint $\hat{t}(k+1) \approx 118$ sec. The robot is making this prediction based on relatively constant speed assumption. It is a derivation from information source No. 3 in Table 5.5: the assumption that the flow speed does not change much within short distance and duration. Then the robot waits until the recording of the next candidate and checks if the prediction is valid. The next candidate arrives at $t(k+1) = 118.8$ sec, within 1 sec of its prediction, as shown in Fig. 5-17-(c). The black line is the actual signal and the red line is the prediction.

The prediction is valid, so candidate $t(k) = 105$ sec is believed to be a pipe joint and added to the end of HCZ_{temp}.

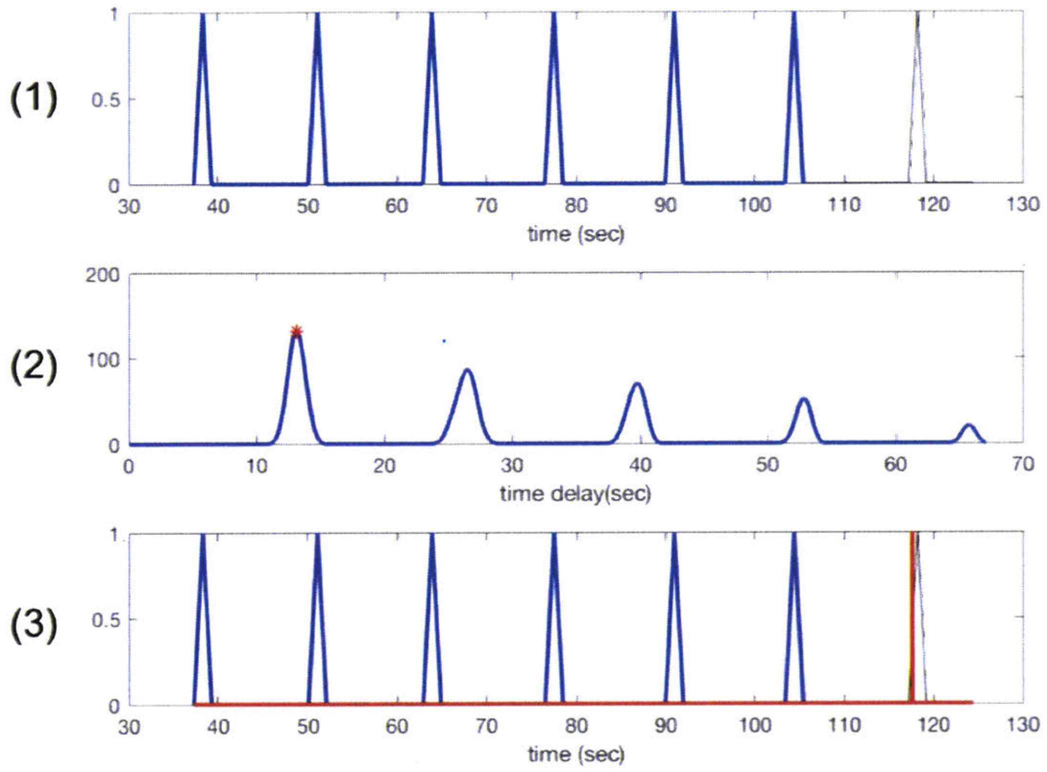


Figure 5-17: Step 2-1: Piece-wise autocorrelation as a method to identify consecutive pipe joints, example of satisfying condition

The second example shows when an extrusion is not recognized as a pipe joint and it terminates the current High Confidence Zone queue. As shown in Fig. 5-18, the robot proceeds to $t = 131$ sec and records a new candidate. It performs the autocorrelation on the latest 6 candidates and predicts the next candidate at $\hat{t}(k+1) = 145$ sec. However, the next candidate arrives at $t(k+1) = 142$ seconds, which is outside the 1 second tolerance zone of the prediction (red line in Fig. 5-18-(3)). There could be many possible reasons. Candidate $t(k) = 131$ sec could be a valve or a tuberculation. The pipe segment the robot passes between 131 sec and 142 sec may be a rare, irregular one. The flow may have accelerated during this period of time and so does the speed of this passive, flow-driven robot. The robot is unsure what have changed, so it chooses not to believe candidate $t(k) = 131$ sec is a regular pipe joint. At the same time, the robot terminates HCZ_{temp}, the recording of the latest High Confidence Zone, and push this HCZ_{temp} into memory HCZ which stores the recordings of all past High Confidence Zones.

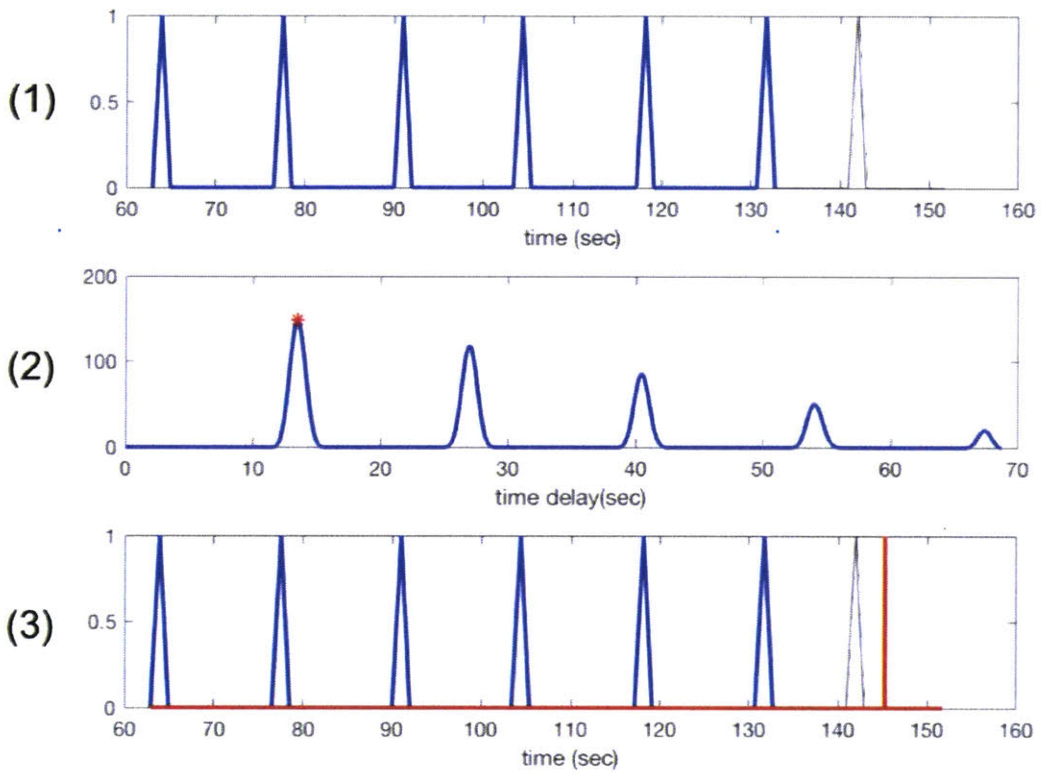


Figure 5-18: Step 2-2: Piece-wise autocorrelation as a method to identify consecutive pipe joints, example of non-satisfying condition

There are four important features in this HCZ identification algorithm. The first one is the window size. As the autocorrelation is performed on a piece of the candidate queue $y(t)$ of window size ΔT, this window size significantly affects the correlation result. If ΔT is too short, for example it only covers one of the recent candidate, then autocorrelation will produce nothing useful. In the other extreme, if ΔT is too long and covers all candidates in the past, the autocorrelation will not be able to capture recent changes in the candidate queue. Instead, it is very likely for the autocorrelation to produce the same delay prediction Δt_{corr} at every step. For example, if the flow speed changed recently and so was the delay between recent two candidates, autocorrelation on a long queue would yield a prediction as if this change never happened. Therefore, the window size must be chosen carefully; it should contain enough recent candidates to make a valid prediction, while not too many to neglect recent changes in the motion. After careful tuning, a feasible window size is found to be covering the last 6 candidates. If all 6 candidates were real pipe joints, then within this window of time the robot should travel through 5 standard pipe segments of length L_{seg}.

The second important feature is the triangular wave representation of the candidate queue. The candidate queue from Step 1 is a 1D array of time stamps, $t(1), t(2), .., t(k), .., t(k_{end})$. It does not produce anything useful to perform autocorrelation on this 1D array. Instead, this 1D array is translated into a 2D array y(t), where $y(t) = 1$ at the time stamp of all candidate $t(k)$ and $y(t) = 0$ everywhere else. When autocorrelation is performed on this 2D array, it can produce correlation value and the corresponding delay Δt_{corr}. However, a impulse train $y(t)$ as a representation of the candidate queue cannot account for variations in the flow speed. As the flow speed varies, the time it takes the robot to pass through one standard pipe segment of length L_{seg}also varies. The time delay between every pair of adjacent pipe joints are then not expected to be the same. One way to add tolerance to the algorithm and account for the variations in this time delay is to widen the impulse in $y(t)$ that represents each candidate. The duration of each impulse is increased from infinitesimal to $2\delta t$. This $2\delta t$ can be treated as tolerance or confidence interval. δt is determined by the expected change in the time to pass a standard pipe segment. In this case, $\delta t = 1$ sec.

$$\delta t = c_f \frac{L_{seg}}{\hat{V}_{flow}} = 0.06 \frac{6m}{0.4m/s} \approx 1sec \tag{5.24}$$

In the definition of δt, $\frac{L_{seg}}{\hat{V}_{flow}}$ is then the estimated time of passing one standard pipe segment. c_f is the expected maximum percentage change in the flow speed through a pipe segment. Based on two additional, logical assumptions, c_f is chosen to be 0.06. The first assumption is the flow speed change is mainly due to the active household connections, each of which reduces the flow rate by $\Delta V_f = 3\%$. The second assumption is that there are at most 2 household connections within the distance of one pipe segment length $L_{seg} = 6$ meters.

$$c_f = 2\Delta V_f = 2 * 0.03 = 0.06 \tag{5.25}$$

δt is also dependent on $\hat{V}_{flow}$, the guessed flow speed. From the benchmark algorithm in the last section, $\hat{V}_{flow}$ is estimated to be around $0.4m/s$. If the robot is asked to make conservative predictions or not make guesses for $\hat{V}_{flow}$, it can always start with a relatively large $\delta t = 3$ seconds for example, and gradually converge to a smaller value as it collects more data in the pipe.

The third important feature is the prediction of the expected arrival time of the next candidate. If the current candidate $t(k)$ is a true pipe joint, it should be followed by a standard pipe segment of length L_{seg}. The robot is expected to see another pipe joint after passing through this this standard pipe segment. The robot is predicting this delay, or passage time based on the relatively constant speed assumption. It is not expected that the flow speed changes much within one pipe segment. If there are any changes in the flow speed and thus in the delay, it can be accounted by the tolerance in the prediction, which is the same as δt.

The forth and last important feature is the minimum length of a valid High Confidence Zone. While majority part of the pipeline is made of standard pipe segments, there are randomly distributed valves, tuberculations and other false positives for pipe joints. To minimize the possibility of mistaking a false positive into the High Confidence Zone, a valid High Confidence Zone must contains multiple consecutive pipe joints. After careful tuning, the minimum number of pipe joints in a High Confidence Zone is found to 3. At this setting, the robot identified three High Confidence Zones as indicated in Fig. 5-16, and the maximum length of any High Confidence Zones is 6. If this minimum number is set high, 7 for example, then the robot will derive zero High Confidence Zones.

Step 3: Speed Estimation Inside High Confidence Zones

Within each High Confidence Zone, robot's speed can be determined with high accuracy. The robot will temporarily neglect all the data outside the High Confidence Zones, and focus on the state estimation within the High Confidence Zone, as indicated in Fig. 5-19. Within each High Confidence Zones, the robot believes the distance between each pair of adjacent pipe joints to be L_{seg}, and it can measure the time of passage to further estimate its average speed between each pair of adjacent pipe joints. Now the robot knows its acceleration, its estimation model based on its dynamics, and two more pieces of newly available information: the total distance between these two pipe joints and the travel time between them. In addition to the assumption that the flow speed does not vary much within this short period and distance, the robot now has enough information to accurately estimate its speed variation throughout the High Confidence Zone.

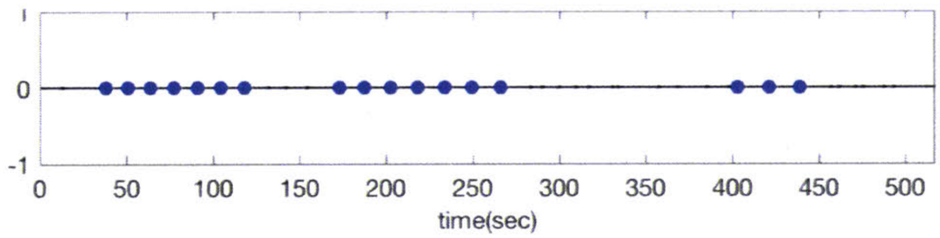


Figure 5-19: Step 3 Input: Focus on High Confidence Zones (blue dots), estimate the robot speed within these zones first

The speed estimation is performed through a data fusion process. There are two streams of measurement data: the acceleration data that is collected regularly at a high sampling rate, and the distance data that is only available once when the robot arrives at the next pipe joint. Since these two streams of data are collected at different sampling rate, the previous EKF in Equation (5.11-5.17) is not capable of fusing them together. It needs to be augmented with another data fusion algorithm. While the distance data is unavailable, the EKF can make predictions and estimate the robot's motion based on model and acceleration. This process can be visualized as the dash line in Fig. 5-20. From time $t1$ to $t2$, the EKF gives the estimated state trajectory starting with initial condition at $t1$, and end with a predicted state at $t2$

with a large error covariance indicated by the dash circle. When the robot arrives at the next pipe joint at time $t2$, it knows it has traveled a distance of L_{seg} from the last pipe joint and thus obtained a new measurement about its state at $t2$. This measured state has a much smaller error covariance, or in another word, much higher confidence. This measured state can be different from the state predicted from EKF. In order to incorporate this distance measurement and correct the robot's past state estimation that leads up to the current state, Rauch-Tung-Striebel (RTS) smoother algorithm [51] is implemented. RTS algorithm revisits the past state estimations and creates a new smooth estimated state trajectory from $t1$ to $t2$, as shown by the solid curve in Fig. 5-20. The state at $t1$ can be treated as the initial condition in this piece of the localization problem from time $t1$ to $t2$, and the state at $t2$ can be treated as the final condition. EKF+RTS is one of the optimal algorithms to extrapolate the trajectory from initial condition to the final condition under all constraints. It is optimal because it is expected to produce the minimum error covariance.

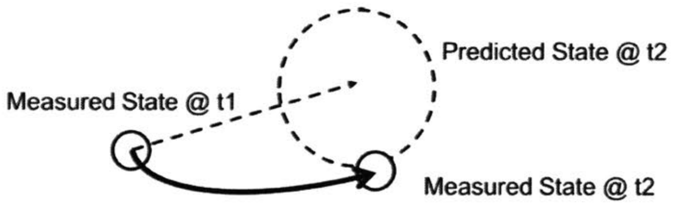


Figure 5-20: Illustration of the effect of RTS smoother

Here are the details of the RTS implementation. Assume currently the robot is at pipe joint k_{n+1} within a High Confidence Zone and it lastly visited pipe joint k_n. Define the initial condition at $t(k_n)$ as $\hat{X}(k_n|k_n)$ with error covariance $P(k_n|k_n)$.

$$\hat{X}(k_n|k_n) = \begin{bmatrix} \hat{s}(k_n|k_n) \\ \dot{\hat{s}}(k_n|k_n) \\ \ddot{\hat{s}}(k_n|k_n) \\ \hat{V}_{flow}(k_n|k_n) \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{L_{seg}}{(k_{n+1}-k_n)\Delta t} \\ Y(k_n) \\ \frac{L_{seg}}{(k_{n+1}-k_n)\Delta t} \end{bmatrix} \tag{5.26}$$

$$P(k_n|k_n) = \begin{bmatrix} (0.01m)^2 & 0 & 0 & 0 \\ 0 & (0.02\dot{\hat{s}}(k_n|k_n))^2 & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & (0.02\hat{V}_{flow}(k_n|k_n))^2 \end{bmatrix} \tag{5.27}$$

Three assumptions are made in the initial state. The first assumption, the initial distance $\hat{s}(k_n|k_n) = 0$, is made for simplicity in the calculation. The goal is to estimate the robot's speed, so precise distance from the starting point is not necessary. As long as the robot knows that it takes the time it takes to go from one pipe joint to the next and the distance is equivalent to the length of a standard pipe segment, L_{seg}. The second assumption is about the robot's speed at k_n. Although the exact $\dot{\hat{s}}(k_n|k_n)$ is still unknown to the robot, it can take the average speed in this pipe segment between joint k_n to joint k_{n+1} as an estimation. Given the overall assumption that the flow speed does not vary much in short internal and short distance, this approximation of $\dot{\hat{s}}(k_n|k_n)$ is reasonable. The third assumption is that the flow speed at k_n and k_{n+1} is set to be the same as the robot speed $\dot{\hat{s}}$ at k_n. This is a valid assumption given that the robot is passive and flow driven.

The initial error covariance matrix (Eq. 5.27) is carefully chosen to compensate for the inaccuracies in the initial state assumptions. Standard deviation in the error of the speed estimations are assumed to be a small percentage (2%) of the expected value. The error variance in the acceleration is the measurement error covariance $R = (0.5m/s^2)^2$ since the robot has the acceleration measurement. Error covariance in the distance cannot be zero, or the P matrix is not positive definite or invertible. The P matrix must be positive definite.

Given these initial conditions, perform an EKF as described in Equations (5.11-5.17) from $t(k_n)$ to $t(k_{n+1})$. EKF will predict its own $\hat{X}(k_{n+1}|k_{n+1})$ and $P(k_{n+1}|k_{n+1})$. However, the robot choose to believe the the measurement at k_{n+1} since the additional distance data is available. The final conditions in the interval $t(k_n)$ to $t(k_{n+1})$ are updated to be

$$\hat{X}(k_{n+1}|k_{n+1}) = \begin{bmatrix} \hat{s}(k_{n+1}|k_{n+1}) \\ \dot{\hat{s}}(k_{n+1}|k_{n+1}) \\ \ddot{\hat{s}}(k_{n+1}|k_{n+1}) \\ \hat{V}_{flow}(k_{n+1}|k_{n+1}) \end{bmatrix} = \begin{bmatrix} L_{seg} \\ \frac{L_{seg}}{(k_{n+2}-k_{n+1})\Delta t} \\ Y(k_{n+1}) \\ \frac{L_{seg}}{(k_{n+2}-k_{n+1})\Delta t} \end{bmatrix} \tag{5.28}$$

$$P(k_{n+1}|k_{n+1}) = \begin{bmatrix} (0.01m)^2 & 0 & 0 & 0 \\ 0 & (0.02\dot{\hat{s}}(k_{n+1}|k_{n+1}))^2 & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & (0.02\hat{V}_{flow}(k_{n+1}|k_{n+1}))^2 \end{bmatrix} \tag{5.29}$$

The final conditions are chosen with mostly the same logics as the the initial conditions, except two items. The first one is the final distance, $\hat{s}(k_{n+1}|k_{n+1}) = L_{seg}$. To estimate the robot's speed, the relative displacement from the last pipe joint, L_{seg}, is sufficient. The second item is the robot's final speed inbetween joint k_n to joint k_{n+1}, denoted as $\dot{\hat{s}}(k_{n+1}|k_{n+1})$. Although the exact value is unknown to the robot, the robot can use the average speed in the next pipe segment between joint k_{n+1} to joint k_{n+2} as $\dot{\hat{s}}(k_{n+1}|k_{n+1})$. The question is then whether this assumption is valid for the last pipe joint in a High Confidence Zone. The answer is yes. Given High Confidence Zone identification method described in Step 2, each pipe joint in a High Confidence Zone is always followed by a standard pipe segment. Therefore, even if k_{n+1} is not the end of a High Confidence Zone, it is still followed by a standard pipe segment and the average speed within that pipe segment can be known.

Then RTS smoother is performed to update all past states by working backward from $t(k_{n+1})$ to $t(k_n)$. First, the RTS gain is calculated for each time stamp t(k) between $t(k_{n+1})$ and $t(k_n)$, starting with $k = k_{n+1} - 1$.

$$C(k) = P(k|k)A_{est}(k)^T P(k+1|k)^{-1} \tag{5.30}$$

Then the state estimation $\hat{X}(k|k_{n+1})$ is updated.

$$\hat{X}(k|k_{n+1}) = \hat{X}(k|k) + C(k)(\hat{X}(k+1|k_{n+1}) - \hat{X}(k+1|k)) \tag{5.31}$$

It is worth to note that the notation of $\hat{X}(k|k_{n+1})$. Just like in the Extended Kalman Filter, here the state of interest is $\hat{X}(k)$, and it is conditioned on the state $\hat{X}(k_{n+1})$. After the state estimation update, its error covariance is also recalculated.

$$P(k|k_{n+1}) = P(k|k) + C(k)(P(k+1|k_{n+1}) - P(k+1|k))C(k)^T \tag{5.32}$$

Now the state and error covariance for time stamp $t(k)$ are updated, the step counter k is updated $k = k-1$ to move onto an earlier state. Equation (5.30 -5.32) is repeated for the new k. The process terminates when $k = k_n$. That is when the robot was at the previous pipe joint.

After the EKF+RTS is performed for each pair of adjacent pipe joints in all High Confidence Zones, the robot obtains a good estimation of its speed trajectories in those High Confidence Zones. The result in this example is shown in Fig. 5-21. As the colored curves in the figure indicates, the EKF+RTS algorithm is performed 14 times for each of the 14 pairs of adjacent pipe joints. The result captures the both the fast fluctuation and the slow steady state shift in the robot speed. This result is already a major improvement on the benchmark in Fig. 5-13, which is regular EKF with assumed average speed.

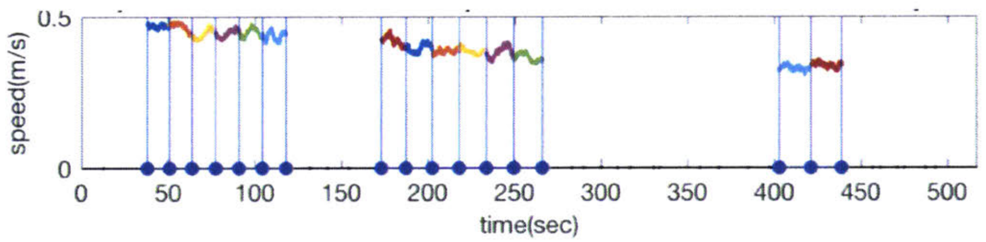


Figure 5-21: Step 3 outcome: Example robot speed estimation within High Confidence Zones

Step 4: Speed Estimation In Low Confidence Zones

However, the speed estimation outside the High Confidence Zones is still missing. These areas are denoted as Low Confidence Zones. In this step, the robot will attempt to estimate these missing pieces of speed trajectory with information from the High Confidence Zones. The idea is the same; populate the speed measurement data space in order to capture the speed variations. As described in Fig. 5-14, the robot need to obtain at least two data points of speed measurement before it can develop an accurate speed trajectory estimation. There are already two data points available, one at the end of each Low Confidence Zones. These are also the robot speed at the end of High Confidence Zones.

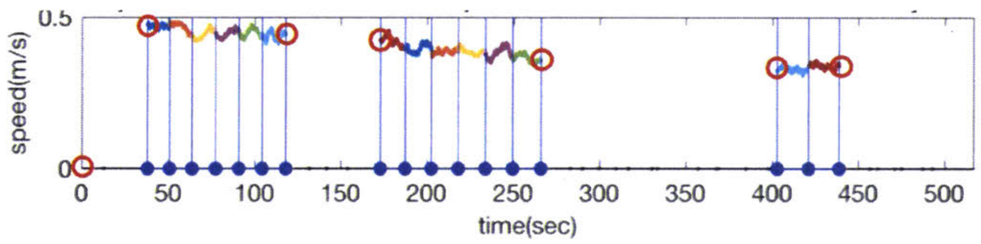


Figure 5-22: Step 4: Estimate robot speed outside High Confidence Zones, with initial and final condition derived from High Confidence Zones

The same RTS algorithm is implemented to produce the speed trajectory estimation within the Low Confidence Zones with differently formulated initial and final conditions. Since no distance information is available in the Low Confidence Zones, the RTS smoother should neglect distance correction. Instead, the RTS smoother will correct speeds. For example, if k_n is the end of the previous High Confidence Zone and k_{n+1} is the beginning of the next High Confidence Zone, then the duration between $t(k_n)$ and $t(k_{n+1})$ is a Low Confidence Zone. The initial condition for this Low Confidence Zone is then transferred from the $\hat{X}(k_n|k_n)$ at the end of the last High Confidence Zone. The distance value is reset to zero for simplicity in calculation; its absolute value is unknown and not important here.

$$\hat{X}(k_n|k_n) = \begin{bmatrix} 0 \\ \dot{\hat{s}}(k_n|k_n) \\ \ddot{\hat{s}}(k_n|k_n) \\ \hat{V}_{flow}(k_n|k_n) \end{bmatrix} \tag{5.33}$$

$$P(k_n|k_n) = \begin{bmatrix} (0.01m)^2 & 0 & 0 & 0 \\ 0 & (0.02\dot{\hat{s}}(k_n|k_n))^2 & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & (0.02\hat{V}_{flow}(k_n|k_n))^2 \end{bmatrix} \tag{5.34}$$

Given these initial conditions, perform EKF as described in Equation (5.11-5.17) from $t(k_n)$ to $t(k_{n+1})$. EKF will predict $\hat{X}(k_{n+1}|k_{n+1})$ and $P(k_{n+1}|k_{n+1})$. Since the robot has a new speed measurement from the next High Confidence Zone, it will update the velocity part of the predicted $\hat{X}(k_{n+1}|k_{n+1})$ with the speed measurement. Since the robot has no new information about its distance, it will not update the distance part of the $\hat{X}(k_{n+1}|k_{n+1})$.

$$\hat{X}(k_{n+1}|k_{n+1}) = \begin{bmatrix} \hat{s}(k_{n+1}|k_{n+1}) \\ \dot{\hat{s}}(k_{n+1}|k_{n+1}) \\ \ddot{\hat{s}}(k_{n+1}|k_{n+1}) \\ \hat{V}_{flow}(k_{n+1}|k_{n+1}) \end{bmatrix} = \begin{bmatrix} \hat{s}(k_{n+1}|k_{n+1}) \\ \frac{L_{seg}}{(k_{n+2}-k_{n+1})\Delta t} \\ Y(k_{n+1}) \\ \frac{L_{seg}}{(k_{n+2}-k_{n+1})\Delta t} \end{bmatrix} \tag{5.35}$$

The error covariance $P(k_{n+1}|k_{n+1})$ in the final condition is also updated accordingly. The error covariance in the distance stays the same as that from the EKF results. The error covariance of the speeds are tightened in comparison to that from the EKF results. After careful tuning, the standard deviation of the speeds is chosen to be 20% of the expected speed values. It is much bigger than its counterpart in the High Confidence Zones to reflect the fact that the robot has lower confidence in the estimations outside the High Confidence Zones.

$$P(k_{n+1}|k_{n+1}) = \begin{bmatrix} P(k_{n+1}|k_{n+1})[1,1] & 0 & 0 & 0 \\ 0 & (0.2\hat{\dot{s}}(k_{n+1}|k_{n+1}))^2 & 0 & 0 \\ 0 & 0 & R & 0 \\ 0 & 0 & 0 & (0.2\hat{V}_{flow}(k_{n+1}|k_{n+1}))^2 \end{bmatrix} \tag{5.36}$$

Then the same RTS smoother as in Equation (5.30 -5.32) is performed from $\hat{X}(k_{n+1}|k_{n+1})$ to $\hat{X}(k_n|k_n)$. RTS updates the past state trajectory estimation based on the discrepancy between the EKF estimation of the final conditions and the measured final conditions. Since the estimation of the distance and its error covariance in the final state are the same as that from the EKF, the RTS smoother will not update the distance estimations. The other states in the RTS estimation are unaffected by the distance state. It is similar to perform the EKF and RTS in the Low Confidence Zones on three states $[\dot{s}, \ddot{s}, V_{flow}]^T$ instead of the four states $[s, \dot{s}, \ddot{s}, V_{flow}]^T$ as in the High Confidence Zones. Instead of reducing the states and write separate code for the Low Confidence Zones, the algorithm achieves the same goal in a more concise way by change the initial and final conditions.

This algorithm works in-between the High Confidence Zones, but it needs to be changed for the Low Confidence Zones before the first High Confidence Zone and after the last High Confidence Zone. Each of these two Low Confidence Zones are connected to only one High Confidence Zone so there is only one boundary condition available. The other boundary condition must be derived in another way. For example, in the first Low Confidence Zone starting at time $t = 0$ sec, the initial condition is assumed to be

$$\hat{X}(0|0) = \begin{bmatrix} \hat{s}(0|0) \\ \hat{\dot{s}}(0|0) \\ \hat{\ddot{s}}(0|0) \\ \hat{V}_{flow}(0|0) \end{bmatrix} = \begin{bmatrix} 0m \\ 0m/s \\ \frac{c_d \rho A(m+m_a)}{2}(V_{flow}(k_1) - \dot{s}(k_1))^2 \\ \hat{V}_{flow}(k_1) \end{bmatrix} \tag{5.37}$$

$$P(0|0) = \begin{bmatrix} (0.01m)^2 & 0 & 0 & 0 \\ 0 & (0.01m/s)^2 & 0 & 0 \\ 0 & 0 & (\ddot{s}(0|0))^2 & 0 \\ 0 & 0 & 0 & (0.2\hat{V}_{flow}(0|0))^2 \end{bmatrix} \tag{5.38}$$

where the robot is certain that it starts at distance 0 and initial speed 0. The initial flow speed is assumed to be the same as the flow speed at the beginning of the first High Confidence Zone. The initial acceleration is estimated from the drag force equation as in Eg. 5.4.

The RTS smoother is not performed on the Low Confidence Zone after the last High Confidence Zone. There is not a final condition so the robot cannot perform effective RTS smoother in this zone. Thus only the EKF is performed in this zone. The initial condition for the EKF also follows Equation (5.33 and 5.34).

The output of Step 4 completes the robot's speed trajectory estimation. For example, as shown in Fig. 5-23, this Low Confidence Zone algorithm is performed four times in the four blank regions outside the High Confidence Zones, and fills in the robot's speed estimations. Now the robot has a complete speed trajectory estimation. This estimation indicates that the robot is slowing down toward the end of the pipeline, and so is the water flow speed. It agrees with the fact that multiple active household connections are drawing water from the pipeline and reducing the water flow. The combined algorithm from step 1 to 4 is able to capture this spatial variation in the pipe flow. From the speed trajectory, the robot can integrate the speed to produce in-line distance at every point of time, and complete the localization.

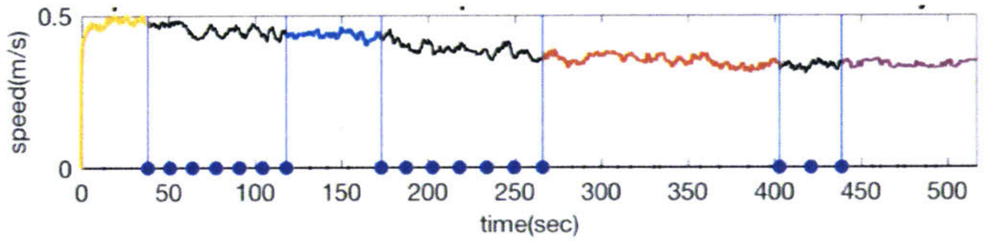


Figure 5-23: Step 4 outcome: Example robot speed estimation outside High Confidence Zones

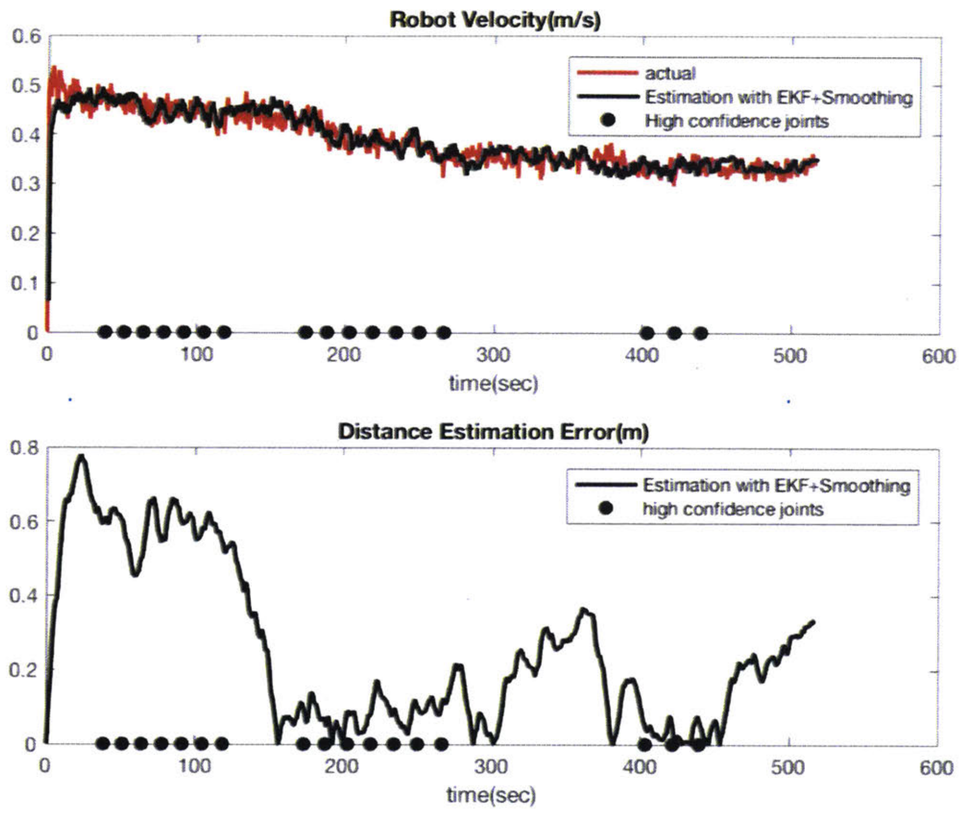


Figure 5-24: Localization result of the Consecutive High Confidence Joints method

The method of obtaining robot speed information from consecutive pipe joints produces accurate localization results at very low cost. As the comparisons in Fig. 5-24 indicate, the robot is able to track its speed with this method well and produces an in-line distance tracking error on the order of 1 meter in this 200-meter run. The tracking error is about 0.5 % of the total distance of the robotic inspection. In comparison to the benchmark as in Fig. 5-13, this method produces an order of magnitude lower distance error, and it is uniquely capable of capturing the slow variations in the robot speed and flow speed throughout the pipeline. Moreover, this performance is achieved by a robot with IMU but not a single actual speed sensor. All speed measurements are derived from the tactile measurements of the pipe joints. These speed measurements are not available at every time step as an actual speed sensor can produce; instead this method only produces speed measurements within the High Confidence Zones. However, given the dynamic constraint in a typical water pipeline that water flow changes slowly, these few isolated pieces of speed information sufficiently constraint the robot's estimation and limit the error propagation. This is a low cost, minimum sensing requirement approach to in-pipe robot localization problem.

It is also worth attention how the distance tracking error changes in different zones. As shown in Fig. 5-24, the distance tracking error is almost constant in the High Confidence Zones. This is the effect of the RTS smoother, especially due to the distance constraint in the boundary conditions. The RTS smoother makes sure the robot's distance estimation within each pair of pipe joints in the High Confidence Zones are bounded to the pipe segment length. This keeps the distance tracking error bounded within the High Confidence Zones.

In comparison, the distance tracking error in the Low Confidence Zones is a problem. The tracking error fluctuates significantly within each Low Confidence Zone. Since there is no information available about either the absolute or relative distance in any Low Confidence Zones, this tracking error is expected to scale up with the duration of the Low Confidence Zones. It sets the lower bound of the total tracking error.

There are at least two feasible ways to address this Low Confidence Zone Challenge and further improve the localization accuracy. The first approach is to identify more joints in the candidates. If the robot can identify pipe joints more effectively and keep each of the Low Confidence Zones short, the total tracking error can be further reduced. It is possible to improve Step 2 the identification of High Confidence Zones with other pattern recognition methods than autocorrelation.

The other approach is to obtain more information out of the candidate queue, especially those outside the High Confidence Zones. In Step 3 and 4, all the candidates outside the High Confidence Zones are neglected. Is it possible to reuse these candidates and extract from them information for robot localization? The robot can then populate the speed measurement space with more data points than it can with the current method, and produce more accurate speed and distance tracking. This seems impossible to achieve with the current on-board sensors. However, what is the minimum number of additional sensors to enable the robot to collect useful information from each candidate in the queue? In the next section, I propose a second method to address these two questions.

5.8 Method 2: Tactile Speed Sensor

To enable the robot to collect speed measurements in the Low Confidence Zones, the robot is modified to include a tactile speed sensor. In the last section, the robot utilizes a ring of tactile sensors to identify if the robot passes a pipe joint. The speed measurement is obtained through dividing the known distance between two adjacent pipe joint and the time it takes the robot to go from one pipe joint to the next. Thus the speed measurement is only obtainable when the robot passes multiple consecutive pipe joints. The robot cannot obtain speed measurements inside the Low Confidence Zones because it cannot know if it has passed multiple consecutive pipe joints. To populate the speed data space inside each Low Confidence Zone, the robot can be modified to measure its speed from passing only one pipe joint or a single obtrusion. A tactile speed sensor is designed to deliver this capability. This speed sensor is simply two rings of original tactile sensors separated at a known distance, as shown in Fig. 5-25. When the robot passes an obtrusion, both ring of tactile sensors will be bended by the obtrusion and register tactile signals. There is going to be a delay between the two tactile sensors from the two rings, since the two rings arrive at the obtrusion one after another. Dividing the known distance between the two rings by this time delay, the robot can obtain its instantaneous speed over this obtrusion.

There are two advantages this tactile speed sensor has over the other type of speed sensors on this robot. First, it is easy to integrate. There is already a ring of tactile sensors integrated in the robot. To construct this tactile speed sensor, the same ring of tactile sensors are simply duplicated. The integration is much easier when compared to adding wheel encoders, acoustic range finders or vision systems. The

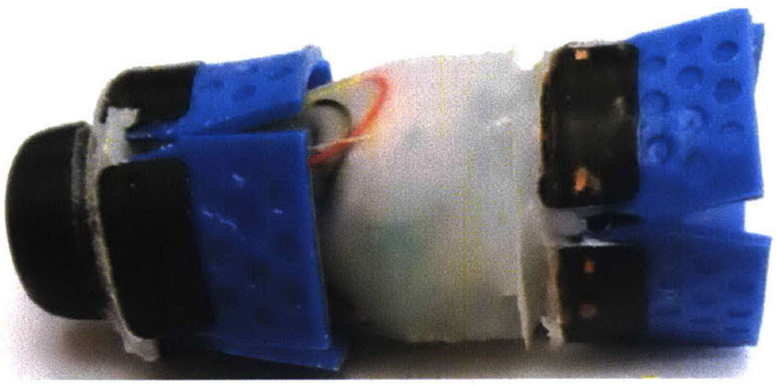


Figure 5-25: Robot prototype with the tactile speed sensor for measuring instantaneous speed at every in-pipe obtrusion

second advantage is that the tactile speed sensor is multipurpose. Collectively this array of multiple tactile sensors can be used to measure speed. More importantly each individual tactile sensor is a leak sensor. It can be used to monitor the occurrence and magnitude of leaks in the pipe. If those tactile sensors are constructed in the same way as the soft bending angle sensors in the previous chapter of this thesis, they can also detect the height of obtrusions in the pipe. This two-ring arrangement adds another functionality to the leak sensors.

Accuracy of the Tactile Speed Sensor

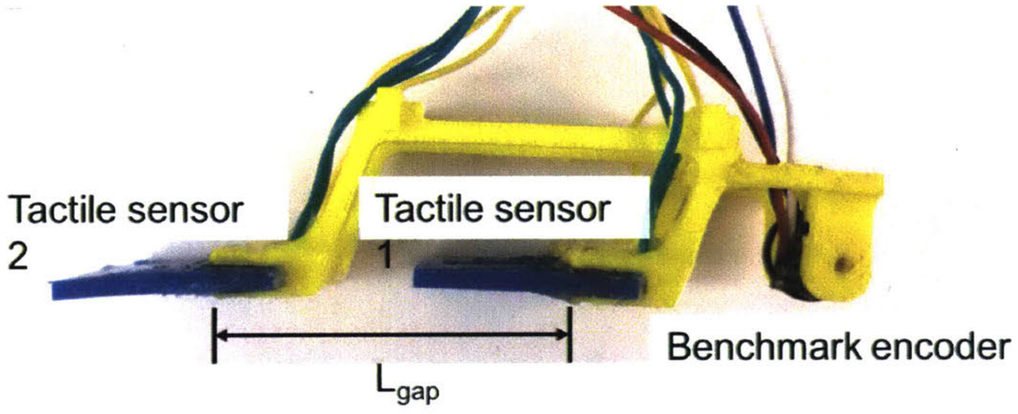


Figure 5-26: Experimental device for evaluating the accuracy of the tactile speed sensor

To determine its accuracy, the tactile speed sensor is compared to a wheel encoder in a benchmark test. A test vehicle as shown in Fig. 5-26 is constructed. It is very similar to the test vehicle used in the soft bending angle sensor experiments as in Fig. 4-20. In the front (right side of the figure), there is an optical wheel encoder. In the back, there are two soft tactile sensors placed L_{gap} apart. The bottom of the tactile sensors and the bottom of the wheel encoder are aligned. When the vehicle is pushed to the right on a non-slip surface with multiple obtrusions, both the wheel encoder and the tactile speed sensors will produce speed measurements. If the wheel encoder is assumed to be perfectly accurate, the difference between the output of the tactile speed sensor and that of the encoder is the error in the tactile speed sensors.

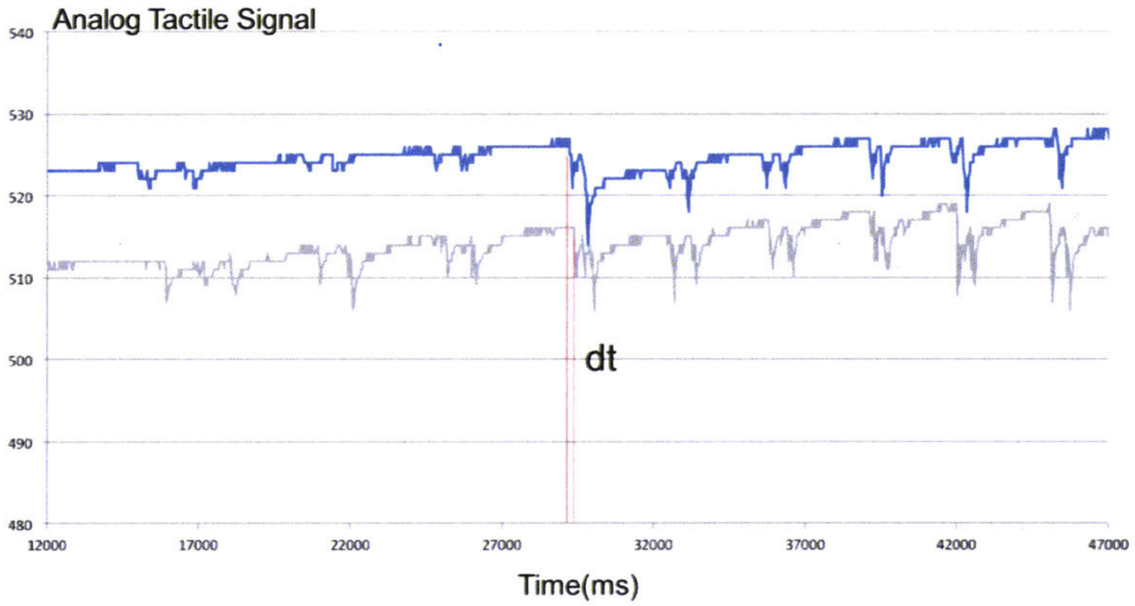


Figure 5-27: Sample experiment output from the Experimental device in Fig. 5-26. The blue trace is the data from the tactile sensor in the front. The gray trace is the data from the tactile sensor in the back. The device ran over two consecutive obtrusions at various speed in each trial, and data for ten trials are reported here.

The tactile speed sensor estimates the device's speed by comparing the signals from the two tactile sensors. The signals from the two tactile sensors are shown in Fig. 5-27. The blue curve is the reading from the tactile sensor in the front of the vehicle, and the gray curve is from the sensor in the back. For every obtrusion, the front sensor signal leads the back sensor signal. The time delay between them, measured from the rising edge in the front signal to the next closest rising edge of the back signal, is denoted as dt. The distance between the front and the back sensor is known to be L_{gap}. This means it takes the robot dt to move forward a distance of L_{gap}, and the robot's speed $\dot{s}$ over this obtrusion k is estimated to be

$$\dot{s}(k) = \frac{L_{gap}}{dt} \tag{5.39}$$

The error in the tactile speed sensor output scales with its time resolution and the vehicle's speed. Tactile speed sensor produces speed estimation based on time measurement. If measurement dt is off by one resolution $1/f_s$, then the speed error $\delta\dot{s}$ is

$$\delta\dot{s} = |\frac{L_{gap}}{dt_{true} + 1/f_s} - \frac{L_{gap}}{dt_{true}}| = |\frac{L_{gap}}{L_{gap}/\dot{s} + 1/f_s} - \dot{s}| = \frac{\dot{s}}{\frac{f_s L_{gap}}{\dot{s}} + 1} \tag{5.40}$$

For example, if the true speed is $\dot{s} = 100mm/s$ and the sensor sampling rate is $f_s = 50$ Hz, the expected speed measurement error is calculated to be $\delta\dot{s} = 4mm/s$. This prediction agrees with the experimental result. In 10 experiments with the vehicle is moving at a speed around $100mm/s$, the average difference between speed estimation from the tactile speed sensor and that from the encoder is $5mm/s$.

When the robot travels at high speed, it needs to sample the tactile sensors at a higher rate in order to measure its speed accurately. In the simulation example used in this Chapter, the robot speed is around $\dot{s} = 500mm/s$. At $f_s = 50$ Hz, the expected error is $\delta\dot{s} = 80mm/s$. In order to limit the error within $5mm/s$ or 5% of the actual speed, the sampling rate must be increased to $f_s = 200$ Hz. This sampling rate is still obtainable by commercially available micro-controller platforms such as the ESP32 this robot is using.

Integrate Tactile Speed Sensor readings in Localization

The Consecutive Pipe Joints method described in the last section needs minimal modification to integrate the tactile speed sensor data. All algorithms stays the same, and the only changes are in the boundary conditions in Step 4 where the robot speed inside Low Confidence Zones are estimated. Given the tactile speed sensors provide an instantaneous speed measurement at every candidate including those in the Low Confidence Zones, the robot can connect those instantaneous speed measurements to produce the speed trajectory in the Low Confidence Zones. Instead of connecting from the end of the last High Confidence Zone to the beginning of the next High Confidence Zone in a single EKF+RTS operation, the robot can now perform the same operation between each pair of adjacent candidates in the Low Confidence Zones. As shown in Fig. 5-28, the speed measurement at each candidate is now available. For each pair of adjacent candidates in the Low Confidence Zones, the robot can take the measured speeds and accelerations as the initial and final condition, and use the same EKF+RTS as in Step 4 in the last section to estimate the speed trajectory from a candidate to the next one.

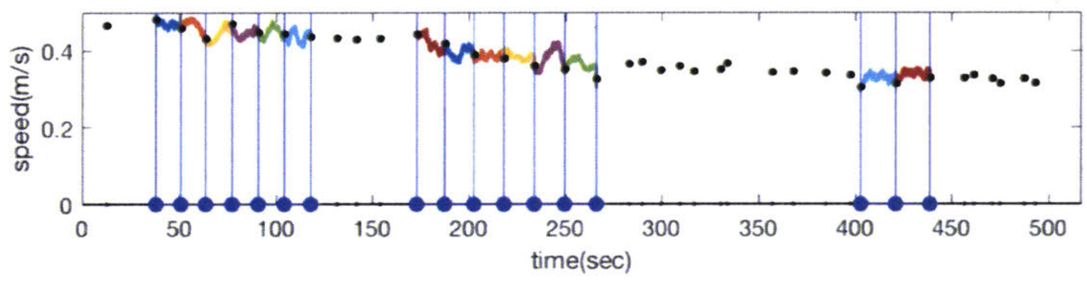


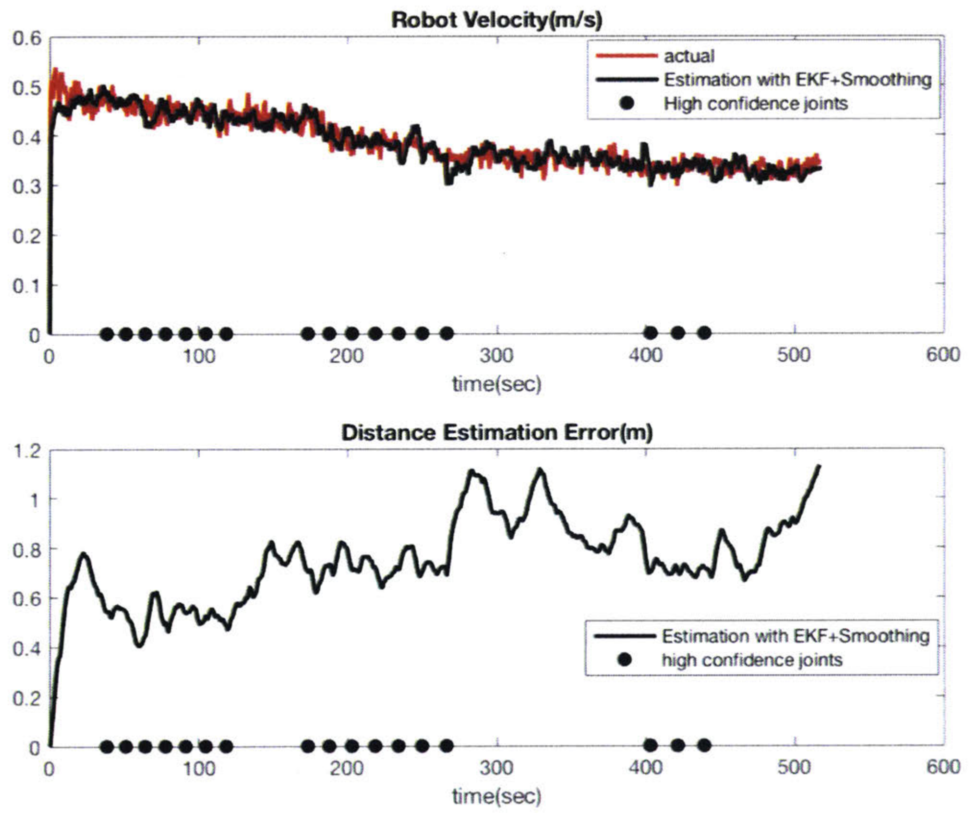
Figure 5-28: Revamped Step 4: Estimate robot speed outside High Confidence Zones with instantaneous speed measurements

Result And Discussion

The tactile speed sensor successfully limits the distance tracking error growth in the Low Confidence Zones. As shown in Fig. 5-29, this method tracks the speed variations throughout the pipeline equally well when compared to the consecutive pipe joint method. The distance tracking error is also on the order of 0.5 % of the total distance, but the distance error now fluctuates much less in the Low Confidence Zones than the results of the previous method. The distance tracking error in the Low Confidence Zones, although not bounded, now grows much slower than it does in the previous method. This is the benefit of populating the speed measurement space with more data points. Each additional data point adds a constraint to the robot's localization process. With more constraints in place, the robot increases its likelihood to accurately track its speed and distance.

This tactile speed sensor method and the previous consecutive pipe joint method have their pros and cons. The consecutive pipe joint method is simpler, and it requires fewer sensors than the tactile speed sensor method. The consecutive pipe joint method works well when it can observe multiple High Confidence Zones, or consecutive, evenly distributed pipe joints, in the pipeline. Its error scales with the length of Low Confidence Zones, so it is not effective in a pipeline where there are few consecutive, evenly distributed pipe joints. This is the scenario in many aged pipes around the world where heavy tuberculation on the pipe wall makes the pipe joints unobservable. In contrast, the tactile speed sensor method works in all kinds of pipes.

Figure 5-29: Localization result of the tactile speed sensor method



It is more effective in pipes where there are more obtrusions such as those from the tuberculation. The tactile sensor method requires slightly more complex hardware, but it is versatile and well suited for applications in real water systems.

5.9 Conclusion and Future Work

In this chapter, I presented two approaches to localize the robot in a pipeline without remote sensing. These approaches only require a minimum number of on-board sensors, and they are a IMU and the tactile leak sensors. Although the robot has no means to measure speed or distance directly, it utilizes the common knowledge about a water pipeline to constraint its motion estimation and produces accurate results. In the first localization method, the robot uses the tactile sensors to identify consecutive pipe joints among all obtrusions in the pipe. From the data history of pipe joints it derives its speed and distance with confidence. This method is capable of locating the in-pipe robot with a small error equal to 0.5 % of the total distance. However, its performance is worse in heavily tuberculated pipes where pipe joints are difficult to observe. Thus a second method is developed to measure robot speed directly with tactile sensors. Both methods are well suited for this soft leak detection robot. They keep the total cost of the robot low while providing high quality localization results.

These localization methods can be further improved in three ways. First, the algorithms can be improved to account for pipe depth variations. The methods I presented consider pipes in a 2D plane only. In reality, pipes can go up and downs even they are underground. The second improvement would be the sensing of pipe alignment. So far it is assumed that each pipe segment is straight. It is also assumed that the robot can accurately sense its direction and thus the direction of the pipe with the compass in the IMU. The accuracy of this direction sensing needs to be evaluated. If it is accurate enough, the robot will be able to measure the bending in the pipe segment and misalignment in the pipe joints. These information may be early indicators of pipe leaks. Last but not the least, these methods can be further validated in experiments and field tests. Given the limited field test opportunities, it was necessary to conduct much of the research in simulations. With more experimentations and even field tests, new observations may be made on how to improve the accuracy of these methods even further.

Chapter 6

Conclusion and Recommendations

6.1 Conclusions

In this thesis, low cost, soft robots for leak detection in real and active water pipes are presented. It can be deployed into real underground water pipes through any Tee junctions, flow passively through the pipeline, go around pipe bends and pinpoint leaks. It can do all these when the water service is on. There are four research contributions, all of which are validated in both simulations, and lab and field experiments:
(1) A soft and passive robot design that is more reliable and less complex than the state-of-the-art. It is the first soft body robot that went through real underground water pipes in the world.
(2) A unique method to detect leaks. It uses a soft membrane sensor to measure the suction force occurred at the leak locations due to the out flux of water flow. I made this method work in operating water pipes, which was never done before.
(3) A design that allows the membrane sensor to differentiate leaks form false positives. Through smart utilization of the difference in material properties, these sensors, made of a composite of ordinary fabrics and rubber, can measure leaks and in-pipe obtrusions differently at the same time.
(4) A practical, minimalism approach to determine the location of the soft robots inside pipes. It is a model based algorithm that best utilizes common knowledge about pipelines. It enables the robots to perform localization using a minimal amount of

on-board sensors and zero remote sensors.

The best attribute of my leak detection robots is that they are low cost but effective. They are fabricated with ordinary material with ordinary tools. They find leaks in real water pipes. It is a sub-500-dollar solution to a multi-billion-dollar water and infrastructure problem.

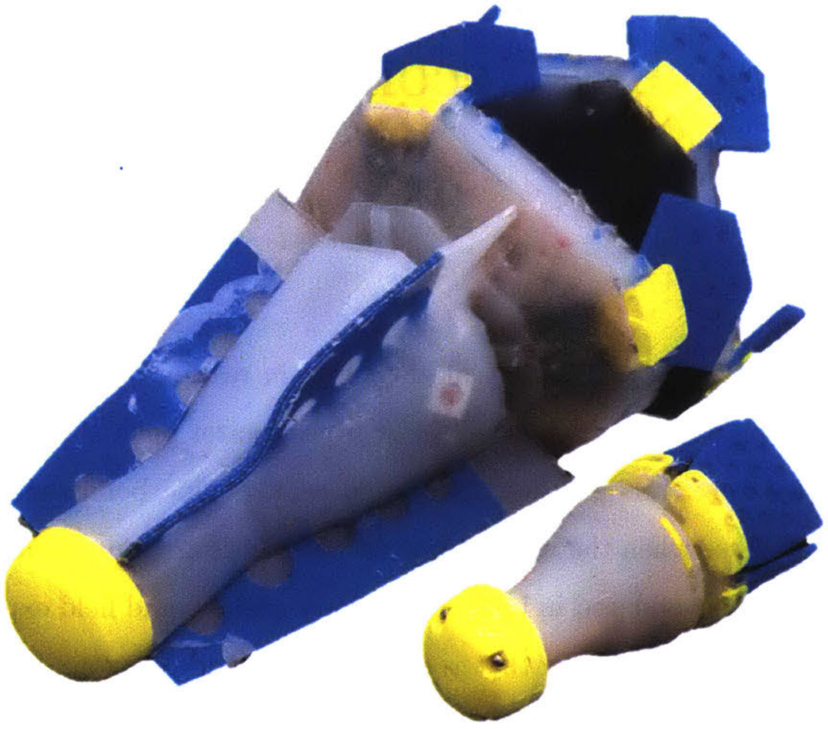


Figure 6-1: Two of my soft leak detection robots: 6" Lighthouse and 2" Daisy

6.2 Recommendations

The thesis lays the foundation of efficient in-pipe inspection with soft robots and sensors, and there are at least three areas that future innovations can be build upon. The first opportunity is to make the robot active rather than passive. The current design allows the robot to flow with the water stream in the pipe. It is an effective way to travel through a long distance pipelines with minimum energy consumption. However, it is ineffective when the robot is navigating through a pipe network, where it is required to make turns at certain Tee junctions independent of the flow. Most pipes in urban areas are in network configuration. Enabling the robot to actively steer its direction at Tee junctions will make the robot easier to use in urban areas.

The second opportunity is to build and validate the localization algorithm in 3D. The current localization algorithms assumes the pipes are in 2D plane and do not yet account for pipe depth variations. In reality, pipes can go up and downs even they are underground. It is also assumed in the current algorithm that the robot can accurately sense its direction and thus the direction of the pipe with the compass in the IMU. The accuracy of this direction sensing needs to be evaluated. If it is accurate enough, the robot will be able to measure the bending in the pipe segment and misalignment in the pipe joints. These information may be early indicators of pipe leaks. Moreover, these algorithms can be further validated in experiments and field tests. Given the limited field test opportunities, it was necessary to conduct much of the research in simulations. With more experimentations and even field tests, new observations may be made on how to improve the accuracy of these methods even further.

The third opportunity is to explore more applications of the soft bending angle sensor. In this thesis, the soft bending angle sensor has been demonstrated as an effective tool for differentiating leaks from false alarms. It has also been shown as a minimum approach to add sensing and feedback to any soft robot fish. There should be more applications where this kind of low-cost, thin, multi-purpose soft sensors are more advantageous than existing solutions. This can be a very exciting topic for future research.

Appendix A

Low Cost Robot Fabrication Process

A.1 2" Robot Cast Molding

Given the small size of the robot, it can use 3D printed molds. At this size, the 3D printed molds takes about 4 hours total to be printed on a Stratasys 250mc printer.

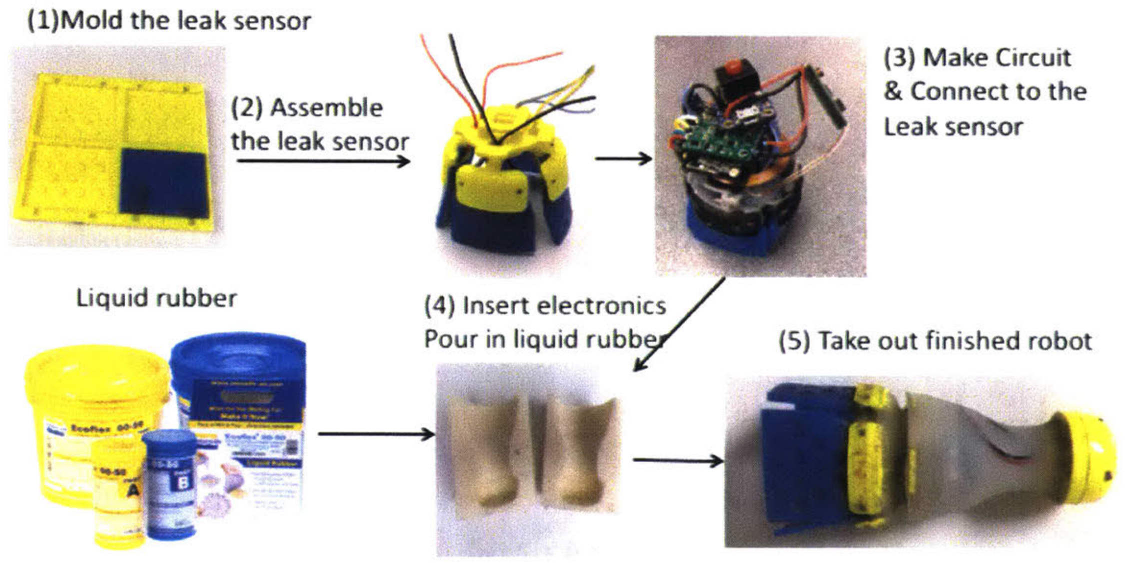

(1)Mold the leak sensor
(2) Assemble the leak sensor
(3) Make Circuit & Connect to the Leak sensor
Liquid rubber
(4) Insert electronics Pour in liquid rubber
(5) Take out finished robot

Figure A-1: 2" robot fabrication process

A.2 4-6" Robot Cast Molding

At this size, 3D printed molds can be expensive. Instead, the mold is assembled from acrylic sheets. The acrylic sheets are laser cut into the precise dimensions, and put together like Lego pieces. This method saves a lot of cost in comparison to 3D printing the mold.

Please also note that step 1: assemble the leak sensor should be repeated twice. There are now two rings of the the leak sensors in the final robot. Each ring covers 180 degrees in the 360-degree pipe circumference. Each ring is compressible.

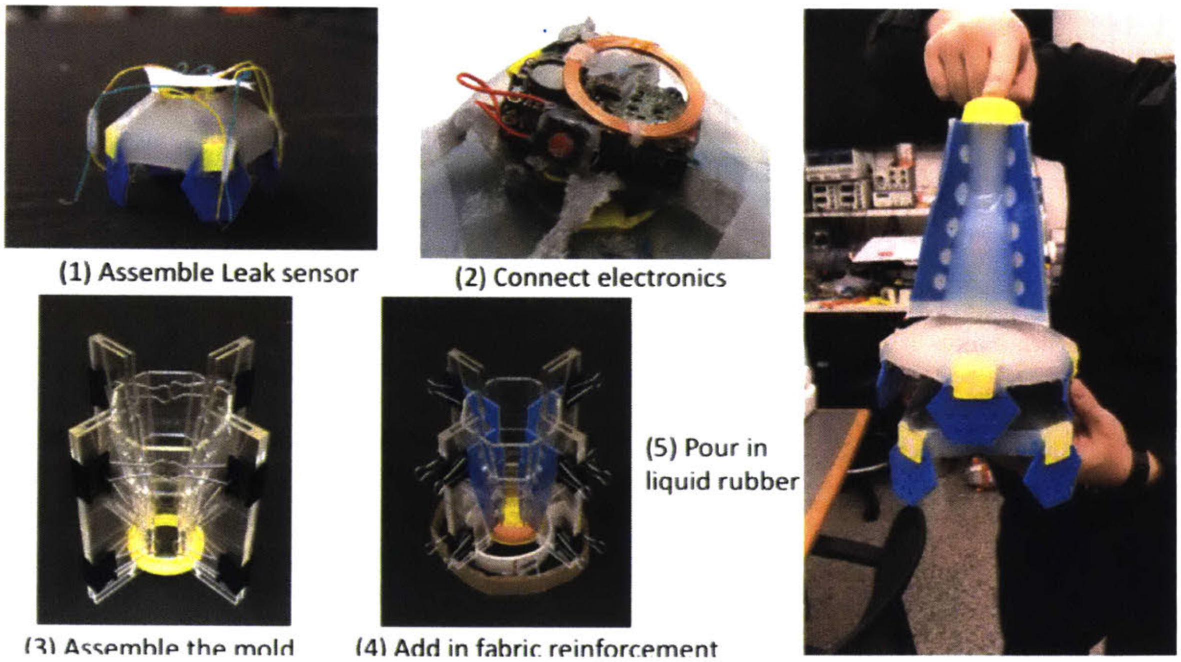


Figure A-2: 4-6" robot fabrication process

A.3 12" Robot Cast Molding

This is an alternative way to fabricate soft robots with sheet-assembled mold. Instead of Acrylic sheets, the mold can be assembled from foam core boards. This is very cheap, and it can also be used as a design iteration tool for the mold.

Although being made in super low cost molds, the outcome is still very good.

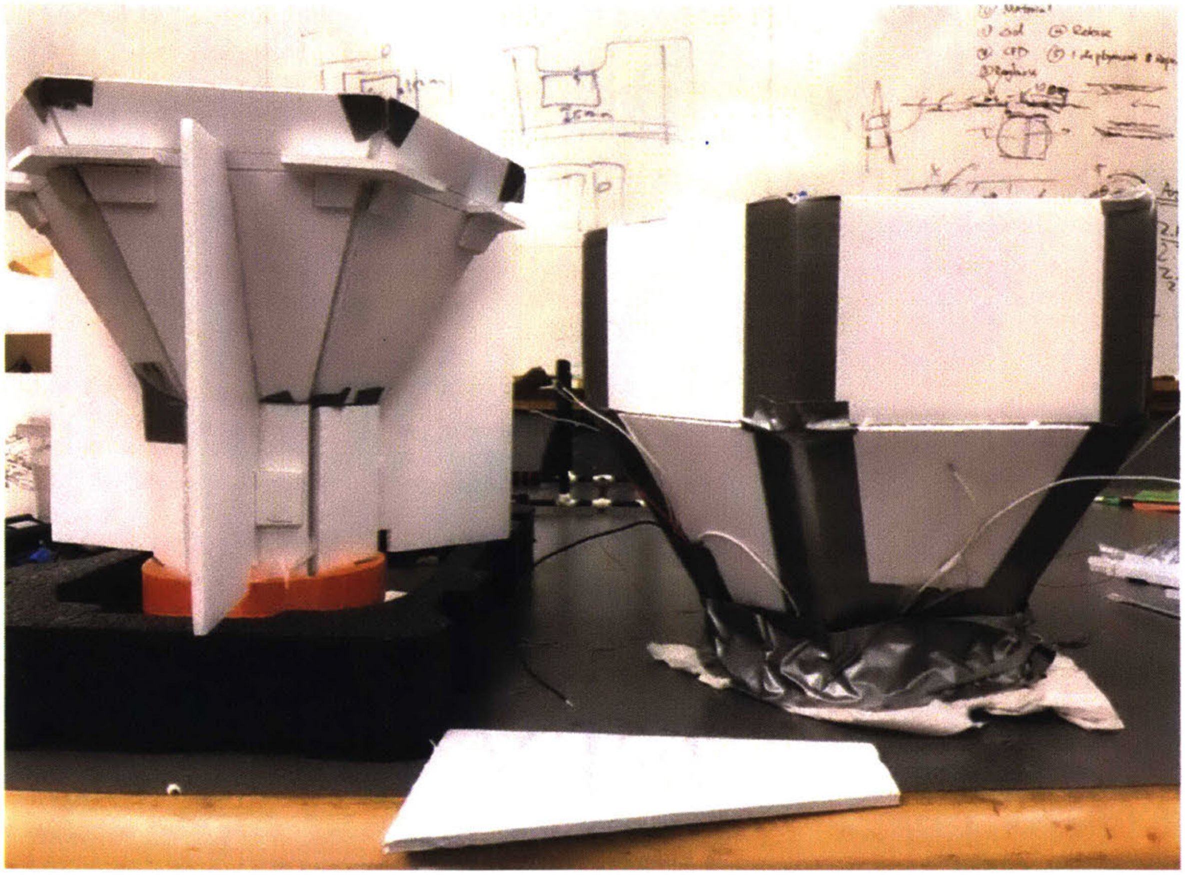


Figure A-3: 12" robot mold made from foam core boards

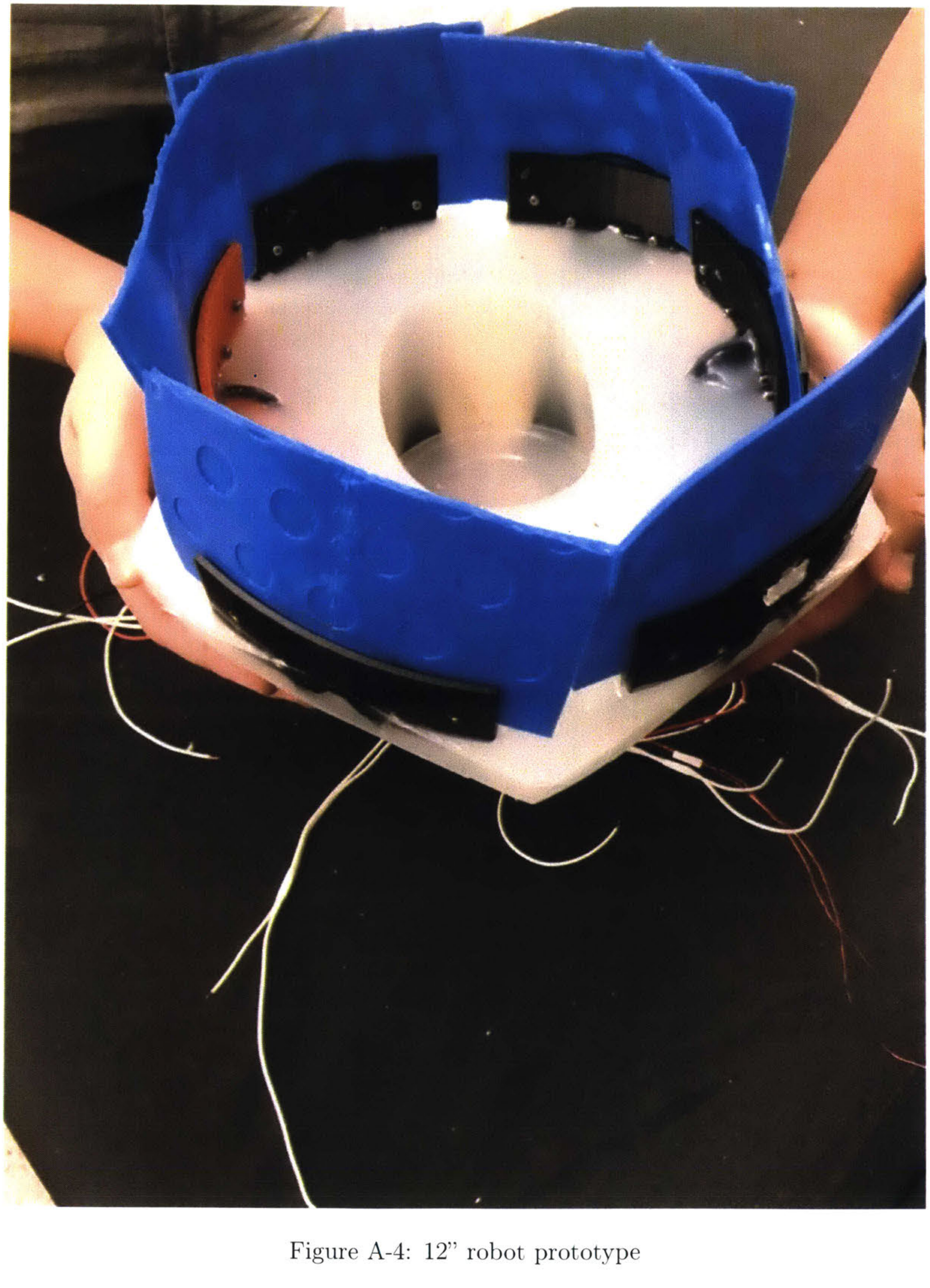


Figure A-4: 12" robot prototype

Appendix B

Field Test Report, Differentiate Leaks from Obtrusions Through Data Correlation

MIT,KFUPM & PipeTech Collaboration
Robotic Leak Detection Report #2

You Wu (youwu@mit.edu)

March 30, 2017

Abstract

This document reports the results of the field test of the Leak Detection Robot at Pipetech's facility during Jan 16-19, 2017. During the field test, an improved version of the leak detection robot successfully detected a 4mm diameter leak. Data from both the leak sensor and the motion sensor were collected for pipe features other than leaks, such as pipe elbows and joints. Signal analysis were performed on the data to differentiate leaks from various disturbances. This field test marked the first time we were able to detect leaks with our robotic technology in a real water pipeline.

Background

MIT, KFUPM and PipeTech LLC in Saudi Arabia have been collaborating on the robotic leak detection project since 2015. The goal was to take the MIT-KFUPM leak detection technology from a lab prototype stage to validation in real water pipe systems. The particular real water pipe system is provided by PipeTech LLC in their test facility in Dammam, Saudi Arabia, and it is a 52mm(2in) inner diameter, 1.5km(0.9mile) long cast iron pipeline with multiple 90 degree bends. The robot is supposed to detect leaks in this pipeline.

This report discusses the second test we conducted during January 16-19, 2017. In this second test, we achieved significantly better results than we did in the first test. In the first test in March 2016, we demonstrated a soft body robot that could reliably get through the water pipeline while been propelled by water flow. We also validated the method of inserting the robot into the pipe and retrieving it. However, the leak sensor was unable to register the expected leak signal. The MIT, KFUPM and Pipetech team investigated this problem and determined that both design issues and the construct of the artificial leak caused the problem. Thus before the second test, the MIT team redesigned the leak sensor and PipeTech constructed new artificial leak points. The goal of the second test was to validate the capability of the new leak sensor in finding leaks and it was achieved.

The Leak Detection Robot

The new leak detection robot (Figure 2) has the similar soft robot body as the previous one(Figure 1). Similar to a pipeline inspection gauge(PIG), this robot had no actuation, and it was propelled by the water flow in the pipes. Unlike regular PIGs or any other kind of pipeline robot platforms, this robot was soft and it was made of soft silicone rubber Ecoflex 0050 from Smooth-on Inc. The soft body could be bent, allowing the robot to turn around pipe elbows with ease. Embedded in the soft body of the robot, a 3.7V Lithium Ion battery powered an Arduino Mini Pro 3.3V/8MHz micro-controller, a Pololu 9 DoF Inertia Measurement Unit (IMU), and a Pololu SD card.

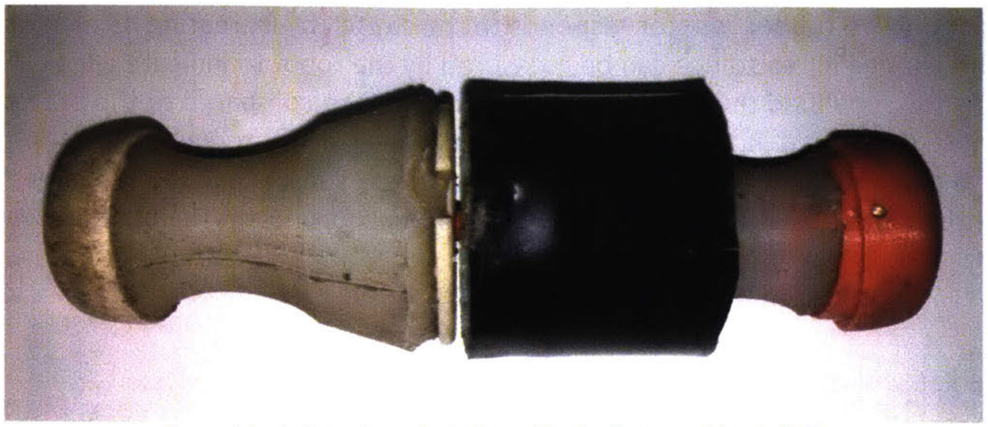


Figure 1 Leak detection robot v1 used in the first test, March 2016

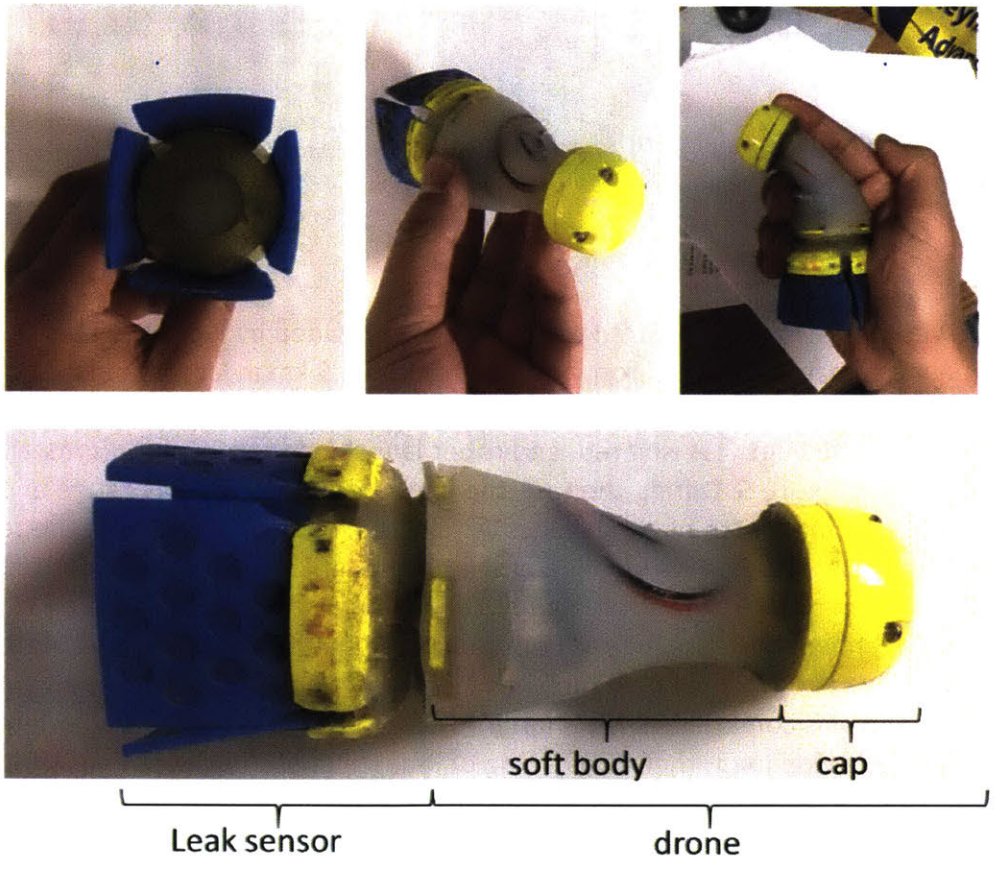


Figure 2 Leak detection robot v2 used in the second test, Jan 2017. Top left: rear view of the leak sensor; top mid: isometric view of the robot; top right: illustration of how the robot bend; bottom: the robot and its components

The new leak detection robot is equipped with a new leak detection sensor. This sensor consists of four pieces of identical blue silicone membranes connected to a yellow plastic support structure. As shown in Figure 3, those membranes have embedded conductive rubber resistors which increase their electrical resistance when stretched. The support structure is spring loaded like an umbrella; when water is pushing the robot in the back, the support structure will be fully expanded. Thus it can maintain the membranes at its end close to the pipe wall. When the leak sensor passes a leak, the membrane will be pulled by the suction force at the leak and thus stretch. The conductive rubber resistor responds to the stretch by increasing its electrical resistance. This change in resistance can be measured by the robot to indicate a leak. This design significantly simplified the leak sensor mechanism when compared to the previous versions we have developed. It is less affected by the hydrodynamic forces, such as disturbances and added mass effect, and this allows the leak sensor to respond faster to leaks. This new leak sensor has a calculated bandwidth of 130Hz, much faster than the previous version's 30Hz.

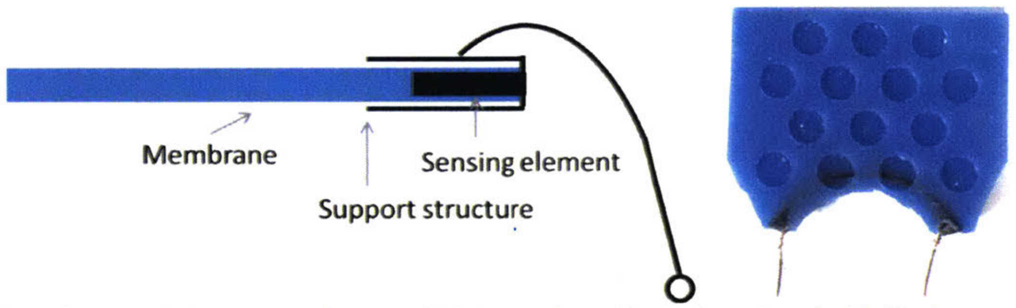


Figure 3 The leak sensor design. Sensing element, which is a conduct rubber resistor, is embedded in the membrane. The membrane is fixed inside the support structure at the two ends of the conductive rubber resistor.

The Field Test Result

The new robot was tested in the industrial water pipeline provided by PipeTech LLC in Saudi Arabia. The pipeline was made of cast iron, and it measured 1.5 km in length and 52mm in nominal inner diameter. The entire pipeline was in the horizontal plane. A section of 221 m long and 1.2km away from the inlet was isolated, and robot launch and retrieval tools installed on both ends, as shown in Figure 5. During the test, the inlet pressure was 2 Bar gauge and the flow rate was about 0.4 m/s. With pipe head loss considered, the line pressure at the test section was about 1.7 Bar gauge. This pipeline provided enough in-pipe features for testing. The first feature was a 4 mm pinhole leak on the pipe. There was a water tap wielded on top of it to turn it on and off, as shown in Figure 4(b). A bucket was used to collect the leaked water and measure the leak flow rate. The leak flow rate was measured to be about 5.6 L/min (1.48 Gal/min).This pipeline was constructed with hundreds of 6m long, 52mm inner diameter, metal pipe segments. At each pipe joint, there was a ring of 3mm diameter reduction, as shown in Figure 4(c). Thus each pipe joint, 6m apart, was an obstacle that could affect the leak sensor readings. Additionally, there were many pipe elbows. The pipeline had been in service for 6 years so there was a small amount of rust in the pipeline.

The test result showed that various in-pipe features could be differentiated on the measured signals. The robot travelled through the 221m long test section in an average of 550 seconds. The average speed was 0.4 m/s. The robot completed 2 tests, and in each test, it passed 1 leak, 4 pipe elbows and 41 pipe joints and recorded data for all of them. The readings from the leak sensor for these features were plotted in Figure 14. Since leaks were expected to show up as high frequency, impulse-like signals, only the changes in pulling forces were studied. The steady-state values of the leak sensor readings were removed with a high-pass filter (2Hz cutoff frequency).

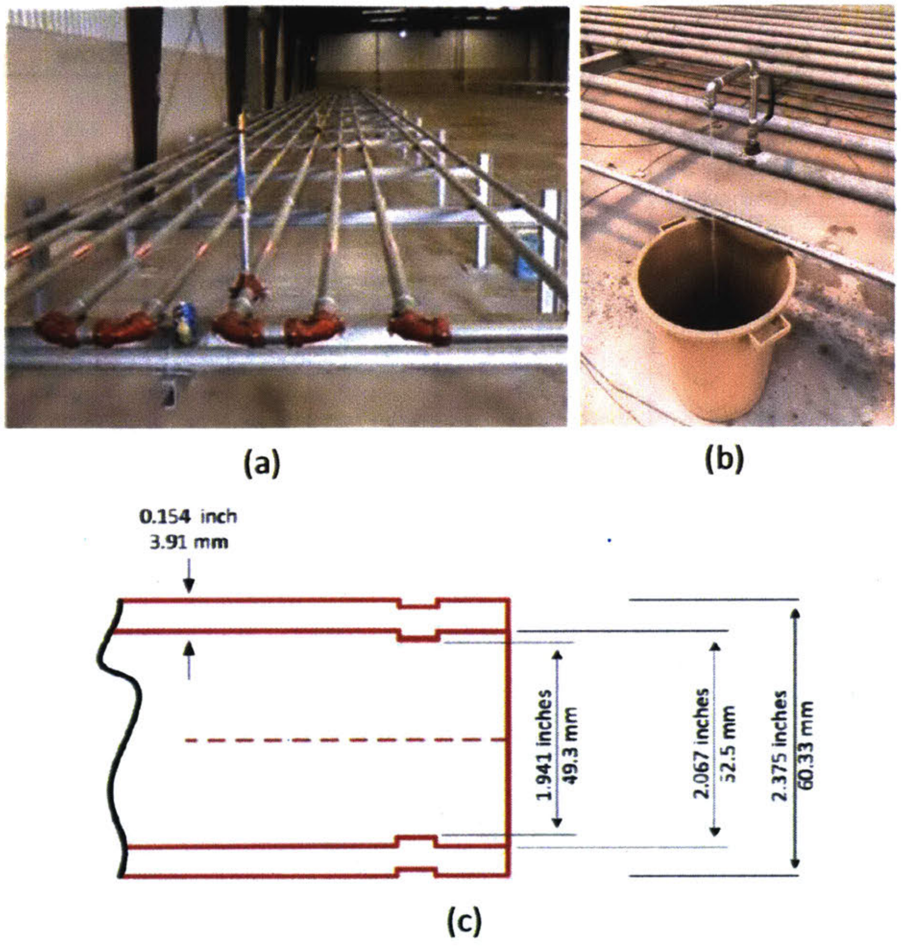


Figure 4 (a) The industrial pipeline used for testing (b) the leak on the pipe (c) Schematics of the pipe segments

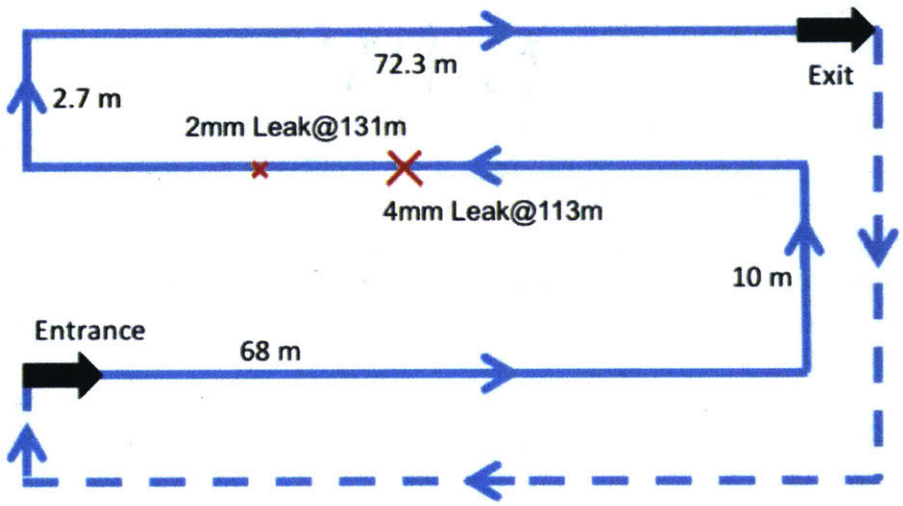


Figure 5 Schematics of the water pipeline for test

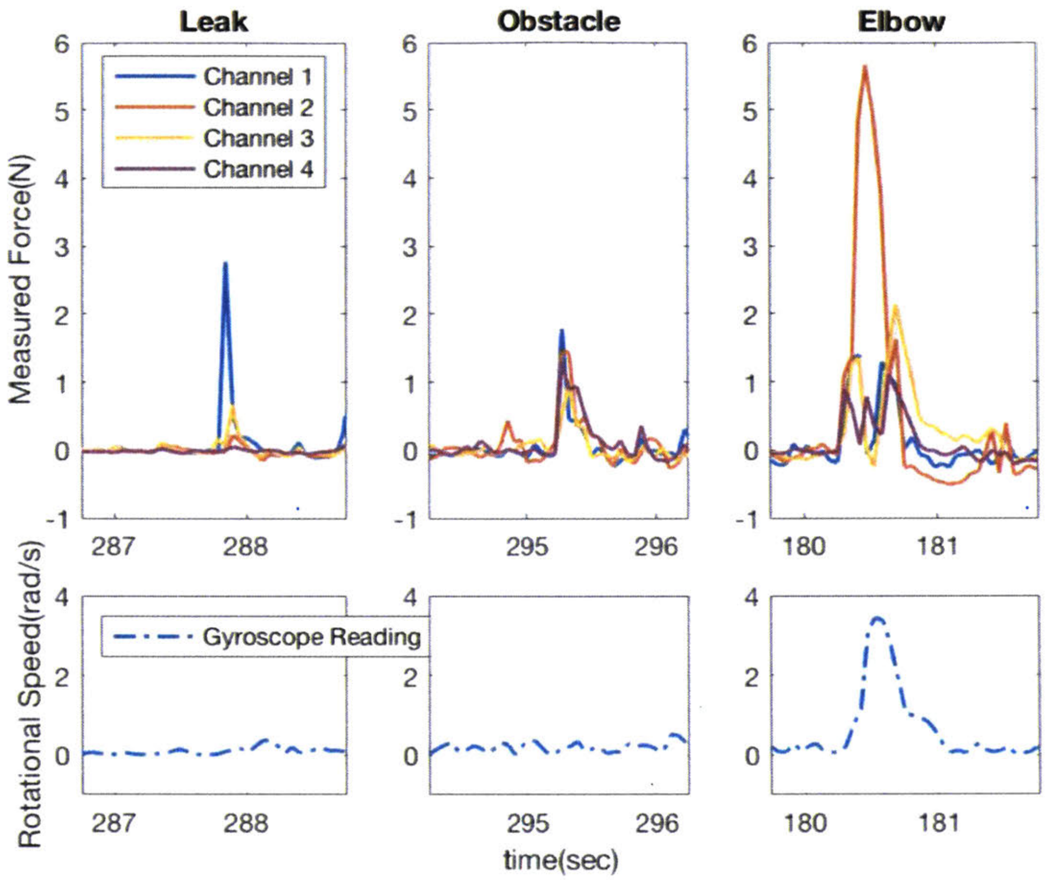


Figure 6 Leak sensor readings and gyroscope reading for rotational speed in the horizontal plane in three cases: 4mm pinhole leak, pipe joint as obstacle and pipe elbow.

At the leak, the leak sensor registered a high frequency, large magnitude change. This change was only in the channel corresponding to the membrane that was right on top of the leak, and the changes were much smaller for the other channels since their membranes didn't touch the leak. The frequency of the signal was 10Hz, which is at the aliasing limit for this 20Hz-sampling-rate robot.

At the obstacles on pipe joints, the leak sensor registered slower changes than it did at the leak. Since the diameter reduction at the joints was on all sides of the pipe as shown in Figure 4(c), all four channels detected changes. The signal at one of such obstacle was shown as the main peak in the middle plot of Figure 14.The dominant frequency was about 4Hz. The average magnitude was 1.2N and the standard deviation was 0.7N, however, its distribution was close to uniform distribution.

The robot can easily spot pipe elbows. The gyroscope part of the on-board IMU measured the robot's rotational speed in the horizontal plane, as shown in the bottom row in Figure 6. The gyroscope plots here were low pass filtered (5Hz cut off frequency) for reduced noise. At the elbow, the robot registered a significant change in its rotational speed. At leaks or obstacles, it did not. The leak sensor measured multiple oscillations in all four channels at the elbow, while single pulses at leaks and obstacles. Those oscillations are of similar dominant frequency as that of the obstacle signal.

Based on the above analysis, there are two analytical methods to differentiate leaks from obstacles and pipe elbows. The first method is in frequency domain, such as high pass filter. Sensor response to leaks is of higher frequency than responses to obstacles. However, due to the low sampling rate of this robot (20Hz) used during the test, much frequency content was not captured. Thus the frequency domain method was not selected.

The second method is data fusion. There are four channels in the leak sensor, one for each quarter of the pipe cross-section. There is also the gyroscopic reading which indicates the rotational speed of the robot. As discussed above, leaks most likely trigger only one channel of the leak sensor, while pipe elbows and the joints most likely trigger more channels. Thus the skewness of the data from the four channels of the leak sensor at any single point of time can be used to differentiate leaks from obstacles. One way to measure the skewness is to look at the difference between the mean value of the four channel readings, and ¼ of the largest value of the four channel readings. This difference should be between 0 and 0.75 times the largest value of the four channel readings. A smaller difference indicates a larger skewness and thus a higher chance to be a leak. We denote this difference as skewness, S_k:

$$S_k(t) = |\mathrm{mean}(\mathrm{F1(t), F2(t), F3(t), F4(t)}) - \frac{\max(\mathrm{F1(t), F2(t), F3(t), F4(t)})}{4}|$$

And we define a Leak factor FX1:

$$FX1(t) = \frac{Max\big(F1(t), F2(t), F3(t), F4(t)\big)}{S_k(t) + C}$$

while C is a small constant in place to avoid the denominator being zero. In practice, C=0.05 is chosen.

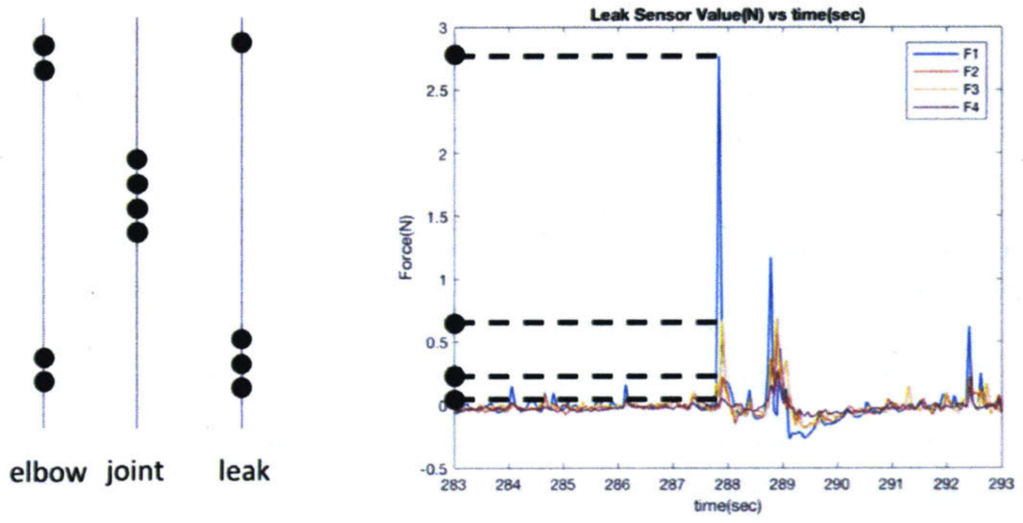


Figure 7 Data fusion method to categorize signals for leaks and obstacles. Ideally, S_k=0.25Max(F) for elbows, 0.75Max(F) for joints and 0 for leaks.

The result of the calculations is shown in Figure 8. The original leak sensor reading is plotted in Figure 8-top. On the horizontal axis is distance. It is estimated with average speed of the robot and time with error correction from information about the 90-degree bends. Arrivals at the 90-degree bends were monitored with the gyroscopic reading as shown in Figure 8-bottom. Figure 9-bottom indicates that the robot turned left twice and then turned right twice, which agrees with the pipe map shown in Figure 5. From Figure 5, we also know that the distances from the starting point to the bends are 68m, 78m, 146m and 148.7m. When a high pass filter (2Hz cutoff frequency) is applied to remove the steady state value of in the leak sensor reading and reveal only the changes, we have the second plot in Figure 8. The sensor readings after the high pass filter are used in the calculation of S_k and FX1. FX1 is plotted in the third part of Figure 8. As we can see, FX1 has high values at all four 90-degree bend locations. Those locations can be identified with the gyroscopic reading of the robot's rotational speed in the radial direction, as shown in Figure 8-bottom. In order to eliminate the false alarms at 90-degree bends, rotational speed (w(t)) should be considered. Thus FX1 was modified and became FX2:

$$FX2(t) = \frac{Max\big(F1(t), F2(t), F3(t), F4(t)\big)}{S_k(t) + |w(t)| + C}$$

By placing the rotational speed w(t) in the denominator, this new leak indication factor reduces when the rotational speed of the robot is high. That is the case when the robot pass a 90-degree bend.

FX2 is an effective leak indicator. It takes into account the location of the leak along the pipe cross-sectional circumference, the magnitude of the leak sensor reading, and the motion information of the robot. As shown in the fourth plot in Figure 8, FX2 was high at only one point in the entire test, of an estimated distance 112.3m from the starting point. The artificial pinhole leak of 4mm diameter was at 113m from the starting point. The leak detection robot found the leak with a distance estimation error of 0.7m. By using FX2, we were able to filter out most of the false alarms and clearly see the location of the leak.

Discussion

The main conclusion is that this field test conducted during Jan 16-19, 2017 was successful. It achieved our goal in validating the effectiveness of the MIT-KFUPM leak detection robot in finding leaks in real water pipes. A pinhole leak of 4mm diameter was detected at 1.7 Bar line gauge pressure and 0.4 m/s water flow speed. The estimated leak flow rate through the pinhole was 5.6 L/min. We also succeeded again in deploying the robot into the water pipe and retrieve it. Now we can confidently state that we have a working robot prototype for putting into real water pipes under operating conditions, detect leaks and later taking out of the pipe.

Several aspects of the robot and the experiment can be improved. The first aspect is the waterproofness of the robot. The robot actually ran through the pipeline 8 times, but it was only able to record complete data for the first two runs. Its electronics were water damaged for the remaining 6 runs and did not record complete data. The second aspect is about testing both the lower and upper limits of the robot's performance. In the test, there was another 2mm pinhole leak 18m behind the 4mm leak. However, the leak flow rate was under 0.2 L/min and the leak itself was blocked by the rust in the pipe early in the test. The robot was not able to register significant readings at this 2mm pinhole leak, partially due to the leak was blocked. More tests of the robot at lower pressure, higher pressure as well as different leak sizes would be beneficial for understanding the limits of this robot and leak sensor design.

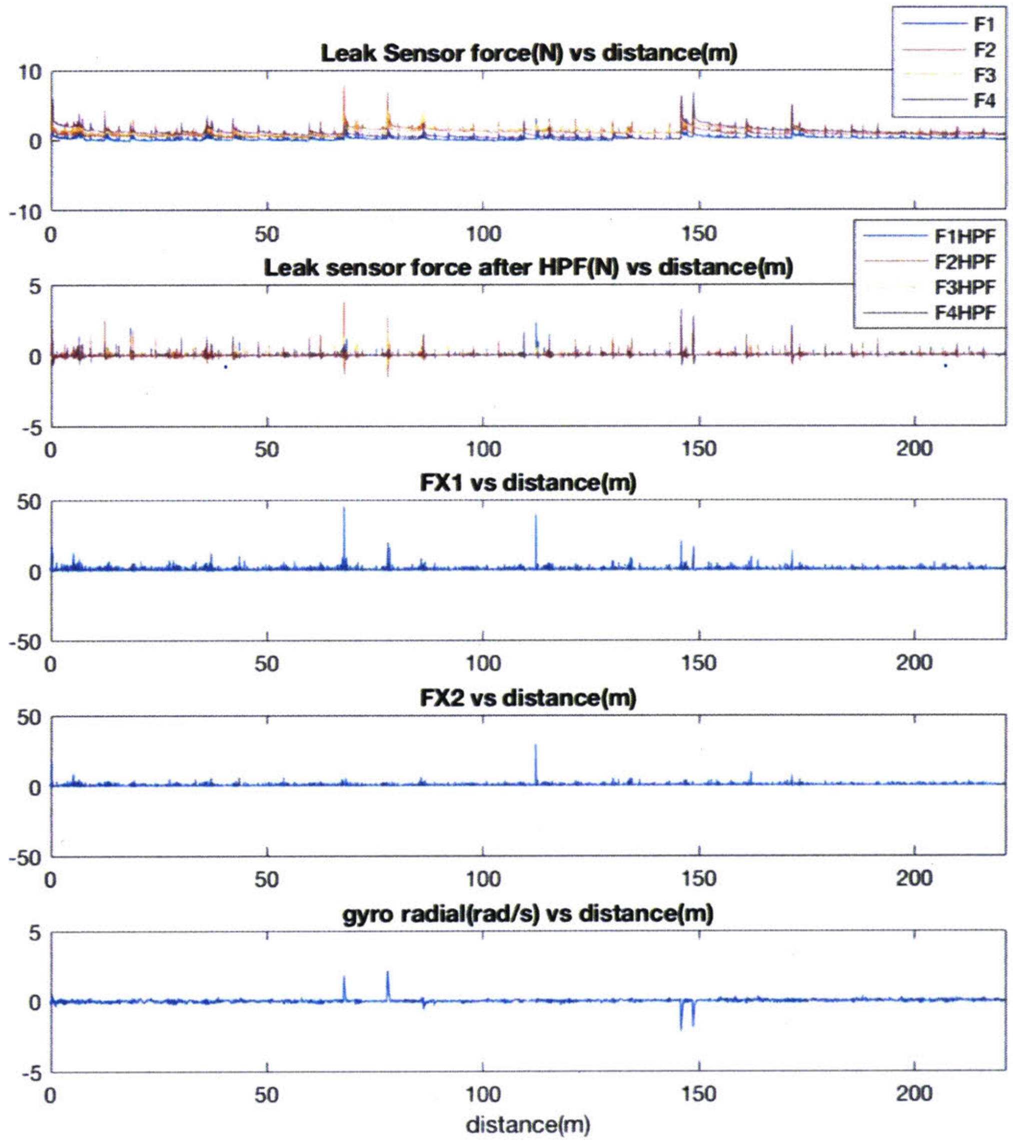


Figure 8 Leak Detection Result

Acknowledgement

The author of this document sincerely appreciates the support from MIT, KFUPM and PipeTech which made this project possible. Many people in the author's lab at MIT contributed to the remaking of the robot, including Solene Demay, Kristina Kim, Michael Finn-Henry and Prof. Kamal Youcef-Toumi. Dr. Rached Ben Mansour from KFUPM, Mr. Mowaffak Midani, Mr. Yousef Senan and Mr. Abbas Amrulla from PipeTech generously provided significant help to the author during the tests in Saudi Arabia.

Appendix C

Computer Program Codes

C.1 Arduino Code for the Leak Detection Robot

The robot has two modes:

(1) Data logger mode

Record data from leak sensors, IMU and store them to a SD card

(2) Data Transmission mode

Put all data files on a downloadable web page

The robot select the mode at initialization. If the robot is initialized inside the wireless charging dock, it will be in data transmission mode. Otherwise it will be in data logger mode.

```
*/
////Import libraries
//WIFI
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
//SD
#include <SPI.h>
#include <SD.h>
//I2C,IMU,ADC
#include <Wire.h>
#include <LIS3MDL.h>
#include <LSM6.h>
#include <Adafruit_ADS1015.h>

////wifi definitions
#define DBG_OUTPUT_PORT Serial
const char *ssid     = "YouHouse3";
const char *password = "ilovepurdue";
const char* host = "esp8266sd";

ESP8266WebServer server(80);

////SD definitions
#define SS 15
static bool hasSD = false;
#define FILE_BASE_NAME "/Dat"
char fileName[13] = FILE_BASE_NAME "00.csv";
long int lasttime = 0;

////IMU definitions
//#define SDA_pin 4 //D2
//#define SCL_pin 5 //D1
#define SDA_pin 5 //D1
#define SCL_pin 4 //D2
LSM6 imu;
LIS3MDL mag;
int agm[9]; //IMU outputs
int val[4]; //ADC outputs
////ADC definitions
Adafruit_ADS1015 ads1(0x48);     /* Use thi for the 12-bit version */
//Adafruit_ADS1015 ads2(0x49);
```

```
////switch mode definition 0
#define WatchPin 16
int modeKey=0;///1 --wifi transmit; 0---datalogger.
////indicator light
bool heartbeat=true;
long int lastbeattime = 0;

void setup(void){
  ////initialize debug output through serial
  DBG_OUTPUT_PORT.begin(115200);
  DBG_OUTPUT_PORT.setDebugOutput(true);
  DBG_OUTPUT_PORT.print("\n");
  pinMode(LED_BUILTIN, OUTPUT);

  ////initialize SD card and create a new file for datalogging
  if (SD.begin(SS)){
     DBG_OUTPUT_PORT.println("SD Card initialized.");
     hasSD = true;
     digitalWrite(LED_BUILTIN, LOW);
     delay(500);
     digitalWrite(LED_BUILTIN, HIGH);
     delay(500);
     digitalWrite(LED_BUILTIN, LOW);
     delay(500);
     digitalWrite(LED_BUILTIN, HIGH);
     delay(500);
     digitalWrite(LED_BUILTIN, LOW);
     delay(500);
  } else {
    DBG_OUTPUT_PORT.println("SD Card initialization error");
    return;
    }
  pinMode(WatchPin,INPUT);
  modeKey=digitalRead(WatchPin);
  DBG_OUTPUT_PORT.print("modeKey value is ");
  DBG_OUTPUT_PORT.println(modeKey);
  ////setup wifi connection and the website
  if (modeKey==1) {

    DBG_OUTPUT_PORT.println("transmission mode on");
    initializeWIFI();
    server.on("/", HTTP_GET, printDirectory);
    server.onNotFound(handleOthers);
    server.begin();
    DBG_OUTPUT_PORT.println("HTTP server started");
```

```
     digitalWrite(LED_BUILTIN, LOW);
    delay(250);
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
    delay(250);
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
    delay(250);
  }
  else {
    DBG_OUTPUT_PORT.println("datalogger mode on");
    ////start IMU
    initiateIMU();
    ADCinit();
    if (hasSD==true){
       findName();
       DBG_OUTPUT_PORT.print("New file will be named: ");
       DBG_OUTPUT_PORT.println(fileName);
    }
  }
}

void loop(void){
  if (modeKey==0){
    if ((millis() - lasttime) >= 20) {
      lasttime = millis();
      String dataString = "";
      dataString +=String(lasttime);
      ReadIMU();
      for (uint8_t i = 0; i < 9; i++) {
        dataString += ",";
        dataString += String(agm[i]);
      }
      ReadADC4();
      for (uint8_t j = 0; j < 4; j++) {
        dataString += ",";
        dataString += String(val[j]);
      }
      //store to SD card
      File file2write = SD.open(fileName, FILE_WRITE);
      if (file2write) {
        file2write.println(dataString);
        file2write.close();
        if (DBG_OUTPUT_PORT.available()){
          DBG_OUTPUT_PORT.println(dataString);
        }
      }
      /*else {
        DBG_OUTPUT_PORT.print("error opening ");
        DBG_OUTPUT_PORT.println(fileName);
      }
      */
    }
    if ((millis()-lastbeattime)>=2000 && !heartbeat) {
      heartbeat=true;
      digitalWrite(LED_BUILTIN, heartbeat);
    }
    if ((millis()-lastbeattime)>=2500 && heartbeat) {
      lastbeattime=millis();
      heartbeat=false;
      digitalWrite(LED_BUILTIN, heartbeat);
    }
  } else {
    server.handleClient();

  }
}

void ADCinit() {
  ads1.setGain(GAIN_ONE);        // 1x gain   +/- 4.096V  1 bit = 2mV      0.125mV
  //ads2.setGain(GAIN_ONE);        // 1x gain   +/- 4.096V  1 bit = 2mV      0.
125mV
  ads1.begin();
  //ads2.begin();
}

void ReadADC4() {
  val[0]= map(ads1.readADC_SingleEnded(0), -2048, 2048, -410, 410);
  val[1]= map(ads1.readADC_SingleEnded(1), -2048, 2048, -410, 410);
  val[2]= map(ads1.readADC_SingleEnded(2), -2048, 2048, -410, 410);
  val[3]= map(ads1.readADC_SingleEnded(3), -2048, 2048, -410, 410);
}

void initiateIMU() {
  Wire.begin(SDA_pin,SCL_pin);
  Wire.setClock(400000L);
  if (!mag.init())
  {
    DBG_OUTPUT_PORT.println("Failed to detect and initialize magnetometer!");
    while (1);
  }

  mag.enableDefault();

  //magnetic sensor
  // 0x58 = 0b01011000
    // OM = 10 (high-performance mode for X and Y); DO = 110 (40 Hz ODR)
    mag.writeReg(LIS3MDL::CTRL_REG1, 0x58);

    // 0x00 = 0b00000000
    // FS = 00 (+/- 4 gauss full scale)
    mag.writeReg(LIS3MDL::CTRL_REG2, 0x00);

    // 0x01 = 0b00000001
    // MD = 01 (continuous-conversion mode)
    mag.writeReg(LIS3MDL::CTRL_REG3, 0x00);

    // 0x0C = 0b00001000
    // OMZ = 10 (high-performance mode for Z)
    mag.writeReg(LIS3MDL::CTRL_REG4, 0x08);
  delay(1000);
  if (!imu.init())
  {
    DBG_OUTPUT_PORT.println("Failed to detect and initialize IMU!");
    while (1);
  }
  imu.enableDefault();

  // Accelerometer
// 0x4B = 0b01001011
// ODR = 0100 (104 Hz); FS_XL = 10 (+/-4 g full scale); BW_XL=11 (50Hz
antialiasing filter bandwidth)
  imu.writeReg(LSM6::CTRL1_XL, 0x4B);
// Gyro

    // 0x48 = 0b01001000
    // ODR = 0100 (104 Hz); FS_XL = 10 (1000 dps);
  imu.writeReg(LSM6::CTRL2_G,0x48);
// Common
    // 0x04 = 0b00000100
    // IF_INC = 1 (automatically increment register address)
    imu.writeReg(LSM6::CTRL3_C, 0x04);
    delay(1000);
```

```
}

void ReadIMU() {
  imu.read();
  mag.read();
  agm[0]=map(imu.a.x, -32768, 32768,-100, 100);
  agm[1]=map(imu.a.y, -32768, 32768,-100, 100);
  agm[2]=map(imu.a.z, -32768, 32768,-100, 100);
  agm[3]=map(imu.g.x, -32768, 32768,-100, 100);
  agm[4]=map(imu.g.y, -32768, 32768,-100, 100);
  agm[5]=map(imu.g.z, -32768, 32768,-100, 100);
  agm[6]=map(mag.m.x, -32768, 32768,-100, 100);
  agm[7]=map(mag.m.y, -32768, 32768,-100, 100);
  agm[8]=map(mag.m.z, -32768, 32768,-100, 100);
}

void findName() {;
  const uint8_t BASE_NAME_SIZE = sizeof(FILE_BASE_NAME) - 1;
  if (BASE_NAME_SIZE > 6) {
      DBG_OUTPUT_PORT.println("FILE_BASE_NAME too long");
    }
    while (SD.exists(fileName)) {
      if (fileName[BASE_NAME_SIZE + 1] != '9') {
        fileName[BASE_NAME_SIZE + 1]++;
      } else if (fileName[BASE_NAME_SIZE] != '9') {
        fileName[BASE_NAME_SIZE + 1] = '0';
        fileName[BASE_NAME_SIZE]++;
      } else {
        DBG_OUTPUT_PORT.println("Can't create file name");
      }
    }
}

void handleOthers(){
  String path=server.uri();
  String message;
  DBG_OUTPUT_PORT.print("URI: ");
  DBG_OUTPUT_PORT.println(server.uri());

  if(hasSD && loadFromSdCard(path)) {
    return;
  }
  else if (hasSD && path.substring(0,8) == "/remove/") {
    DBG_OUTPUT_PORT.println("enter remove operation");
    char __dataFileName[sizeof(path.substring(8))+1];
    path.substring(8).toCharArray(__dataFileName, sizeof(__dataFileName)+1);
    DBG_OUTPUT_PORT.println(__dataFileName);
    if (SD.exists(__dataFileName)) {
      DBG_OUTPUT_PORT.println("file exist");
      message+="deleting ";
      message+=__dataFileName;
      DBG_OUTPUT_PORT.println(message);
      if(SD.remove(__dataFileName)){
        DBG_OUTPUT_PORT.println("delete completed ");
        message+=" completed";
      } else {
        DBG_OUTPUT_PORT.println("delete failed ");
        message+=" failed";
        }
    }
    else {
      message+="requested file to delete: ";
      message+=__dataFileName;
      message+=" is not found";
      DBG_OUTPUT_PORT.println(message);
    }
    server.send(404, "text/plain", message);
    return;
  }
  else {
    message = "SDCARD Not Detected\n\n";
    message += "URI: ";
    message += server.uri();
    message += "\nMethod: ";
    message += (server.method() == HTTP_GET)?"GET":"POST";
    message += "\nArguments: ";
    message += server.args();
    message += "\n";
    for (uint8_t i=0; i<server.args(); i++){
      message += " NAME:"+server.argName(i) + "\n VALUE:" + server.arg(i) + "\n";
    }
    server.send(404, "text/plain", message);
    DBG_OUTPUT_PORT.print(message);
    return;
  }
}


bool loadFromSdCard(String path){
  String dataType = "text/plain";
  if(path.endsWith(".src")) path = path.substring(0, path.lastIndexOf("."));
  else if(path.endsWith(".htm")) dataType = "text/html";
  else if(path.endsWith(".css")) dataType = "text/css";
  else if(path.endsWith(".js")) dataType = "application/javascript";
  else if(path.endsWith(".png")) dataType = "image/png";
  else if(path.endsWith(".gif")) dataType = "image/gif";
  else if(path.endsWith(".jpg")) dataType = "image/jpeg";
  else if(path.endsWith(".ico")) dataType = "image/x-icon";
  else if(path.endsWith(".xml")) dataType = "text/xml";
  else if(path.endsWith(".pdf")) dataType = "application/pdf";
  else if(path.endsWith(".zip")) dataType = "application/zip";

  File dataFile = SD.open(path.c_str());
  if(dataFile.isDirectory()){
    path += "/index.htm";
    dataType = "text/html";
    dataFile = SD.open(path.c_str());
  }

  if (!dataFile)
    return false;

  if (server.hasArg("download")) dataType = "application/octet-stream";

  if (server.streamFile(dataFile, dataType) != dataFile.size()) {
    DBG_OUTPUT_PORT.println("Sent less data than expected!");
  } else {DBG_OUTPUT_PORT.println("sent successfully");}

  dataFile.close();
  return true;
}

void initializeWIFI() {
    WiFi.begin(ssid, password);
  DBG_OUTPUT_PORT.print("Connecting to ");
  DBG_OUTPUT_PORT.println(ssid);

  // Wait for connection
  uint8_t i = 0;
  while (WiFi.status() != WL_CONNECTED && i++ < 20) {//wait 10 seconds
    delay(500);
  }
  if(i == 21){
    DBG_OUTPUT_PORT.print("Could not connect to");
    DBG_OUTPUT_PORT.println(ssid);
    while(1) delay(500);
  }
  DBG_OUTPUT_PORT.print("Connected! IP address: ");
  DBG_OUTPUT_PORT.println(WiFi.localIP());

  if (MDNS.begin(host)) {
    MDNS.addService("http", "tcp", 80);
    DBG_OUTPUT_PORT.println("MDNS responder started");
    DBG_OUTPUT_PORT.print("You can now connect to http://");
    DBG_OUTPUT_PORT.print(host);
    DBG_OUTPUT_PORT.println(".local");
  }
}


void returnOK() {
  server.send(200, "text/plain", "");
}

void returnFail(String msg) {
  server.send(500, "text/plain", msg + "\r\n");
}

void printDirectory() {
  DBG_OUTPUT_PORT.print("URI: ");
  DBG_OUTPUT_PORT.println(server.uri());
  String message="welcome to ESP8266 file display page <br>";
    File root;
    root = SD.open("/");
    if (!root) {
        message+=("Failed to open directory <br>");
        server.send(404, "text/plain", message);
        DBG_OUTPUT_PORT.println(message);
        return;
     }
     if (!root.isDirectory()) {
        message+=("Not a directory <br>");
        server.send(404, "text/plain", message);
        DBG_OUTPUT_PORT.println(message);
        return;
      }
    File file = root.openNextFile();
    while(file) {
      if (file.isDirectory()) {
          message+=("  DIR : <a href=\"");
```

```
            message+=(file.name());
            message+=("\">");
            message+=(file.name());
            message+=("</a> <br>");
          } else {
            message+=("  FILE: <a href=\"");
            message+=(file.name());
            message+=("\">");
            message+=(file.name());
            message+=("</a> SIZE: ");
            message+=(String(file.size()));
            message+=("<br>");
          }
          file = root.openNextFile();
      }
      file.close();
      server.send(200, "text/html", message);
  }
```

C.2 Arduino Code for the Soft Bending Angle Sensor Data Collection

This program measures the bending angle sensor outputs and print them to a computer communication port.

```
int s0 =0;
int s1 =0;
int s2 =0;
int s3 =0;
unsigned long lasttime = 0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
}

void loop() {
  if (millis()-lasttime>=20) {
    lasttime=millis();
    // put your main code here, to run repeatedly:
    s0 = analogRead(A0);
    s1 = analogRead(A1);
    s2 = analogRead(A2);
    s3 = analogRead(A3);
    Serial.print(lasttime);
    Serial.print(",");
    Serial.print(s0);
    Serial.print(",");
    Serial.print(s1);
    Serial.print(",");
    Serial.print(s2);
    Serial.print(",");
    Serial.println(s3);
  }
}
```

C.3 Matlab Code for In-pipe Robot Localization

C.3.1 Main Program

This program is the main body of the simulator. It calls all linked subprograms and run the entire simulation from the beginning to the end.

```
%%map parameters

clc;
clear all;

%%% important factors
map_error=5/100;   % 5% error, normal distribution
Tees=10;            %flow reduction due to customer usage--flow steady state drift
obstacles = 1;      %%could be noise to joints
V_fluctuation=1/100;%100% stdev---flow noise

%%pipe characteristics
ninetyBend = 1;
valve = 2;          %%noise to joints in tactile signal
lengthPipe = 200;
SegPipe=6;
joints=floor(lengthPipe/SegPipe);
lengthPipe_est=lengthPipe*(1+map_error*randn(1));
V_mean=0.5;   %average flow speed, unit: m/s
tactile_noise=0.2;  % 20percent noise
acc_wn=0.01;%m/s^2
cd=12;
%%sensor parameters
%sensor sampling rate
f_imu=50;%imu low pass filter cut off frequency=sampling frequency/2
dt_L=1/f_imu;
%tactile speed sensor standard deviation
V_tac_var=0.01;%V_mean*dt_L;%m/s=2inch/sec
V_tac_scale=1;
%KF setup
R_acc_est=acc_wn^2;%(m/s^2)^2
Q_flow_est=1^2;%m^2/s^5

%%%%generator%%%%
%%map creation
prep1_map_genV2;
%%flow and motion creation
prep2_flow_genV4;

%%%%ensor%%%%
prep3_observer_genV4;
%%%%estimator%%%%
%%benchmark
ns1_deadreckoning;
%%method 1: consecutive pipe joints
%step 1,2
ns_autocor_final;
%step 3,4
ns_HCZLCZ_final;
%%method 2
ns_tx2_final;
```

```
%%%%%%%Generator%%%%%%%
%map gen
close all

output = cell(ninetyBend+Tees+valve+obstacles+joints+1, 4);
outCount = 1;
leftT = floor(Tees*rand(1));
rightT = Tees-leftT;
leftBend=ceil(ninetyBend*rand(1));
rightBend=ninetyBend-leftBend;

%50 pts per meter (line space)
%generate matrix in x,y

%placement of features
leftBendloc = lengthPipe* rand(1, leftBend);
rightBendloc = lengthPipe* rand(1, rightBend);
leftTloc = lengthPipe*rand(1, leftT);
rightTloc = lengthPipe*rand(1, rightT);
valveloc = lengthPipe*rand(1, valve);
obstacleloc = lengthPipe*rand(1, obstacles);
%jointloc=SegPipe*rand(1)+SegPipe*[0:1:joints-1];
jointloc=[SegPipe*rand(1)];
while jointloc(end)<=lengthPipe
    if rand(1)>0.9
        jointloc=[jointloc,jointloc(end)+SegPipe*rand(1)];
    else
        jointloc=[jointloc,jointloc(end)+SegPipe];
    end
end
%jointloc=SegPipe*[0:1:joints-1];


jointX = zeros(1,joints);
jointY = zeros(1,joints);
jointcount = 1;
leftTX= zeros(1,leftT);
leftTY = zeros(1,leftT);
leftTcount = 1;
rightTX= zeros(1,rightT);
rightTY = zeros(1,rightT);
rightTcount = 1;
valveX= zeros(1,valve);
valveY = zeros(1,valve);
valvecount = 1;
obstacleX= zeros(1,obstacles);
obstacleY = zeros(1,obstacles);
obstaclecount = 1;
dirx = 1;
diry = 0;
curx = 0;
cury = 0;
j=0;
jstep = 0.02;
counter = 2;
x(1) = 0;
y(1)=0;
while j<lengthPipe
```

```
curx = curx+ dirx*jstep;
cury = cury+diry*jstep;
x(counter) = curx;
y(counter) = cury;

if sum(jointloc>j-jstep & jointloc<=j)
        output{outCount, 1} = curx;
        output{outCount, 2} = cury;
        output{outCount, 3} = 'joint';
        output{outCount, 4} = j;
        outCount = outCount+1;
        jointX(jointcount) = curx;
        jointY(jointcount) = cury;
        jointcount = jointcount+1;
end

if sum(leftTloc>j-jstep & leftTloc<=j)
        output{outCount, 1} = curx;
        output{outCount, 2} = cury;
        output{outCount, 3} = 'left T';
        output{outCount, 4} = j;
        outCount = outCount+1;
        leftTX(leftTcount) = curx;
        leftTY(leftTcount) = cury;
        leftTcount = leftTcount+1;
end

if sum(rightTloc>j-jstep & rightTloc<=j)
    output{outCount, 1} = curx;
    output{outCount, 2} = cury;
    output{outCount, 3} = 'right T';
    output{outCount, 4} = j;
    outCount = outCount+1;
    rightTX(rightTcount) = curx;
    rightTY(rightTcount) = cury;
    rightTcount = rightTcount+1;
end

if sum(valveloc>j-jstep & valveloc<=j)
    output{outCount, 1} = curx;
    output{outCount, 2} = cury;
    output{outCount, 3} = 'valve';
    output{outCount, 4} = j;
    outCount = outCount+1;
    valveX(valvecount) = curx;
    valveY(valvecount) = cury;
    valvecount = valvecount+1;
end

if sum(obstacleloc>j-jstep & obstacleloc<=j)
    output{outCount, 1} = curx;
    output{outCount, 2} = cury;
    output{outCount, 3} = 'obstacle';
    output{outCount, 4} = j;
    outCount = outCount+1;
    obstacleX(obstaclecount) = curx;
    obstacleY(obstaclecount) = cury;
    obstaclecount = obstaclecount+1;
    end

    if sum(leftBendloc>j-jstep & leftBendloc<=j)
        %turn left
        output{outCount, 1} = curx;
        output{outCount, 2} = cury;
        output{outCount, 3} = 'left bend';
        output{outCount, 4} = j;
        outCount=outCount+1;
        if(dirx==0)
            dirx = -diry;
            diry =0;
        else
            diry = dirx;
            dirx = 0;
        end
    end

    if sum(rightBendloc>j-jstep & rightBendloc<=j)
       output{outCount, 1} = curx;
       output{outCount, 2} = cury;
       output{outCount, 3} = 'right bend';
       output{outCount, 4} = j;
       outCount=outCount+1;
       if(dirx==0)
           dirx = diry;
           diry =0;
       else
           diry = -dirx;
           dirx = 0;
       end
    end

    j = j+jstep;
    counter = counter+1;
end
output{outCount, 1} = curx;
output{outCount, 2} = cury;
output{outCount, 3} = 'end';
output{outCount, 4} = j;

plot(x,y);
hold on
plot(jointX, jointY,'.');
plot(leftTX, leftTY,'*');
plot(rightTX, rightTY,'*');
plot(valveX, valveY,'o');
plot(obstacleX, obstacleY,'s');
legend('pipe','joint','leftT','rightT','valve','obstacle');
output
```

```
%%4 tactile channels

close all;
%%noise in friction, rotation are correlated
%flow gen->V(t,l)
frobot=500;  % motion similation sampling rate for the robot, unit: Hz
dt=1/frobot;
%V_mean=0.5;   %average flow speed, unit: m/s
T=lengthPipe/V_mean*3;      % Time for the robot to go through the pipeline, unit: seconds
t=0:dt:T;
min_feature_size=dt*V_mean; %m
%%%simulate flow fluctuation
%%V_fluctuation=2;%200%
flowpass=10;      %low pass filter cut off frequency 50Hz, flow fluctuation simulation
V_random=V_mean*(1+V_fluctuation*randn(1,length(t)+20/flowpass*frobot)); %flow speed
tau=2*pi()/flowpass;
a=dt/tau;
V_lpf=filter(a,[1 a-1], V_random);
V_filtered=V_lpf(20/flowpass*frobot+1:end);
%plot(t,V_filtered);

%%robot dynamics, 6inch diameter robot
d_robot=0.15; %diameter of the robot, unit: m
cd=2;
cf=0.01;
A_rc=(d_robot/2)^2*3.14; %cross sectional area
m=A_rc*d_robot*4/3*1000*2/3; %mass of the robot, unit: kg
ma=(d_robot/2)^3*3.14*2/3*1000; %added mass,
mtotal=m+ma;

friction_norm=(m*9.8)/5*cf;
imax=length(t);
V_flow=V_filtered;
Drag=zeros(1,imax);
%friction=zeros(1,imax);
dis_r=zeros(1,imax);
v_r=zeros(1,imax);
acc_r=zeros(1,imax);
rot_r=zeros(1,imax);

dis_r(1)=0;
v_r(1)=0;%V_flow(1);
acc_r(1)=0;
rot_r(1)=0;%degrees/sec

%%%flow disturbance definition
%obstacle half size
hs_obs=floor(0.02/V_mean*frobot);
fVobstacle=1.00;
fVbend=1.1;
fVtee=0.9;
ffmean_obstacle=10;
ffvar_obstacle=0.2;
ffmean_bend=3;
ffvar_bend=0.2;
ffvar_norm=0.1;
friction_nofilter=friction_norm*(1+ffvar_norm*randn(1,imax+20/flowpass*frobot));
friction=friction_nofilter(20/flowpass*frobot+1:end);
```

```
%%tactile sensor definitions
tactile=zeros(4,imax);
tac_mean=0;
tac_obs=-0.5;
tac_bend=-2;
tac_out=2;%outflux, leaks, servicelines
tac_nm=-0.001;%normal noise magnitude

% %%tactile speed measurement
% Vtac_var=0.05;%m/s=2inch/sec
% Vtac=zeros(2,ninetyBend+Tees+valve+obstacles+joints);

%flow reduction due to active service lines
FlowReduce=1;

dis_i=0;
i=1;
tac_count=0;
while dis_i<=lengthPipe
    i=i+1;
    dice=randn(1);
    if sum(jointloc>dis_r(i-1) & jointloc<=dis_r(i-1)+min_feature_size)
        V_flow(i)=V_filtered(i)*fVobstacle;
        rot_change=15/(2*hs_obs*dt)*(2+1*dice)*ones(1,hs_obs);
        rot_r(i:i+hs_obs-1)=rot_r(i:i+hs_obs-1)+rot_change;
        rot_r(i+hs_obs:i+hs_obs*2-1)=rot_r(i+hs_obs:i+hs_obs*2-1)-rot_change;
        friction(i:i+hs_obs*2-1)=friction(i:i+hs_obs*2-1)+friction_norm*ffmean_obstacle*↙
(1+ffvar_obstacle*dice)*ones(1,hs_obs*2);
        tactile(:,i:i+hs_obs*2-1)=tac_mean+tac_obs*(1+0.1*randn(4,1))*ones(1,hs_obs*2);
%           tac_count=tac_count+1;
%           Vtac(:,tac_count)=[i;v_r(i-1)+Vtac_var*dice];
%
    elseif sum(obstacleloc>dis_r(i-1) & obstacleloc<=dis_r(i-1)+min_feature_size)
        V_flow(i)=V_filtered(i)*fVobstacle;
        rot_change=10/(2*hs_obs*dt)*dice*ones(1,hs_obs);
        rot_r(i:i+hs_obs-1)=rot_r(i:i+hs_obs-1)+rot_change;
        rot_r(i+hs_obs:i+hs_obs*2-1)=rot_r(i+hs_obs:i+hs_obs*2-1)-rot_change;
        friction(i:i+hs_obs*2-1)=friction(i:i+hs_obs*2-1)+friction_norm*ffmean_obstacle*↙
(1+ffvar_obstacle*dice)*ones(1,hs_obs*2);
        tac_num=floor(1.33*rand(1,4));
        tactile(:,i:i+hs_obs*2-1)=tac_mean+tac_obs*(1+dice)*tac_num'*ones(1,hs_obs*2);
%           tac_count=tac_count+1;
%           Vtac(:,tac_count)=[i;v_r(i-1)+Vtac_var*dice];
%
    elseif sum(valveloc>dis_r(i-1) & valveloc<=dis_r(i-1)+min_feature_size)
        V_flow(i)=V_filtered(i)*fVobstacle;
        rot_change=10/(2*hs_obs*dt)*dice*ones(1,hs_obs);
        rot_r(i:i+hs_obs-1)=rot_r(i:i+hs_obs-1)+rot_change;
        rot_r(i+hs_obs:i+hs_obs*2-1)=rot_r(i+hs_obs:i+hs_obs*2-1)-rot_change;
        friction(i:i+hs_obs*2-1)=friction(i:i+hs_obs*2-1)+friction_norm*ffmean_obstacle*↙
(1+ffvar_obstacle*dice)*ones(1,hs_obs*2);
        tactile(:,i:i+hs_obs*2-1)=tac_mean+tac_obs*(1+0.5*randn(4,1))*ones(1,hs_obs*2);
%           tac_count=tac_count+1;
%           Vtac(:,tac_count)=[i;v_r(i-1)+Vtac_var*dice];
%
    elseif sum(leftBendloc>dis_r(i-1) & leftBendloc<=dis_r(i-1)+min_feature_size)
        V_flow(i)=V_filtered(i)*fVbend;
        rot_r(i:i+149)=rot_r(i:i+149)+90/(150*dt)*ones(1,150);
        friction(i:i+149)=friction(i:i+149)+friction_norm*ffmean_bend*(1+ffvar_bend*dice)↙
*ones(1,150);
        tactile(:,i:i+149)=tac_mean+tac_bend*([0.5+0.5*randn(1);0.5+0.5*randn(1);1+0.↙
5*randn(1);1+0.5*randn(1)])*ones(1,150);
%         tac_count=tac_count+1;
%         Vtac(:,tac_count)=[i;v_r(i-1)+Vtac_var*dice];
%
    elseif sum(rightBendloc>dis_r(i-1) & rightBendloc<=dis_r(i-1)+min_feature_size)
        V_flow(i)=V_filtered(i)*fVbend;
        rot_r(i:i+149)=rot_r(i:i+149)-90/(150*dt)*ones(1,150);
        friction(i:i+149)=friction(i:i+149)+friction_norm*ffmean_bend*(1+ffvar_bend*dice)↙
*ones(1,150);
        tactile(:,i:i+149)=tac_mean+tac_bend*([1+0.5*randn(1);1+0.5*randn(1);0.5+0.↙
5*randn(1);0.5+0.5*randn(1)])*ones(1,150);
%         tac_count=tac_count+1;
%         Vtac(:,tac_count)=[i;v_r(i-1)+Vtac_var*dice];

    elseif sum(rightTloc>dis_r(i-1) & rightTloc<=dis_r(i-1)+min_feature_size) || sum↙
(leftTloc>dis_r(i-1) & leftTloc<=dis_r(i-1)+min_feature_size)
        %if rand()>0.5
            FlowReduce=FlowReduce*0.97;
            %friction(i:i+hs_obs*2-1)=friction(i:i+hs_obs*2-1)*0.5;
            tactile(:,i:i+hs_obs*5-1)=tac_mean+tac_nm*dice*ones(4,hs_obs*5);
            tactile(ceil(4*rand(1)),i:i+hs_obs*5-1)=tac_mean+tac_out*(1+dice)*ones(1,↙
hs_obs*5);
%             tac_count=tac_count+1;
%             Vtac(:,tac_count)=[i;v_r(i-1)+Vtac_var*dice];
        %end


    else
        V_flow(i)=V_filtered(i);
        tactile(:,i)=tac_mean+tac_nm*dice*ones(4,1);
        %rot_r(i)=0;
        %friction(i)=friction_norm*(1+ffvar_norm*dice);
    end
    %Drag(i)=cd*1000*A_rc/2*(V_flow(i)-v_r(i-1))*abs(V_flow(i)-v_r(i-1));
    %%bernoulli's equation
    V_flow(i)=V_flow(i)*FlowReduce;
    Drag(i)=cd*1/2*1000*(V_flow(i)^2-v_r(i-1)^2)*A_rc;
    acc_r(i)=(Drag(i)-friction(i))/mtotal;
    v_r(i)=v_r(i-1)+acc_r(i)*dt;
    dis_r(i)=dis_r(i-1)+v_r(i)*dt;
    dis_i=dis_r(i);
end
imax=i;
t(imax+1:end)=[];
acc_r(imax+1:end)=[];
V_flow(imax+1:end)=[];
Drag(imax+1:end)=[];
friction(imax+1:end)=[];
v_r(imax+1:end)=[];
dis_r(imax+1:end)=[];
rot_r(imax+1:end)=[];
tactile(:,imax+1:end)=[];
```

```
x_r=zeros(1,imax);
y_r=zeros(1,imax);
yaw_r=zeros(1,imax);
yaw_r(1)=0;%degrees
x_r(1)=0;
y_r(1)=0;
for i=2:1:imax
    yaw_r(i)=yaw_r(i-1)+rot_r(i)*dt;
    x_r(i)=x_r(i-1)+v_r(i)*dt*cos(yaw_r(i)/180*pi());
    y_r(i)=y_r(i-1)+v_r(i)*dt*sin(yaw_r(i)/180*pi());
end
figure(1);
subplot(5,1,1);plot(t,acc_r);title('robot acceleration');
subplot(5,1,2);plot(t,V_flow,'b');
hold on
plot(t,v_r,'r');
title('robot velocity');
legend('Flow','Robot');
hold off
subplot(5,1,3);plot(t,Drag,'b');
hold on
plot(t,friction,'r');
title('Forces on the robot');
legend('Drag','Friction');
hold off
subplot(5,1,4);plot(t,rot_r,'b');
title('robot rotational input(deg/sec)');
hold off
subplot(5,1,5);plot(t,tactile);
title('tactile sensor input');

figure(2);
plot(x,y,'b-','LineWidth',3);
hold on
plot(x_r,y_r,'r-');
plot(jointX, jointY,'.','MarkerSize',12);
plot([leftTX, rightTX], [leftTY, rightTY],'x','MarkerSize',12);
plot(valveX, valveY,'o','MarkerSize',12);
plot(obstacleX, obstacleY,'s','MarkerSize',12);
legend('pipe','robot path','joint','Tee','valve','obstacle');
xlabel('x distance(m)');
ylabel('y distance(m)');
title('original pipe map');
hold off


% figure(3)
% plot(t,rot_r,'.-');
% title('robot rotational speed over time');

% figure(4)
% plot(t,yaw_r);
% title('robot yaw');

%figure(5)
%plot(t,friction);
```

```
%%%Sensor%%%%%%%
%%robot localization with IMU and tactile
%generate accelerometer and gyro data
%%4 tactile sensors
close all;

%%imu(50hz) acts as a low pass filter
tau=2*pi()/f_imu;
a=dt/tau;
%acc
acc_f=filter(a,[1 a-1],acc_r);
%gyro
windowSize=1*floor(frobot/f_imu);
b = (1/windowSize)*ones(1,windowSize);
gyro_r=filter(b,1,rot_r);
%gyro_f=filter(a,[1 a-1],gyro_r);
gyro_f=filter(a,[1 a-1],rot_r);
%%encoder speed measure
V_enc=filter(a,[1 a-1],v_r);

f_tac=10;
tau=2*pi()/f_tac;
a=dt/tau;
%tactile_f=filter(a,[1 a-1],tactile);
windowSize=2*floor(frobot/f_imu);
b = 1/windowSize*ones(1,windowSize);
%tactile_ff=zeros(4,length(t));
tactile_f=zeros(4,length(t));
for i=1:1:4
    tactile_ff=conv(b,tactile(i,:));
    tactile_ff(1:floor(length(b)/2-1))=[];
    tactile_ff(length(tactile)+1:end)=[];
    tactile_f(i,:)=filter(a,[1 a-1],tactile_ff);
    %tactile_f1(i,:)=filter(b,1,tactile_ff(i,:));
end
%downsample to imu frequency
t_L=downsample(t,frobot/f_imu);
acc_L=downsample(acc_f,frobot/f_imu);
gyro_L=downsample(gyro_f,frobot/f_imu);
tactile_L=zeros(4,length(t_L));
tactile_L(1,:)=downsample(tactile_f(1,:),frobot/f_imu);
tactile_L(2,:)=downsample(tactile_f(2,:),frobot/f_imu);
tactile_L(3,:)=downsample(tactile_f(3,:),frobot/f_imu);
tactile_L(4,:)=downsample(tactile_f(4,:),frobot/f_imu);
dis_r_L=downsample(dis_r,frobot/f_imu);
V_enc_L=downsample(V_enc,frobot/f_imu);

%%add measurement noise
%acc_wn=0.1;%m/s^2
gyro_wn=2;%deg/sec
gyro_bias_rw=0.75;
enc_wn=0.05;

acc_output=acc_L+acc_wn*randn(1,length(t_L));

gyro_bias_dot=gyro_bias_rw*randn(1,imax);
gyro_bias=zeros(1,length(t_L));
```

```
V_enc=zeros(1,length(t_L));
 for i=2:1:length(t_L)
     gyro_bias(i)=gyro_bias(i-1)+gyro_bias_dot(i)*sqrt(1/f_imu);
     V_enc(i)=V_enc_L(i)+enc_wn*randn(1);
 end
gyro_output=gyro_L+gyro_wn*randn(1,length(t_L))+gyro_bias;

tactile_output=1000*tactile_L.*(1+tactile_noise*randn(1,length(t_L)));%sensor calibration↙
1000Ohm/N


%%generate tactile speed measurement
f_t_hpf=1;
tau=2*pi()/f_t_hpf;
a=dt_L/tau;
f_t_lpf=10;
tau2=2*pi()/f_t_lpf;
a2=dt_L/tau2;
tactile_HPF=zeros(4,length(tactile_output));
tactile_BPF=zeros(4,length(tactile_output));
for i=1:1:4
    tactile_HPF(i,:)=filter([1-a a-1],[1 a-1],tactile_output(i,:));
    tactile_BPF(i,:)=filter(a2,[1 a2-1],tactile_HPF(i,:));
end
tactile_abs_mean=1/4*ones(1,4)*abs(tactile_BPF);
%tactile_abs=1./(ones(1,4)*(gyro_output+(abs(tactile_HPF)-tactile_abs_mean).↙
/tactile_abs_mean).^2);
tactile_max=max(abs(tactile_BPF));
tactile_min=min(abs(tactile_BPF));
tactile_x=tactile_abs_mean.*(tactile_min./tactile_max+0.01);
%tactile_x=tactile_abs_mean./(tactile_max-tactile_abs_mean+0.1);
 tactile_threshold=0.2;
[tactile_pksx,tactile_locsx]=findpeaks(tactile_x,'MinPeakHeight',↙
tactile_threshold,'MinPeakDistance',5/dt_L);%'maxPeakWidth',1/dt_L);
[tactile_pks0,tactile_locs0]=findpeaks(tactile_max,'MinPeakHeight',↙
tactile_threshold,'MinPeakDistance',2/dt_L);%'maxPeakWidth',1/dt_L);
tactile_abs=tactile_x;
v_r_L=downsample(v_r,frobot/f_imu);
V_tac_L=v_r_L(tactile_locs0);
V_tac_noise=V_tac_var*randn(1,length(tactile_locs0));
V_tac_est=V_tac_scale*abs(V_tac_L+V_tac_noise);



figure(201);
subplot(5,1,1);
plot(t_L,acc_output,'b-');
hold on;
plot(t_L,acc_L,'r-');
legend('accelerometer output','real acceleration');
title('accelerometer output');
hold off;

subplot(5,1,2);
%%plot(t,gyro_r,'b-');
hold on;
%plot(t,gyro_r,'b-');
plot(t_L,gyro_output,'r-');
```

```
title('gyroscope output');
%legend('real yaw speed','gyroscope output');
hold off;

subplot(5,1,3);
%plot(t_L,tactile_BPF);
%plot(t_L,tactile_abs_mean,'r-');
hold on
plot(t_L,tactile_x,'b.');
%plot(t_L,tactile_BPF);
plot(t_L(tactile_locsx),tactile_pksx,'ro');
title('robot tactile output');
hold off

subplot(5,1,4);
%plot(t_L,tactile_BPF);
%plot(t_L,tactile_abs_mean,'r-');
plot(t_L,abs(tactile_BPF));
hold on
plot(t_L(tactile_locs0),tactile_pks0,'ro');
title('robot tactile output');
hold off

subplot(5,1,5);plot(t_L,v_r_L,'b-');
hold on
plot(t_L(tactile_locs0),V_tac_est,'ro');
title('robot speed');
legend('real speed','tactile speed measure');
hold off
```

```
f_acc_f=10;
tau_acc_f=2*pi()/f_acc_f;
dt_L=1/f_imu;
a_acc_f=dt_L/tau_acc_f;

acc_bpf=filter(a_acc_f,[1 a_acc_f-1],acc_output);

V_0=0;
V_flow_est=lengthPipe_est/t(end);

V_est_dd=zeros(1,length(acc_output));
V_est_dd(1)=V_0;
for i=2:1:length(acc_output)
    V_est_dd(i)=V_est_dd(i-1)+acc_bpf(i-1)*dt_L;
end

d_accbpf0=zeros(1,length(V_est_dd));
for i=2:1:length(V_est_dd)
    d_accbpf0(i)=d_accbpf0(i-1)+(V_est_dd(i)+V_est_dd(i-1))/2*dt_L;
end

dis_dd=d_accbpf0;


Z_obsv=acc_output;
coef=cd*1000*A_rc/(m+ma);

Q_VfVf=(0.1/10/f_imu)^2;
Q_VrVf=Q_VfVf;
Q_VrVr=0;
Q_aa=0.2^2;
Q_Vra=0;

Q_ss=Q_VrVr*dt_L^2;
Q_process=1*[Q_ss 0 0 0 ;
     0 Q_VfVf Q_VrVf 0;
     0 Q_VrVf Q_VfVf Q_Vra;
     0 0 Q_Vra Q_aa];





H = [0 0 0 1];

K = zeros(4,1,length(acc_output));

X_aposteriori=zeros(4,length(acc_output));

X_apriori=zeros(4,length(acc_output));

P_apriori = zeros(4,4,length(acc_output));
```

```
P_aposteriori = zeros(4,4,length(acc_output));
V_flow_est=lengthPipe*(1+map_error)/max(t_L);
V_0=0;
a_0=cd*1000*A_rc*(0-V_flow_est)^2/(m+ma);
X_aposteriori(:,1)=[0; V_flow_est; V_0; a_0];
X_apriori(:,1)=X_aposteriori(:,1);
P_aposteriori(:,:,1) = [0.0001^2 0 0 0;
    0 0.1^2 0 0;
    0 0 0.01^2 0;
    0 0 0 1^2*a_0^2];




for i=2:1:length(Z_obsv)
    Ai=[1   0     dt_L    dt_L^2;
        0   1     0       0;
        0   0     1       dt_L;
        0 coef*abs(X_aposteriori(2,i-1)-X_aposteriori(3,i-1)) -coef*abs(X_aposteriori(2,↙
i-1)-X_aposteriori(3,i-1)) 0];

    X_apriori(:,i)=Ai*X_aposteriori(:,i-1);




    Qi = Q_process;

    Ri= R_acc_est;


    P_apriori(:,:,i) = Ai*P_aposteriori(:,:,i-1)*Ai' + Qi';


    K(:,:,i) = P_apriori(:,:,i)*H' / (H*P_apriori(:,:,i)*H'+Ri);


    X_aposteriori(:,i) = X_apriori(:,i) + K(:,:,i) * (Z_obsv(:,i) - H*X_apriori(:,i));


    P_aposteriori(:,:,i) = (eye(4) - K(:,:,i)*H) * P_apriori(:,:,i);
end
a_ekfj=X_aposteriori(4,:);
a_ekfj_var=P_aposteriori(4,4,1:end);
a_ekfj_std=reshape(sqrt(a_ekfj_var),1,[]);

V_ekfj=X_aposteriori(3,:);
V_ekfj_var=P_aposteriori(3,3,1:end);
V_ekfj_std=reshape(sqrt(V_ekfj_var),1,[]);

Vf_ekfj=X_aposteriori(2,:);

d_ekfj=X_aposteriori(1,:);
d_ekfj_var=P_aposteriori(1,1,:);
d_ekfj_std=reshape(sqrt(d_ekfj_var),1,[]);
```

```
d_est_dd=zeros(1,length(V_est_dd));
for i=1:1:length(V_est_dd)-1
    d_est_dd(i+1)=d_est_dd(i)+(V_est_dd(i)+V_est_dd(i+1))/2*dt_L;
end
figure(101);
subplot(3,1,1);
plot(t_L,v_r_L,     ,            ,1);
hold
plot(t_L,V_est_dd,     ,            ,2);
title(                           );
xlabel(           );
ylabel(                            );
legend(         ,                                    );


hold
subplot(3,1,2);
plot(t_L,v_r_L,     ,            ,1);
hold
plot(t_L,V_ekfj,    ,            ,2);
title(                                   );
xlabel(           );
ylabel(               );
legend(         ,                                           );


hold

subplot(3,1,3);
hold
```

```
plot(t_L,abs(d_ekfj-dis_r_L),    ,           ,2);
title(                              );
xlabel(          );
ylabel(              );
legend(                                        );
ylim([0,10]);
hold
set(findall(gcf,          ,          ),          ,12);
```

```
%%find all possible joints
%%assume at most 1 false positive between joints
%%assume at most 1 missing joint in a row
%%assume no false positive and false negative are next to each other
close all;

f_g_lpf=1;
tau=2*pi()/f_g_lpf;
a=dt_L/tau;
gyro_HPF=filter([1-a a-1],[1 a-1],gyro_output);
gyro_radius_abs=filter(1/(0.5*f_imu)*ones(1,floor(0.5*f_imu)),1,sqrt(gyro_HPF.^2));
joint_threshold=5;
[joint_pks0,joint_locs0]=findpeaks(gyro_radius_abs,'MinPeakHeight',↙
joint_threshold,'MinPeakDistance',3/dt_L,'MinPeakProminence',5, 'MaxPeakWidth',1/dt_L);

%%eliminate bends
joint_pks1=zeros(1,length(joint_pks0));
joint_locs1=zeros(1,length(joint_pks0));
jj=0;
for j=1:1:length(joint_locs0)
    if joint_pks0(j)<=80
        if rand(1)>=0
        jj=jj+1;
        joint_pks1(jj)=joint_pks0(j);
        joint_locs1(jj)=joint_locs0(j);
        end
   end
end
joint_pks1(jj+1:end)=[];%%corrected array of max rotational speed at joints
joint_locs1(jj+1:end)=[];%%corrected array of time array indexes at joints

%%%autocorrelation to find norminal frequency change
fs=f_imu;
xj=joint_locs1;
yj=zeros(1,length(t_L));
dt_joint1=(xj(2:end)-xj(1:end-1))*dt_L;
yj(xj)=1;
%%convolution
cn=fs;
yjj=1/cn*conv(conv(ones(1,cn),ones(1,cn)),yj);
yjj(1:cn-1)=[];
yjj(length(yj)+1:end)=[];
[acor_full,lag_full] = xcorr(yjj,yjj);
halfway_full=(length(acor_full)-1)/2;
acor_full(halfway_full+1-cn*2:halfway_full+1+cn*2)=0;
lag_full(1:halfway_full+1)=[];
acor_full(1:halfway_full+1)=[];
[maxcor_full,mainlag_full]=max(acor_full);
acor_full=acor_full/maxcor_full;
[pksf,locsf]=findpeaks(acor_full,'MinPeakHeight',max(acor_full)*0.75,'MinPeakDistance',↙
fs*2);
mainlag_full=locsf(1);

mainlag=0;
norm_tj2j=zeros(2,length(xj)-1);
acorlast=[1];
for i=1:1:length(norm_tj2j)
    if i<=5
```

```
        i0=1;
        iL=max(xj(5)-i0,60*fs);
    else
         i0=max(min(xj(i-5)-fs,xj(i)-60*fs),1);
         iL=xj(i)-i0;
    end

    maxlag=min(max(mainlag*10,fs*60),iL);
    y_window=yjj(i0:min(i0+iL,length(yjj)));
    [acor,lag] = xcorr(y_window,y_window,maxlag);
    halfway=(length(acor)-1)/2;
    acor(halfway+1-cn*2:halfway+1+cn*2)=0;
    lag(1:halfway+1)=[];
    acor(1:halfway+1)=[];
    [maxcor,mainlag]=max(acor);

     t_mainlag=mainlag/fs;
     norm_tj2j(:,i)=[xj(i);t_mainlag];

    figure(1);
    subplot(3,1,1);plot(t_L(i0:min(i0+iL+1000,length(yjj))),yjj((i0:min(i0+iL+1000,length↙
(yjj)))),'k-');
    hold on
    plot(t_L(i0:min(i0+iL+50,length(yjj))),yjj(i0:min(i0+iL+50,length↙
(yjj))),'b-','LineWidth',2);
    xlabel('time (sec)');
    hold off

    subplot(3,1,2);plot(lag/fs,acor,'b-','LineWidth',2);
    hold on
    plot(t_mainlag,maxcor,'r*');
    xlabel('time delay(sec)');
    hold off
end

i_sel1=[];
tj2j_cor=norm_tj2j(2,:); %length(xj)-1;
tj2j_atrue=t_L(xj(2:end))-t_L(xj(1:end-1)); %length(xj)-1;
tj2j_amiss=(t_L(xj(2:end))-t_L(xj(1:end-1)))/2; %length(xj)-1;
tj2j_afalse=t_L(xj(3:end))-t_L(xj(1:end-2)); %length(xj)-2;
%%%high confidence zone
HCZ_thres=0.1;
for i=1:1:length(tj2j_cor)-2
    %%first find regions where tj2j_cor is mostly constant
    if abs(tj2j_cor(i+1)-tj2j_cor(i))<=tj2j_cor(i)*HCZ_thres
        if abs(tj2j_cor(i+2)-tj2j_cor(i+1))<=tj2j_cor(i+1)*HCZ_thres
            if ismember(i,i_sel1)==false
                i_sel1=[i_sel1 i i+1 i+2];
            elseif ismember(i+1,i_sel1)==false
                i_sel1=[i_sel1 i+1 i+2];
            else
                i_sel1=[i_sel1 i+2];
            end
        end
    end
end
%%low confidence zone
tj2j_cor2=tj2j_cor;
```

```
iLCZ0=1;
iLCZ1=1;
for i=1:1:length(tj2j_cor)
    if ismember(i,i_sel1)==true && ismember(i+1,i_sel1)==false
        iLCZ0=i;
    end
    if ismember(i,i_sel1)==false && ismember(i+1,i_sel1)==true
        iLCZ1=i+1;
        tj2j_cor2(iLCZ0:iLCZ1)=tj2j_cor(iLCZ0)+(tj2j_cor(iLCZ1)-tj2j_cor(iLCZ0))*linspace↙
(0,1,iLCZ1-iLCZ0+1);
    end
end



figure(2);
subplot(2,1,1);plot(t_L,yj*100);
hold on
plot(t_L,gyro_radius_abs);
title('Rotational speed in radial direction(deg/s)');
xlabel('time(sec)');
ylim([0,100]);
hold off
subplot(2,1,2);
plot(t_L(xj(1:end-1)),tj2j_cor,'b-');
hold on
plot(t_L(xj(1:end-1)),tj2j_cor2,'r-');
title('nominal delay estimations(sec)');
xlabel('time(sec)');
plot(t_L(xj(i_sel1)),tj2j_cor(i_sel1),'bo');
plot(t_L(xj(1:end-1)),tj2j_atrue,'r.','markerSize',10);
plot(t_L,yj*50);


i_sel2=[];
xj2=[];
yj2=zeros(1,length(t_L));
HCZ_thres2=0.2;
for i=1:1:length(tj2j_cor2)
    %%consecutive true joints
    if abs(tj2j_cor2(i)-tj2j_atrue(i))<=tj2j_cor2(i)*HCZ_thres2
        i_sel2=[i_sel2 i];
        xj2=[xj2 xj(i)];
    %%miss next true joint
%     elseif abs(tj2j_cor2(i)-tj2j_amiss(i))<=tj2j_cor2(i)*HCZ_thres2
%         i_sel2=[i_sel2 i];
%         xj2=[xj2 xj(i) xj(i)+floor(tj2j_amiss(i)/dt_L)];
    %%next one is a false joint
%     elseif i<=length(tj2j_afalse)
%         if abs(tj2j_cor2(i)-tj2j_afalse(i))<=tj2j_cor2(i)*HCZ_thres2
%             i_sel2=[i_sel2 i];
%             if i+2<length(tj2j_cor2)
%                 xj2=[xj2 xj(i)];
%             else
%                 xj2=[xj2 xj(i) xj(i)+floor(tj2j_afalse(i)/dt_L)];
%             end
%         end
```

```
    end
    plot(t_L(xj(i_sel2)),tj2j_cor2(i_sel2),'b*');
    yj2(xj2)=1;
end

 hold off
```

```
%%find the high confidence zones: at least three joints in a row
i_sel=[];
for i=1:1:length(i_sel2)-1
    if i_sel2(i)+1==i_sel2(i+1)
        if ismember(i_sel2(i),i_sel)==false
            i_sel=[i_sel i_sel2(i) i_sel2(i+1)];
        else
            i_sel=[i_sel i_sel2(i+1)];
        end
    end
end

%i_sel=i_sel2;
v_HCZ=SegPipe./tj2j_atrue;
figure(2);
plot(t_L(xj(i_sel)),v_HCZ(i_sel),'b*');



%%initialize the data record matrices
V_ekf91=zeros(1,length(t_L));
V_ekf92=zeros(1,length(t_L));
%%find the first low confidence zone
i_0j=0;
i_nj=1;
for j=1:1:length(i_sel)-1
    i_0=i_sel(j);
    i_n=i_sel(j+1);
    if i_0+1==i_n%high confidence zone
        %%forward estimation
        d_0=0;
%          V_0=V_ekf9(xj(i_0));
%          if V_0==0
%              V_0=v_HCZ(i_0);
%          end
        V_0=v_HCZ(i_0);
        V_flow_est=V_0;
        a_0=acc_output(xj(i_0));
        X_0j=[d_0; V_flow_est; V_0; a_0];
        P_0j = [0.0001^2 0 0 0;
            0 (V_0*0.02)^2 0 0;
            0 0 (V_0*0.02)^2 0;
            0 0 0 R_acc_est];
        %%%backward smoother
        d_n=SegPipe;
        V_n=v_HCZ(i_n);
        a_n=acc_output(xj(i_n));
        %%%EKF in the HCF Confidence Zone
        ns4_stepEKF_HCZ;
        %%%%%%

        V_ekf91(xj(i_0):xj(i_n))=V_smooth;
        V_ekf92(xj(i_0):xj(i_n))=V_ekfj;
        %                 d_ekf9(xj(i_0):xj(i_n))=d_ekf9(xj(i_0):xj(i_n))+d_smooth;
        %                 d_ekf9(xj(i_n)+1:end)=ones(1,length(t_L)-xj(i_n))*d_ekf9(xj↙
(i_n));
        hold on
        %plot(t_L(xj(i_0):xj(i_n)),V_ekfj,'-');
        plot(t_L(xj(i_0):xj(i_n)),V_smooth,'.');
        hold off

    else %low confidence zone
        %%forward estimation
        V_0=v_HCZ(i_0);
%         V_0=V_ekf91(xj(i_0));
%         if V_0==0
%             V_0=v_HCZ(i_0);
%         end
        V_flow_est=V_0;
        a_0=acc_output(xj(i_0));
        X_0j=[0; V_flow_est; V_0; a_0];
        P_0j = [0.0001^2 0 0 0;
            0 (V_0*0.02)^2 0 0;
            0 0 (V_0*0.02)^2 0;
            0 0 0 R_acc_est];


        %%%backward smoother
        V_n=v_HCZ(i_n);
        a_n=acc_output(xj(i_n));
        %%%EKF in the Low Confidence Zone
        ns4_stepEKF_LCZ;
        %%%

        V_ekf91(xj(i_0):xj(i_n))=V_smooth;
        %V_ekf91(xj(i_0):xj(i_n))=V_ekfj;
        V_ekf92(xj(i_0):xj(i_n))=V_ekfj;
%               d_ekf9(xj(i_0):xj(i_n))=d_ekf9(xj(i_0):xj(i_n))+d_smooth;
%               d_ekf9(xj(i_n)+1:end)=ones(1,length(t_L)-xj(i_n))*d_ekf9(xj(i_n));
        hold on
        plot(t_L(xj(i_0):xj(i_n)),V_ekfj,'-');
        plot(t_L(xj(i_0):xj(i_n)),V_smooth,'.');
%               plot(t_L(xj(i_0j):xj(i_nj)),V_ekfj-V_ekfj_std,'--');
%               plot(t_L(xj(i_0j):xj(i_nj)),V_ekfj+V_ekfj_std,'--');
        hold off
    end
end

%%the segment before the first high confidence zone
i_0=0;
i_n=i_sel(1);
%%forward estimation
V_0=0;
V_flow_est=v_HCZ(i_n);
a_0=cd*1000*A_rc*(0-V_flow_est)^2/(m+ma);
X_0j=[0; V_flow_est; V_0; a_0];
P_0j = [0.0001^2 0 0 0;
            0 (0.01)^2 0 0;
            0 0 (V_flow_est*0.2)^2 0;
            0 0 0 a_0^2];
%%%backward smoother
V_n=v_HCZ(i_n);
a_n=acc_output(xj(i_n));
%%%EKF in the Low Confidence Zone
ns4_stepEKF_LCZ;
%%%
```

```
V_ekf91(1:xj(i_n))=V_smooth;
V_ekf92(1:xj(i_n))=V_ekfj;
%
% for i=2:1:length(V_smooth)
%     d_ekf9(i)=d_ekf9(i-1)+V_ekf9(i)*dt_L;
% end
% %d_ekf9(1:xj(i_n))=d_ekf9(1:xj(i_n))+d_smooth;
% d_ekf9(xj(i_n)+1:end)=d_ekf9(xj(i_n)+1:end)+ones(1,length(t_L)-xj(i_n))*d_ekf9(xj↙
(i_n));
hold on
plot(t_L(1:xj(i_n)),V_ekfj,'-');
plot(t_L(1:xj(i_n)),V_smooth,'.');
hold off


%%%the segment after the last high confidence zone
i_0=i_sel(end);
i_n=length(xj)+1;
%%forward estimation
V_0=v_HCZ(i_0);
V_flow_est=V_0;
a_0=acc_output(xj(i_0));
X_0j=[0; V_flow_est; V_0; a_0];
P_0j = [0.0001^2 0 0 0;
    0 (V_0*0.02)^2 0 0;
    0 0 (V_0*0.02)^2 0;
    0 0 0 R_acc_est];

%%%no backward smoother
%%%EKF in the Low Confidence Zone
ns4_stepEKF_HCZ;
%%%
V_ekf91(xj(i_0):end)=V_ekfj;
V_ekf92(xj(i_0):end)=V_ekfj;

% d_ekf9(xj(i_0):end)=d_ekf9(xj(i_0):end)+d_ekfj;
hold on
plot(t_L(xj(i_0):end),V_ekfj,'-');
%plot(t_L(xj(i_0):end),V_smooth,'.');
%             plot(t_L(xj(i_0j):xj(i_nj)),V_ekfj-V_ekfj_std,'--');
%             plot(t_L(xj(i_0j):xj(i_nj)),V_ekfj+V_ekfj_std,'--');
hold off

d_ekf91=zeros(1,length(V_ekf91));
d_ekf92=zeros(1,length(V_ekf92));
for i=1:1:length(V_ekf91)-1
    d_ekf91(i+1)=d_ekf91(i)+(V_ekf91(i)+V_ekf91(i+1))/2*dt_L;
    d_ekf92(i+1)=d_ekf92(i)+(V_ekf92(i)+V_ekf92(i+1))/2*dt_L;
end

figure(3)
subplot(2,1,1);plot(t_L,v_r_L,'r','LineWidth',2);
hold on
title('Robot Velocity(m/s)');
plot(t_L,V_ekf91,'k','LineWidth',2);
%plot(t_L,V_ekf92,'LineWidth',2);
plot(t_L(xj(i_sel)),zeros(1,length(i_sel)),'k.','MarkerSize',20);
```

```
%plot(t_L,yj*1);
legend('actual','Estimation with EKF+Smoothing','High confidence joints');%'all possible↙
joints');
xlabel('time(sec)');
hold off

subplot(2,1,2);
plot(t_L,abs(d_ekf91-dis_r_L),'k','LineWidth',2);
hold on
title('Distance Estimation Error(m)');
xlabel('time(sec)');
plot(t_L(xj(i_sel)),zeros(1,length(i_sel)),'k.','MarkerSize',20);
%plot(t_L,yj*1);
legend('Estimation with EKF+Smoothing','high confidence joints');%'all possible joints');
hold off

%
```

```
%%%EKF in the errorish range
 if i_0==0
     Z_obsv=acc_output(1:xj(i_n));
 elseif i_n>length(xj)
     Z_obsv=acc_output(xj(i_0):end);
 else
    Z_obsv=acc_output(xj(i_0):xj(i_n));
 end
coef=cd*1000*A_rc/(m+ma);
%state error
Q_VfVf=(0.1/10/f_imu)^2;
Q_VrVf=Q_VfVf/2;
Q_VrVr=0;%Q_aa*dt_L^2;
Q_aa=0.2^2;
Q_Vra=0;%Q_aa*dt_L^2;%Q_aa/50;
%Q_flow_est=1;
Q_ss=Q_VrVr*dt_L^2;
Q_process=1*[Q_ss 0 0 0 ;
     0 Q_VfVf Q_VrVf 0;
     0 Q_VrVf Q_VfVf Q_Vra;
     0 0 Q_Vra Q_aa];
% %+Q_flow_est*[0 0 0 0 0;
%     0   dt_L^3/3    dt_L^2/2    0   0;
%     0   dt_L^2/2    dt_L        0   0;
%     0   0           0           0   0;
%     0   0           0           0   0];

% Measurement-state Jacobian
H = [0 0 0 1];
% Kalman Gain
K = zeros(4,1,length(Z_obsv));
% Apriori state estimates
X_aposteriori=zeros(4,length(Z_obsv));
% Aposteriori state estimates
X_apriori=zeros(4,length(Z_obsv));
% Apriori error covariance estimates
P_apriori = zeros(4,4,length(Z_obsv));
% Aposteriori error covariance estimates
P_aposteriori = zeros(4,4,length(Z_obsv));

X_aposteriori(:,1)=X_0j;
X_apriori(:,1)=X_aposteriori(:,1);
P_aposteriori(:,:,1) = P_0j;
%%forward estimation
for i=2:1:length(Z_obsv)
    Ai=[1   0     dt_L    dt_L^2;
        0   1     0       0;
        0   0     1       dt_L;
        0 coef*abs(X_aposteriori(2,i-1)-X_aposteriori(3,i-1)) -coef*abs(X_aposteriori(2,↙
i-1)-X_aposteriori(3,i-1)) 0];
    % Update apriori estimate
    X_apriori(:,i)=Ai*X_aposteriori(:,i-1);

    % Update state Jacobian

    % Assume knowledge of Q and R (Use system I.D. techniques in practice)
    Qi = Q_process;
    % measurement error
    Ri= R_acc_est;

    % Update aprioiri error covariance estimate
    P_apriori(:,:,i) = Ai*P_aposteriori(:,:,i-1)*Ai' + Qi';

    % Update Kalman gain
    K(:,:,i) = P_apriori(:,:,i)*H' / (H*P_apriori(:,:,i)*H'+Ri);

    % Update aposteriori state estimate
    X_aposteriori(:,i) = X_apriori(:,i) + K(:,:,i) * (Z_obsv(:,i) - H*X_apriori(:,i));

    % Update aposteriori error covariance estimate
    P_aposteriori(:,:,i) = (eye(4) - K(:,:,i)*H) * P_apriori(:,:,i);
end


a_ekfj=X_aposteriori(4,:);
a_ekfj_var=P_aposteriori(4,4,1:end);
a_ekfj_std=reshape(sqrt(a_ekfj_var),1,[]);

V_ekfj=X_aposteriori(3,:);
V_ekfj_var=P_aposteriori(3,3,1:end);
V_ekfj_std=reshape(sqrt(V_ekfj_var),1,[]);

Vf_ekfj=X_aposteriori(2,:);

d_ekfj=X_aposteriori(1,:);
d_ekfj_var=P_aposteriori(1,1,:);
d_ekfj_std=reshape(sqrt(d_ekfj_var),1,[]);

d_ekfj_min=max(d_ekfj-d_ekfj_std,0);
d_ekfj_max=d_ekfj+d_ekfj_std;


%%backward smoother
if i_n>length(xj)

else
    if (i_n-i_0)==1   %%HCZ
        X_nj=[SegPipe; V_n; V_n; a_n];
        P_nj = P_0j;
    else %LCZ
        if i_0==0
            LCZjmid=ceil(V_n*dt_L*(xj(i_n))/SegPipe);


        else
            LCZjmid=ceil((V_ekfj(1)+V_ekfj(end))/2*dt_L*(xj(i_n)-xj(i_0))/SegPipe);
        end
            LCZjdis=6*(LCZjmid+[-3,-2,-1,0,1,2,3]);
            P_LCZj=normpdf(LCZjdis,d_ekfj(end),d_ekfj_std(end));
            [P_LCZ_max,LCZj_est]=max(P_LCZj);
            d_ekfj_real=LCZjdis(LCZj_est);

        X_nj=[d_ekfj_real; V_n; V_n; a_n];
        P_nj = [d_ekfj_std(end)^2 0 0 0;
            0 (V_n*0.2)^2 0 0;
            0 0 (V_n*0.2)^2 0;
            0 0 0 R_acc_est];
    end
    X_back=zeros(4,length(Z_obsv));
    P_back=zeros(4,4,length(Z_obsv));
    C = zeros(4,4,length(Z_obsv));
    X_back(:,length(Z_obsv))=X_nj;
    P_back(:,:,length(Z_obsv))=P_nj;
    for j=length(Z_obsv)-1:-1:1
        Ai=[1   0     dt_L    dt_L^2;
            0   1     0       0;
            0   0     1       dt_L;
            0 coef*abs(X_aposteriori(2,j)-X_aposteriori(3,j)) -coef*abs(X_aposteriori(2,↙
j)-X_aposteriori(3,j)) 0];
        C(:,:,j)=P_aposteriori(:,:,j)*Ai'*inv(P_apriori(:,:,j+1));
        X_back(:,j)=X_aposteriori(:,j)+C(:,:,j)*(X_back(:,j+1)-X_apriori(:,j+1));
        P_back(:,:,j)=P_aposteriori(:,:,j)+C(:,:,j)*(P_back(:,:,j+1)-P_apriori(:,:,j+1))↙
*C(:,:,j)';
    end

    V_smooth=X_back(3,:);
    %d_smooth=X_back(1,:);
end
```

```
%%%EKF in the errorish range
 if i_0==0
     Z_obsv=acc_output(1:xj(i_n));
 elseif i_n>length(xj)
     Z_obsv=acc_output(xj(i_0):end);
 else
    Z_obsv=acc_output(xj(i_0):xj(i_n));
 end
coef=cd*1000*A_rc/(m+ma);
%state error
Q_VfVf=(0.1/10/f_imu)^2;
Q_VrVf=Q_VfVf/2;
Q_VrVr=0;%Q_aa*dt_L^2;
Q_aa=0.2^2;
Q_Vra=0;%Q_aa*dt_L^2;%Q_aa/50;
%Q_flow_est=1;
Q_ss=Q_VrVr*dt_L^2;
Q_process=1*[Q_ss 0 0 0 ;
     0 Q_VfVf Q_VrVf 0;
     0 Q_VrVf Q_VfVf Q_Vra;
     0 0 Q_Vra Q_aa];
% %+Q_flow_est*[0 0 0 0 0;
%     0   dt_L^3/3    dt_L^2/2    0   0;
%     0   dt_L^2/2    dt_L        0   0;
%     0   0           0           0   0;
%     0   0           0           0   0];

% Measurement-state Jacobian
H = [0 0 0 1];
% Kalman Gain
K = zeros(4,1,length(Z_obsv));
% Apriori state estimates
X_aposteriori=zeros(4,length(Z_obsv));
% Aposteriori state estimates
X_apriori=zeros(4,length(Z_obsv));
% Apriori error covariance estimates
P_apriori = zeros(4,4,length(Z_obsv));
% Aposteriori error covariance estimates
P_aposteriori = zeros(4,4,length(Z_obsv));

X_aposteriori(:,1)=X_0j;
X_apriori(:,1)=X_aposteriori(:,1);
P_aposteriori(:,:,1) = P_0j;
%%forward estimation
for i=2:1:length(Z_obsv)
    Ai=[1   0     dt_L    dt_L^2;
        0   1     0       0;
        0   0     1       dt_L;
        0 coef*abs(X_aposteriori(2,i-1)-X_aposteriori(3,i-1)) -coef*abs(X_aposteriori(2,↙
i-1)-X_aposteriori(3,i-1)) 0];
    % Update apriori estimate
    X_apriori(:,i)=Ai*X_aposteriori(:,i-1);

    % Update state Jacobian

    % Assume knowledge of Q and R (Use system I.D. techniques in practice)
    Qi = Q_process;
    % measurement error
    Ri= R_acc_est;

    % Update aprioiri error covariance estimate
    P_apriori(:,:,i) = Ai*P_aposteriori(:,:,i-1)*Ai' + Qi';

    % Update Kalman gain
    K(:,:,i) = P_apriori(:,:,i)*H' / (H*P_apriori(:,:,i)*H'+Ri);

    % Update aposteriori state estimate
    X_aposteriori(:,i) = X_apriori(:,i) + K(:,:,i) * (Z_obsv(:,i) - H*X_apriori(:,i));

    % Update aposteriori error covariance estimate
    P_aposteriori(:,:,i) = (eye(4) - K(:,:,i)*H) * P_apriori(:,:,i);
end


a_ekfj=X_aposteriori(4,:);
a_ekfj_var=P_aposteriori(4,4,1:end);
a_ekfj_std=reshape(sqrt(a_ekfj_var),1,[]);

V_ekfj=X_aposteriori(3,:);
V_ekfj_var=P_aposteriori(3,3,1:end);
V_ekfj_std=reshape(sqrt(V_ekfj_var),1,[]);

Vf_ekfj=X_aposteriori(2,:);

d_ekfj=X_aposteriori(1,:);
d_ekfj_var=P_aposteriori(1,1,:);
d_ekfj_std=reshape(sqrt(d_ekfj_var),1,[]);

d_ekfj_min=max(d_ekfj-d_ekfj_std,0);
d_ekfj_max=d_ekfj+d_ekfj_std;


%%backward smoother
if i_n>length(xj)

else


        d_ekfj_real=d_ekfj(end);

        X_nj=[d_ekfj_real; V_n; V_n; a_n];
        P_nj = [d_ekfj_std(end)^2 0 0 0;
            0 (V_n*0.2)^2 0 0;
            0 0 (V_n*0.2)^2 0;
            0 0 0 R_acc_est];
    X_back=zeros(4,length(Z_obsv));
    P_back=zeros(4,4,length(Z_obsv));
    C = zeros(4,4,length(Z_obsv));
    X_back(:,length(Z_obsv))=X_nj;
    P_back(:,:,length(Z_obsv))=P_nj;
    for j=length(Z_obsv)-1:-1:1
        Ai=[1   0     dt_L    dt_L^2;
            0   1     0       0;
            0   0     1       dt_L;
            0 coef*abs(X_aposteriori(2,j)-X_aposteriori(3,j)) -coef*abs(X_aposteriori(2,↙
j)-X_aposteriori(3,j)) 0];
```

```
        C(:,:,j)=P_aposteriori(:,:,j)*Ai'*inv(P_apriori(:,:,j+1));
        X_back(:,j)=X_aposteriori(:,j)+C(:,:,j)*(X_back(:,j+1)-X_apriori(:,j+1));
        P_back(:,:,j)=P_aposteriori(:,:,j)+C(:,:,j)*(P_back(:,:,j+1)-P_apriori(:,:,j+1))↙
*C(:,:,j)';
    end

    V_smooth=X_back(3,:);
    %d_smooth=X_back(1,:);
end
```

```
%%find the high confidence zones: at least three joints in a row
i_sel=[];
for i=1:1:length(i_sel2)-1
    if i_sel2(i)+1==i_sel2(i+1)
        if ismember(i_sel2(i),i_sel)==false
            i_sel=[i_sel i_sel2(i) i_sel2(i+1)];
        else
            i_sel=[i_sel i_sel2(i+1)];
        end
    end
end
%%%unselect at first run
i_sel=i_sel2;
v_HCZ=SegPipe./tj2j_atrue;
V_tac_noise=V_tac_var/2*randn(1,length(xj));
V_tac_L=zeros(1,length(xj));
V_tac_est=zeros(1,length(xj));
for i=1:1:length(xj)
    V_tac_L(i)=mean(v_r_L(xj(i)-2:xj(i)+2));

    V_tac_est(i)=V_tac_scale*abs(V_tac_L(i)+V_tac_noise(i));
end
figure(2);
plot(t_L(xj),zeros(1,length(xj)),'k.', 'MarkerSize',5);
hold on
plot(t_L,zeros(1,length(t_L)),'k-', 'LineWidth',1);
plot(t_L(xj(i_sel)),zeros(1,length(i_sel)),'b.','MarkerSize',20);
for i=1:1:length(i_sel)-1
    if i_sel(i)+1==i_sel(i+1)
        %plot(t_L(xj(i_sel(i)):xj(i_sel(i+1))),V_ekf91(xj(i_sel(i)):xj(i_sel(i+1))),↙
'LineWidth',2);
        plot(t_L(xj(i_sel(i)):xj(i_sel(i+1))),yj(xj(i_sel(i)):xj(i_sel(i+1))),'b');
    end
end
plot(t_L(xj),V_tac_est,'k.','MarkerSize',10);
hold off
%%initialize the data record matrices
V_ekf91=zeros(1,length(t_L));
V_ekf92=zeros(1,length(t_L));
%%find the first low confidence zone
i_0j=0;
i_nj=1;
for j=1:1:length(xj)-1
    i_0=j;
    i_n=j+1;
    if ismember(i_0,i_sel) && ismember(i_n,i_sel)%high confidence zone
        %%forward estimation
        d_0=0;
%         V_0=V_ekf9(xj(i_0));
%         if V_0==0
%             V_0=v_HCZ(i_0);
%         end
        V_0=V_tac_est(i_0);
        V_flow_est=V_0;
        a_0=acc_output(xj(i_0));
        X_0j=[d_0; V_flow_est; V_0; a_0];
        P_0j = [0.0001^2 0 0 0;
            0 (V_0*0.02)^2 0 0;
            0 0 (V_0*0.02)^2 0;
            0 0 0 R_acc_est];
        %%%backward smoother
        d_n=SegPipe;
        V_n=v_HCZ(i_n);
        a_n=acc_output(xj(i_n));
        %%%EKF in the HCF Confidence Zone
        ns4_stepEKF_HCZ;
        %%%%%%%

        V_ekf91(xj(i_0):xj(i_n))=V_smooth;
        V_ekf92(xj(i_0):xj(i_n))=V_ekfj;
%                   d_ekf9(xj(i_0):xj(i_n))=d_ekf9(xj(i_0):xj(i_n))+d_smooth;
%                   d_ekf9(xj(i_n)+1:end)=ones(1,length(t_L)-xj(i_n))*d_ekf9(xj↙
(i_n));
        hold on
        %plot(t_L(xj(i_0):xj(i_n)),V_ekfj,'-');
        plot(t_L(xj(i_0):xj(i_n)),V_smooth,'.');
        hold off

    else %low confidence zone
        %%forward estimation

            V_0=V_tac_est(i_0);
            V_flow_est=V_0;
            a_0=acc_output(xj(i_0));
            X_0j=[0; V_flow_est; V_0; a_0];
            P_0j = [0.0001^2 0 0 0;
                0 (V_0*0.2)^2 0 0;
                0 0 (V_0*0.2)^2 0;
                0 0 0 R_acc_est];


            %%%backward smoother
            V_n=V_tac_est(i_n);
            a_n=acc_output(xj(i_n));
            %%%EKF in between
            ns4_stepEKF_LCZ;
            %%%

            V_ekf91(xj(i_0):xj(i_n))=V_smooth;
            V_ekf92(xj(i_0):xj(i_n))=V_ekfj;
            hold on
            %plot(t_L(xj(i_0):xj(i_n)),V_ekfj,'-');
            plot(t_L(xj(i_0):xj(i_n)),V_smooth,'.');
            hold off

    end
end

%%the segment before the first high confidence zone
i_0=0;
i_n=i_sel(1);
%%forward estimation
V_0=0;
V_flow_est=v_HCZ(i_n);
a_0=cd*1000*A_rc*(0-V_flow_est)^2/(m+ma);
X_0j=[0; V_flow_est; V_0; a_0];
```

```
P_0j = [0.0001^2 0 0 0;
            0 (V_0*0.2)^2 0 0;
            0 0 (V_0*0.2)^2 0;
            0 0 0 a_0^2];
%%%backward smoother
V_n=V_tac_est(i_n);
a_n=acc_output(xj(i_n));
%%%EKF in the Low Confidence Zone
ns4_stepEKF_LCZ;
%%%

V_ekf91(1:xj(i_n))=V_smooth;
V_ekf92(1:xj(i_n))=V_ekfj;
%
% for i=2:1:length(V_smooth)
%     d_ekf9(i)=d_ekf9(i-1)+V_ekf9(i)*dt_L;
% end
% %d_ekf9(1:xj(i_n))=d_ekf9(1:xj(i_n))+d_smooth;
% d_ekf9(xj(i_n)+1:end)=d_ekf9(xj(i_n)+1:end)+ones(1,length(t_L)-xj(i_n))*d_ekf9(xj↙
(i_n));
hold on
plot(t_L(1:xj(i_n)),V_ekfj,'-');
plot(t_L(1:xj(i_n)),V_smooth,'.');
hold off


%%%the segment after the last high confidence zone
i_0=length(xj);
i_n=length(xj)+1;
%%forward estimation
V_0=V_tac_est(i_0);
V_flow_est=V_0;
a_0=acc_output(xj(i_0));
X_0j=[0; V_flow_est; V_0; a_0];
P_0j = [0.0001^2 0 0 0;
    0 (V_0*0.2)^2 0 0;
    0 0 (V_0*0.2)^2 0;
    0 0 0 R_acc_est];

%%%no backward smoother
%%%EKF in the Low Confidence Zone
ns4_stepEKF_LCZ;
%%%
V_ekf91(xj(i_0):end)=V_ekfj;
V_ekf92(xj(i_0):end)=V_ekfj;

% d_ekf9(xj(i_0):end)=d_ekf9(xj(i_0):end)+d_ekfj;
hold on
plot(t_L(xj(i_0):end),V_ekfj,'-');
hold off

d_ekf91=zeros(1,length(V_ekf91));
d_ekf92=zeros(1,length(V_ekf92));
for i=1:1:length(V_ekf91)-1
    d_ekf91(i+1)=d_ekf91(i)+(V_ekf91(i)+V_ekf91(i+1))/2*dt_L;
    d_ekf92(i+1)=d_ekf92(i)+(V_ekf92(i)+V_ekf92(i+1))/2*dt_L;
end
```

```
figure(3)
subplot(2,1,1);plot(t_L,v_r_L,'r','LineWidth',2);
hold on
title('Robot Velocity(m/s)');
plot(t_L,V_ekf91,'k','LineWidth',2);
%plot(t_L,V_ekf92,'LineWidth',2);
plot(t_L(xj(i_sel)),zeros(1,length(i_sel)),'k.','MarkerSize',20);
%plot(t_L,yj*1);
legend('actual','Estimation with EKF+Smoothing','High confidence joints');%'all possible↙
joints');
xlabel('time(sec)');
hold off

subplot(2,1,2);
plot(t_L,abs(d_ekf91-dis_r_L),'k','LineWidth',2);
hold on
title('Distance Estimation Error(m)');
xlabel('time(sec)');
plot(t_L(xj(i_sel)),zeros(1,length(i_sel)),'k.','MarkerSize',20);
%plot(t_L,yj*1);
legend('Estimation with EKF+Smoothing','high confidence joints');%'all possible joints');
hold off


%
```

Bibliography

[1] D. Martínez Martínez, J. Nohava, and J. T. M. De Hosson, "Influence of load on the dry frictional performance of alkyl acrylate copolymer elastomers coated with diamond-like carbon films," *Journal of Applied Physics*, vol. 118, no. 17, p. 175302, 2015.

[2] R. K. Kramer, C. Majidi, R. Sahai, and R. J. Wood, "Soft curvature sensors for joint angle proprioception," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on.* IEEE, 2011, pp. 1919–1926.

[3] A. D. Valentine, T. A. Busbee, J. W. Boley, J. R. Raney, A. Chortos, A. Kotikian, J. D. Berrigan, M. F. Durstock, and J. A. Lewis, "Hybrid 3d printing of soft electronics," *Advanced Materials*, vol. 29, no. 40, 2017.

[4] J. U. Lind, T. A. Busbee, A. D. Valentine, F. S. Pasqualini, H. Yuan, M. Yadid, S.-J. Park, A. Kotikian, A. P. Nesmith, P. H. Campbell, *et al.*, "Instrumented cardiac microphysiological devices via multimaterial three-dimensional printing," *Nature materials*, vol. 16, no. 3, p. 303, 2017.

[5] P. Valdivia y Alvarado and K. Youcef-Toumi, "Design of machines with compliant bodies for biomimetic locomotion in liquid environments," *ASME Journal of Dynamic Systems measurement and Control*, 2006.

[6] B. MegH. (2016) The difference between knit and woven fabrics. [Online]. Available: http://www.burdastyle.com/blog/the-difference-between-knit-and-woven-fabrics

[7] "Urban life: Open air computers," *The Economist*, 2012.

[8] B. Cohen, "Urbanization, city growth, and the new united nations development agenda," *Cornerstone, The Official Journal of the World Coal Industry*, pp. 4–7, 2015.

[9] R. Liemberger, P. Marin, *et al.*, "The challenge of reducing non-revenue water in developing countries–how the private sector can help: A look at performance-based service contracting," 2006.

[10] "2017 infrastructure report card," *American Society of Civil Engineers*, 2017.

[11] Y. Wu, "Interview, mark gallagher, director of engineering, cambridge water department, massachusetts, us," Mar 2017.

[12] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline, "Pipenet: A wireless sensor network for pipeline monitoring," in *2007 6th International Symposium on Information Processing in Sensor Networks.* IEEE, 2007, pp. 264–273.

[13] P. Schumi, "Advanced leak detection technology utilizing satellite imagery." Water Asset Management Conference, 2017.

[14] A. Cataldo, G. Cannazza, E. De Benedetto, and N. Giaquinto, "A new method for detecting leaks in underground water pipelines," *IEEE Sensors Journal*, vol. 12, no. 6, pp. 1660–1667, 2012.

[15] R. Fletcher and M. Chandrasekaran, "SmartballâĎć: A new approach in pipeline leak detection," in *2008 7th International Pipeline Conference.* American Society of Mechanical Engineers, 2008, pp. 117–133.

[16] S. Folkman, "Water main break rates in the usa and canada: A comprehensive study," *Utah State University Mechanical and Aerospace Engineering Faculty Publications*, vol. 2018, 2018.

[17] B. Mergelas, M. Larsen, B. Bengtsson, L. Lawrence, and R. Thomas, "Using in-line acoustics to identify leaks in pre-commissioned pipelines," *Proceedings of ASCE Pipelines*, 2005.

[18] D. M. Chatzigeorgiou, Y. Wu, K. Youcef-Toumi, and R. Ben-Mansour, "Reliable sensing of leaks in pipelines," in *ASME 2013 Dynamic Systems and Control Conference.* American Society of Mechanical Engineers, 2013, pp. V002T25A004–V002T25A004.

[19] D. Chatzigeorgiou, K. Youcef-Toumi, and R. Ben-Mansour, "Design of a novel in-pipe reliable leak detector," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 824–833, 2015.

[20] R. Fletcher and M. Chandrasekaran, "SmartballÂł: A new approach in pipeline leak detection," in *2008 7th International Pipeline Conference.* American Society of Mechanical Engineers, 2008, pp. 117–133.

[21] Y. Wu, A. Noel, D. D. Kim, K. Youcef-Toumi, and R. Ben-Mansour, "Design of a maneuverable swimming robot for in-pipe missions," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on.* IEEE, 2015, pp. 4864–4871.

[22] J. Quarini and S. Shire, "A review of fluid-driven pipeline pigs and their applications," *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, vol. 221, no. 1, pp. 1–10, 2007.

[23] Schempf, Hagen and Mutschler, Edward and Goltsberg, Vitaly and Skoptsov, George and Gavaert, Alan and Vradis, George, "Explorer: Untethered real-time gas main assessment robot system," in *Proc. of Int. Workshop on Advances in Service Robotics, ASER*, vol. 3, 2003.

[24] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences*, vol. 108, no. 51, pp. 20 400–20 403, 2011.

[25] M. T. Tolley, R. F. Shepherd, B. Mosadegh, K. C. Galloway, M. Wehner, M. Karpelson, R. J. Wood, and G. M. Whitesides, "A resilient, untethered soft robot," *Soft Robotics*, vol. 1, no. 3, pp. 213–223, 2014.

[26] A. Frutiger, J. T. Muth, D. M. Vogt, Y. Mengüç, A. Campo, A. D. Valentine, C. J. Walsh, and J. A. Lewis, "Capacitive soft strain sensors via multicore–shell fiber printing," *Advanced Materials*, vol. 27, no. 15, pp. 2440–2446, 2015.

[27] J. T. Muth, D. M. Vogt, R. L. Truby, Y. Mengüç, D. B. Kolesky, R. J. Wood, and J. A. Lewis, "Embedded 3d printing of strain sensors within highly stretchable elastomers," *Advanced Materials*, vol. 26, no. 36, pp. 6307–6312, 2014.

[28] A. Cloitre, V. Subramaniam, N. Patrikalakis, and P. V. y Alvarado, "Design and control of a field deployable batoid robot," in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob).* IEEE, 2012, pp. 707–712.

[29] P. V. y Alvarado and K. Youcef-Toumi, "Design of machines with compliant bodies for biomimetic locomotion in liquid environments," *Journal of dynamic systems, measurement, and control*, vol. 128, no. 1, pp. 3–13, 2006.

[30] D. Chatzigeorgiou, You Wu, K. Youcef-Toumi and R. Ben-Mansour, "Reliable sensing of leaks in pipelines," in *ASME Dynamic Systems and Control Conference*, 2013.

[31] D. Chatzigeorgiou, K. Youcef-Toumi and R. Ben-Mansour, "Design of a novel in-pipe reliable leak detector," *IEEE/ASME Transactions on Mechatronics*, 2014.

[32] J. Voyer, F. Ausserer, S. Klien, I. Velkavrh, and A. Diem, "Reduction of the adhesive friction of elastomers through laser texturing of injection molds," *Lubricants*, vol. 5, no. 4, p. 45, 2017.

[33] J.-Y. Sun, X. Zhao, W. R. Illeperuma, O. Chaudhuri, K. H. Oh, D. J. Mooney, J. J. Vlassak, and Z. Suo, "Highly stretchable and tough hydrogels," *Nature*, vol. 489, no. 7414, p. 133, 2012.

[34] D. Chatzigeorgiou, R. Ben-Mansour, A. Khalifa and K. Youcef-Toumi, "Design and evaluation of an in-pipe leak detection sensing technique based on force transduction," in *ASME International Mechanical Engineering Congress and Exposition*, 2012.

[35] D. Chatzigeorgiou, K. Youcef-Toumi, A. Khalifa and R. Ben-Mansour, "Analysis and design of an in-pipe system for water leak detection," in *ASME International Design Engineering Technical Conferences and Design Automation Conference*, 2011.

[36] Dimitris Chatzigeorgiou, Kamal Youcef-Toumi and R. Ben-Mansour, "Modeling and analysis of an in-pipe robotic leak detector," in *IEEE International Conference on Robotics and Automation*, 2014.

[37] Y. Wu, K. Kim, M. F. Henry, and K. Youcef-Toumi, "Design of a leak sensor for operating water pipe systems," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on.* IEEE, 2017, pp. 6075–6082.

[38] D. M. Chatzigeorgiou, K. Youcef-Toumi, A. E. Khalifa, and R. Ben-Mansour, "Analysis and design of an in-pipe system for water leak detection," in *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* American Society of Mechanical Engineers, 2011, pp. 1007–1016.

[39] D. M. Vogt, Y.-L. Park, and R. J. Wood, "Design and characterization of a soft multi-axis force sensor using embedded microfluidic channels," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 4056–4064, 2013.

[40] Michael S. Triantafyllou, Franz S. Hover, *Maneuvering and Control of Marine Vehicles*, 2002.

[41] A. Phillips, "Robot fish: Bio-inspired fishlike underwater robots," *Underwater Technology*, vol. 34, no. 3, pp. 143–145, 2017.

[42] G. Liu, A. Wang, X. Wang, and P. Liu, "A review of artificial lateral line in sensor fabrication and bionic applications for robot fish," *Applied bionics and biomechanics*, vol. 2016, 2016.

[43] Dalei Wu, Kamal Youcef-Toumi, Samir Mekid and Rached Ben Mansour, "Relay node placement in wireless sensor networks for pipeline inspection," in *American Control Conference*, 2013.

[44] C. Jun, Z. Deng, and S. Jiang, "Study of locomotion control characteristics for six wheels driven in-pipe robot," in *2004 IEEE International Conference on Robotics and Biomimetics*, Aug 2004, pp. 119–124.

[45] Q. Gan and C. J. Harris, "Comparison of two measurement fusion methods for kalman-filter-based multisensor data fusion," *IEEE Transactions on Aerospace and Electronic systems*, vol. 37, no. 1, pp. 273–279, 2001.

[46] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," in *Springer handbook of robotics.* Springer, 2008, pp. 871–889.

[47] H. H. Aghdam, H. A. Kadir, M. R. Arshad, and M. Zaman, "Localizing pipe inspection robot using visual odometry," in *2014 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2014)*, Nov 2014, pp. 245–250.

[48] Lee, Jung-Sub and Se-gon Roh and Kim, Do Wan and Hyungpil Moon and Hyouk-Ryeol Choi, "In-pipe robot navigation based on the landmark recognition system using shadow images," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 1857–1862.

[49] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on.* IEEE, 2013, pp. 1280–1286.

[50] (2016) Imu noise model. [Online]. Available: https://github.com/ethz-asl/kalibr/wiki/IMU-Noise-Model

[51] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches.* John Wiley & Sons, 2006.