Detection of Dynamic Obstacles out of the Line of Sight for Autonomous Vehicles to increase Safety based on Shadows

by

Felix Maximilian Naser

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2019

Author . . . .
Department of Electrical Engineering and Computer Science
January 18, 2019

Certified by . . . .
Daniela Rus
Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Detection of Dynamic Obstacles out of the Line of Sight for Autonomous Vehicles to increase Safety based on Shadows

by

Felix Maximilian Naser

Submitted to the Department of Electrical Engineering and Computer Science
on January 18, 2019, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Moving obstacles occluded by corners are a potential source for collisions in mobile robotics applications such as autonomous vehicles. In this work, we address the problem of anticipating such potential collisions by proposing a vision-based detection algorithm for obstacles which are outside of a vehicle's direct line of sight.

Our method detects shadows of obstacles hidden around corners and automatically classifies these unseen obstacles as "dynamic" or "static". We evaluate our proposed detection algorithm on real-world corners and a large variety of simulated environments to assess generalizability in different challenging surface and lighting conditions. For the image registration step we compare a visual odometry method (i.e. DSO) with a fiducial marker system (i.e. AprilTags). The mean classification accuracy on simulated data is around 80% and on real-world corners approximately for both image registration methods 70%. Additionally, we integrate our detection system on a full-scale autonomous wheelchair and demonstrate its feasibility as an additional safety mechanism through real-world experiments.

We release our real-time-capable implementation of the proposed ShadowCam algorithm and the dataset containing simulated and real-world data under an open-source license.

Thesis Supervisor: Daniela Rus
Title: Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science

Acknowledgments

First and foremost I want to thank my advisor Prof. Daniela Rus. Her continuous support, interest, guidance and inspiring discussions about our project have made the work presented in this thesis possible. She helped me to focus on the important technical contributions first and enabled me to learn and grow with the project.

I want to thank my friends and colleagues for being supportive and a wonderful community full of crazy ideas, curiosity and knowledge. The past three years would have not been the same without you.

Specifically I want to thank my collaborators and supporters for the work presented in this thesis: Dr. Igor Gilitschenski, Alexander Amini, Thomas Balch, Steve Proulx, Dr. Guy Rosman, Prof. Fredo Durand, Prof. Antonio Torralba, Prof. Gregory W. Wornell, Prof. William T. Freeman and Prof. Sertac Karaman.

I want to thank our sponsors at Toyota Research Institute (TRI) and AWS for the generous support of our work. Toyota Research Institute (TRI) provided funds to assist the authors with their research, but this thesis solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

A special thanks also goes to my UROPs Christina Liao (who joined this project in February 2018) and Puneeth Meruva (who joined in September 2018). Christina contributed to the data collection, annotation and data analysis parts of the project. Puneeth contributed to the literature review.

Last but not least I want to thank my girlfriend and my parents for their support, patience and love.

Contents

List of Figures

List of Tables

Nomenclature

ADAS	Advanced Driver Assistance Systems, page 19
AT	visual fiducial system AprilTags, page 24
DOF	Degrees of Freedom, page 36
DSO	Direct Sparse Odometry, page 24
fps	frames per second, page 52
LiDAR	light detection and ranging, page 21
NL	New Label, page 65
NLoS	Non-Line-of-Sight, page 25
ORB	Oriented FAST and rotated BRIEF, page 35
RADAR	radio detection and ranging, page 21
ROC	Receiver-Operating-Characteristic, page 63
ROI	Region of Interest, page 23
ROS	Robotic Operating System, page 23
SIFT	scale-invariant feature transform, page 35
SNR	signal-to-noise ratio, page 22
SURF	speeded up robust features, page 35

TP	True Positive, page 65
USA	United States of America, page 19
UWB	ultra-wideband, page 25
WHO	World Health Organization, page 19

Chapter 1

Introduction

Safety is a key challenge and promise of future mobility solutions, specifically of self-driving cars. According to the World Health Organization (WHO) around 1.3M people are losing their lives in road accidents every year[1]. Mandatory passive safety features such as the seat-belt and airbags are helping to reduce the consequences of accidents. In addition and more recently active safety features such as Advanced Driver Assistance Systems (ADAS) and autonomous driving research have come a long way to deliver on the promise of safer driving. Even though the number of vehicles on the roads is increasing, the numbers of fatal road accidents show a decreasing tendency in the United States of America (USA) since 1990 (Fig. 1-1).

"Vision Zero is a strategy to eliminate all traffic fatalities and severe injuries, while increasing safe, healthy, equitable mobility for all"[3,4]. But despite the encouraging trend in the USA we still have a long way to go to make Vision Zero a reality. In addition to improvements to existing methods both on the hardware and the algorithmic side, we need to explore novel and innovative ways of how each sub-module of an autonomous system's architecture (e.g. perception, planning, and control) can contribute to safer driving in the future. Traditionally autonomous systems rely on

[1]`https://www.who.int/gho/road_safety/mortality/number_text/en/`

[2]`https://www.statista.com/statistics/191660/fatality-rate-per-100000-licensed-drivers-in-the-us-since-1988/`

[3]`https://visionzeronetwork.org/about/what-is-vision-zero/`

[4]`https://www.itf-oecd.org/sites/default/files/docs/08targetssummary.pdf`

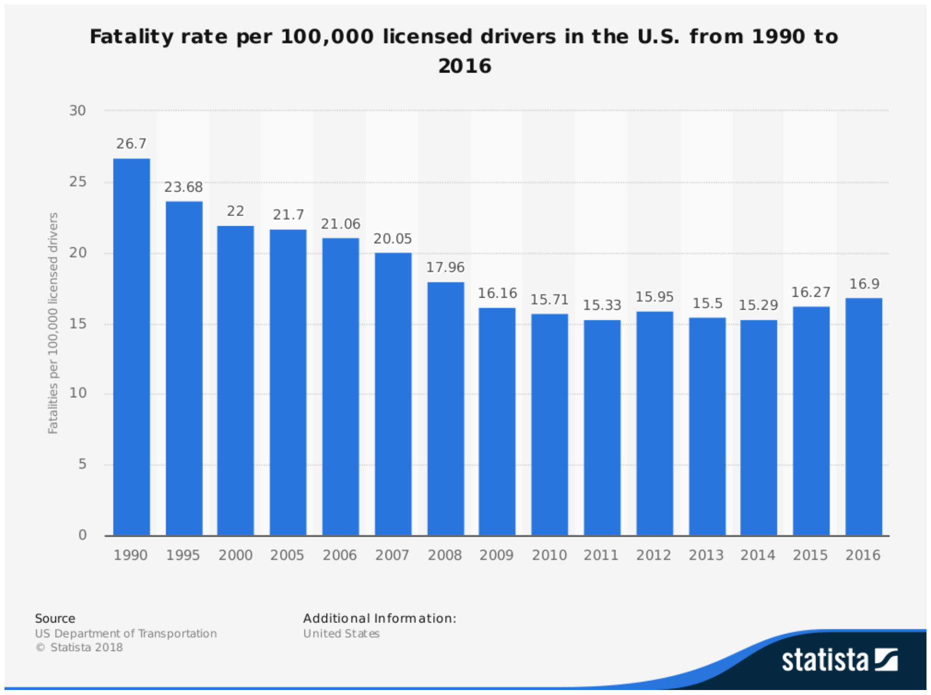


Figure 1-1: Fatality Rate in the USA. "The timeline shows the fatality rate per 100,000 drivers licensed to operate a motor vehicle in the United States from 1990 to 2016. The fatality rate stood at 16.9 deaths per 100,000 licensed drivers in 2016"[2].

a perception module to sense the environment and localize, on a planning module to compute target actions and a control module to actuate the vehicle in order to execute the target actions. On the perception side increasing safety could mean to develop more accurate, robust and weather invariant sensors. It could also mean to use existing sensors in new ways and to exploit a new signal range which could be used for obstacle detection or early collision warning.

1.1 Motivation

The high-level idea is to improve safety by increasing the situational awareness of the human driver or the autonomous car. Perception modules consisting out of sensors and algorithms interpreting the sensor's data help to identify obstacles. Conceptually speaking, increasing the perception horizon (i.e. decreasing the blind spot areas) would allow earlier detections and thus give more time to prevent collisions. Specifically, we aim to detect unexpected dynamic obstacles out of the direct line of sight from the viewpoint of the moving ego vehicle based on shadows. This would help to detect moving persons behind buildings or parked cars.

But current sensor solutions (e.g. LiDAR, RADAR, Ultrasonic, Cameras, etc.) and algorithms widely used in ADAS applications require direct sight of dynamic obstacles for detection and/or classification. Some methods can handle partial occlusion of objects but anticipating collisions with unseen obstacles and thus avoiding dangerous situations have so far been impossible.

Thus, it is a key unsolved problem to detect obstacles even before they are directly visible. Such obstacles could e.g. be pedestrians running on the street while hidden due to parked cars from the viewpoint of an approaching car (Fig. 1-2).

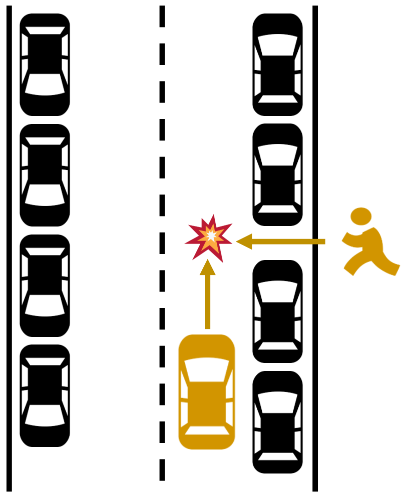


Figure 1-2: ShadowCam use case. A pedestrian runs on the street while hidden from the viewpoint of the approaching car.

The results of this thesis are providing evidence that the proposed algorithms could ultimately help to make driving safer for pedestrians as well as drivers and safe lives.

Observing human drivers' and operators' behavior gives an intuitive idea of how a solution for this problem could look like. In certain situations, humans can perceive obstacles even when they are completely occluded.

One technique used by humans consciously or even sub-consciously for detecting hidden dynamic obstacles is observing changes in illuminance. Via the change in illuminance humans can infer an approaching person or vehicle around a corner or anticipating a car backing out of a driveway. A pronounced and easily observable cue is the illumination change generated by car-lights at night, which is used by humans to anticipate an approaching vehicle e.g. at intersections. Inferring motion of obstacles around corners becomes considerably more challenging during daytime or while operating a mobile robotic platform in a well-lit indoor environment. We explore whether in these scenarios, shadows can be used as a cue, providing information on e.g. whether a potential obstacle behind a corner is in motion. Use of shadows in obstacle detection systems is particularly challenging as it requires motion detection despite a sometimes barely visible shadow (i.e. low signal-to-noise ratio (SNR)).

1.2 Problem Setup

To approach this new problem, we are looking at indoor corners during different times of the day and at corners where it is physically possible to cast a shadow. We aim to classify video sequences into "dynamic" or "static" depending on whether or not someone is moving behind the corner from the viewpoint of a relatively slow-moving autonomous vehicle platform (ca. around 3mph).

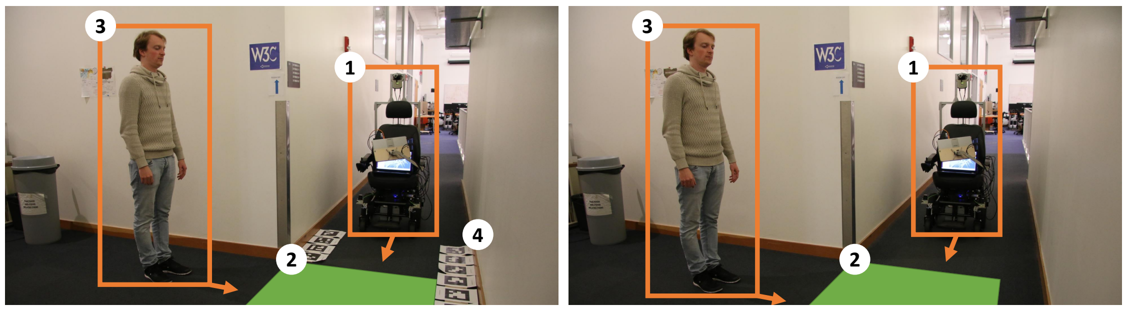


Figure 1-3: Problem Setup. We compare the performance of two image registration methods as part of the ShadowCam pipeline: On the left side we place visual fiducial markers (i.e. AprilTags) on the ground plane and on the right we use a visual odometry method (i.e. DSO) for image registration.

To focus on the algorithmic development, we first registered the video sequences with visual fiducial markers (i.e. AprilTags) on the ground plane. The second approach relies on a visual odometry method (i.e. DSO) to register the sequences into the same coordinate system (Fig. 1-3). This increases the generalizability of our method since we can run the ShadowCam algorithm on any corner without placing AprilTags markers on the ground plane beforehand. In Fig. 1-3 number (1) marks the autonomous wheelchair, (2) the known Region of Interest (ROI) where a shadow is expected to be detected and (3) the dynamic obstacle out of the line of sight. (4) are the visual fiducial markers (i.e. AprilTags) placed on the ground plane.

1.3 Key Contributions

This thesis presents a novel method and algorithm for using **shadows as features** observed with a passive sensor (e.g. camera) to avoid collisions with unseen dynamic obstacles. The proposed algorithm could ultimately help to save lives. Using a shadow as a feature is a recent concept in image processing where the focus lies usually on the removal of the shadow and the corresponding signal gets treated as unwanted noise. Inferring useful information from a shadow or from a small illumination change on the ground is a challenging problem since it is usually a weak signal (i.e. low SNR). Such a weak signal becomes even harder to observe from a **moving platform** as this introduces more noise to the system.

We assume that we operate the system indoors at corners where it is physically possible to cast a shadow for a moving obstacle, the ROI is known as described in Sec. 1.2 and given speeds at around 3mph for both the obstacle and the ego vehicle. Creating a solution for the described problem under these assumptions includes the following key contributions presented in this thesis:

- A novel method for shadow-based motion detection of dynamic obstacles occluded behind corners.
- Extensive evaluations on synthetic data and recordings of real-world corner experiments.
- Implementation of our algorithm as an open-source Robotic Operating System (ROS) package, as well as publication of a comprehensive dataset of evaluation scenes.

- Integration of all code to run in real-time on a full-scale autonomous wheelchair.
- Comparison of the classification performance of our algorithm using visual fiducial tags (i.e. AprilTags (AT)) and a visual odometry (i.e. Direct Sparse Odometry (DSO)) as the image registration methods.

1.4 Outline

The next chapter gives an overview of the related works (Chap. 2).

Chap. 3 presents the technical approach to solve the described problem and gives more details about the two used image registration methods in Sec. 3.2 and Sec. 3.3.

The chapter about the experimental setup (Chap. 4) introduces all platforms ranging from simulation (Sec. 4.2) to the real world experiments (Sec. 4.1 and 4.3) with an autonomous wheelchair (Sec. 4.4). It closes with the description of our annotation tool (Sec. 4.5).

In the result chapter (Chap. 5) we present how the ShadowCam algorithm performs in the experimental settings from Chap. 4 and we close with the conclusions in Chap. 6 and future work outlook in Sec. 6.2.

Chapter 2

Related Work

There have been several works on perception for mobile robotics in non-line-of-sight (NLoS) scenarios. Most research in that context considers ultra-wideband (UWB) systems for localization, as e.g. in [36], and ranges up to using WiFi signals for NLoS perception [1] (Fig. 2-1). A first approach for explicitly seeing around corners for mobile robotics was presented in [44] using a drone which can be launched from a car as an additional source of information (Fig. 2-1). In contrast, the method of this thesis relies on a vision-based approach and does not require hardware infrastructure, assumptions about the occluding material, or deployment of drones.

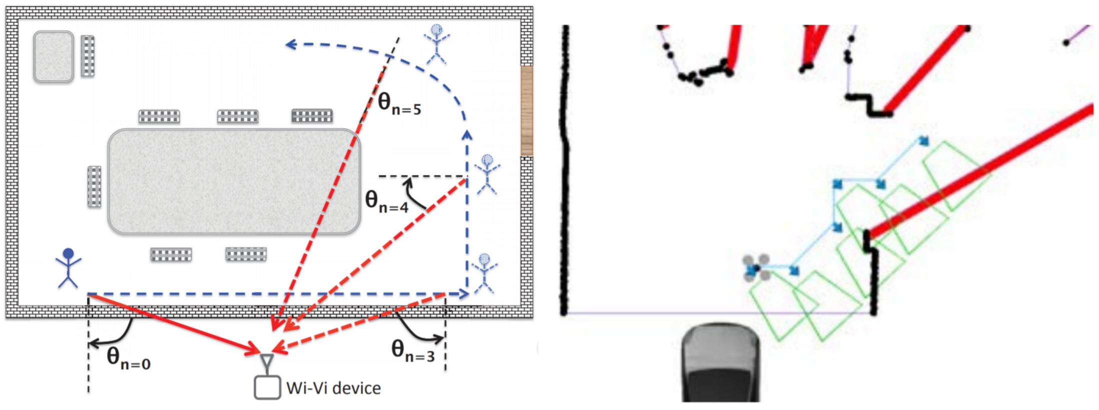


Figure 2-1: NLoS Scenarios. The image on the left “shows the experimental setup of a trial which consisted of a single person moving around in a conference room” [1]. The image on the right “shows the initial plan for the quadcopter to view some of these blind regions” [44].

2.1 Handling Object Occlusion

Consideration of occlusion for intelligent transportation systems mostly focused on improved tracking by improving detectors of other vehicles [27, 13] and pedestrians [29] while assuming partial visibility or a merely a temporary occlusion. In [9], explicit modelling of occlusions was also used for broader scene understanding. In contrast to these approaches, we do not assume even partial visibility of the potential obstacle but use, when available, a shadow instead.

2.2 Shadow Processing

In many fundamental computer vision tasks shadows could cause problems. “For instance, shadows can deteriorate the performance of object recognition, stereo, shape reconstruction, image segmentation and scene analysis” [21]. Therefore shadow processing typically focuses on its removal [21, 32, 12, 2, 35] in single images and in video sequences. “In digital photography, information about shadows and their removal can help to improve the visual quality of photographs. Shadows are also a serious concern for aerial imaging and object tracking in video sequences” [21] (Fig. 2-2).

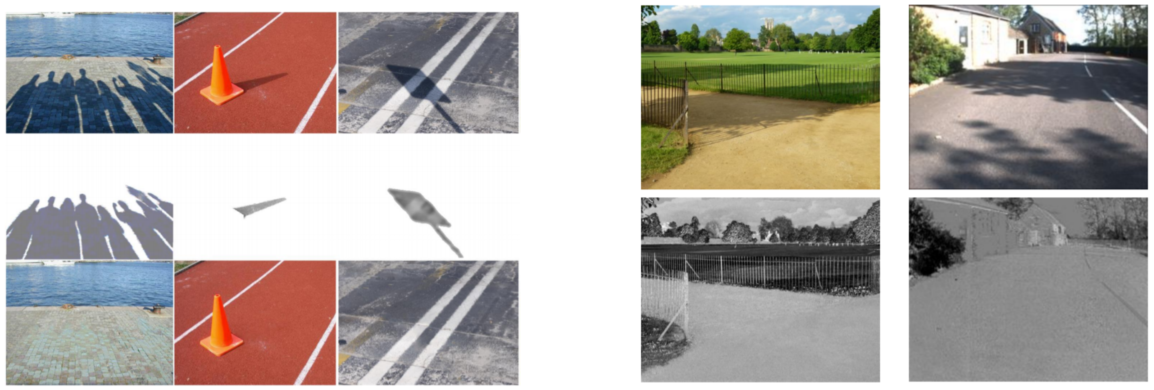


Figure 2-2: Shadow Removal. On the left: “Given an original image with a shadow mask (first row), the method is able to extract exact shadows (second row) and to automatically recover the shadow-less images (third row)” [21]. On the right: On the top images with shadows are shown and below the corresponding shadow invariant image [8].

Another important example where shadow removal is helpful are vision-based localization systems. They “rely on place models based on scene appearance recorded as an image. However, image formation is an interplay of both scene structure and the current lighting conditions. Ideally the same place would always produce the same image, but observations are strongly influenced by illumination” [8]. Therefore in mobile robotic applications, shadow removal is particularly relevant for improving visual localization, because it enables generating a more weather invariant representation of the environment [8, 24] (Fig. 2-2).

In [23] shadows are also used in motion detection, that work assumes a different scenario involving a static camera and also considers visibility of the tracked object. In contrast to these works, we explicitly use shadows as cues in our system.

2.3 Hidden Scene Recovery

Computer vision approaches which infer about hidden scenery usually rely on Time-of-flight cameras [30, 37, 38, 22, 20, 14]. Time-of-flight cameras are prone to interference from other unpredictable lighting sources and therefore mostly rely on carefully controlled environments (Fig. 2-3).

Using an active sensing approach, as described in [43], “a laser pulse that lasts less than one trillionth of a second is used as a flash and the light returning from the scene is collected by a camera at the equivalent of close to 1 trillion frames per second. Because of this high speed, the camera is aware of the time it takes for the light to travel through the scene. This information is then used to reconstruct shape of objects that are visible from the position of the wall, but not from the laser or camera”[1].

It was recently shown that lighting from behind the corner and the created faint penumbra on the ground can be used for creation of a 1D video [3] from a static camera in pure passive sensing setting. The stylized diagram (Fig. 2-3) “shows a typical scenario: two people – one wearing red and the other blue – are hidden from the camera’s view by a wall. Only the region shaded in yellow is visible to the camera.

[1]`http://web.media.mit.edu/~raskar/cornar/`

To an observer walking around the occluding edge (along the magenta arrow), light from different parts of the hidden scene becomes visible at different angles" [3].

Drawing inspiration from this work, our proposed approach considers shadows and uses them for motion detection from a **moving platform**. This contrasts with most perception systems which explicitly consider shadows so far, since they mostly focus on its removal. The underlying intuitive idea here is that shadows change the light intensity of pixels from the ground plane close to a corner when it is physically possible to cast a shadow. On a registered and stabilized image sequence such a change of pixel values can be observed with a filter. A decision module can then be based on a threshold. This allows an autonomous vehicle to interact with (e.g. stop for) dynamic obstacles out of the line of sight and increase safety.

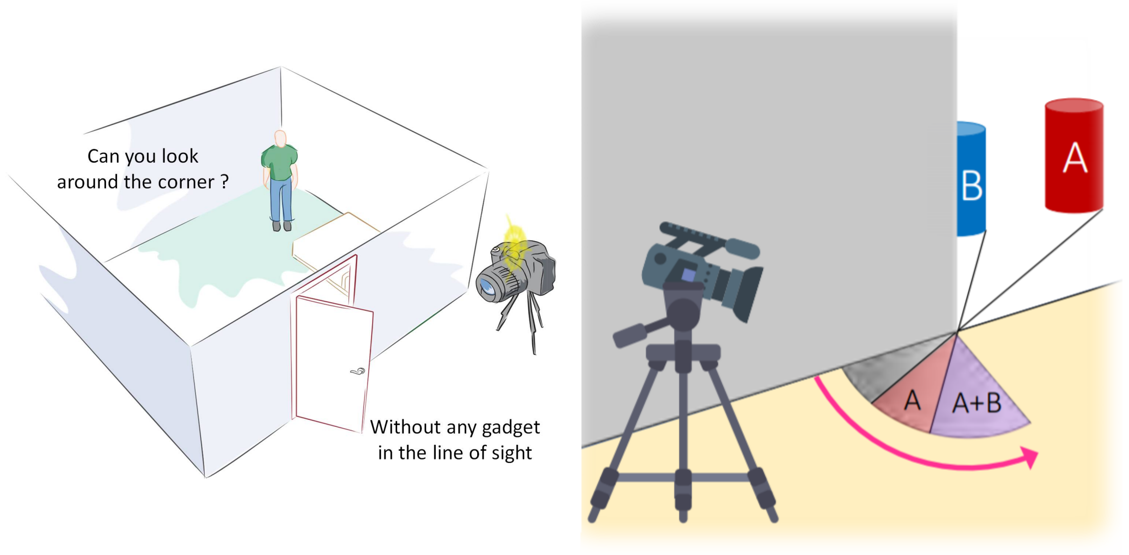


Figure 2-3: Hidden Scene Recovery. On the left: "The seemingly impossible task of recording what is beyond the line of sight is made feasible by ultra-fast imaging. A new form of photography, Femto-photography, exploits the finite speed of light and analyzes "echoes of light""[1] [43]. On the right: To construct a 1-D video of an obscured scene an RGB video is recorded with a consumer camera [3].

2.4 Change Detection

Change detection is usually the problem of identifying gradual or sudden temporal changes between two frames of a video. Change detection is often done in the context of organizing and sequencing videos for the television industry. “Shot-change detection is the process of identifying changes in the scene content of a video sequence so that alternate representations may be derived for the purposes of browsing and retrieval, e.g. key-frames may be extracted from a distinct shot to represent it” [41]. According to [41], there are three major approaches for this problem.

The first approach is using color histograms and compares the color compositions of two frames. Any difference in the histograms that exceeds a certain threshold indicates that a change occurred between the two images. Experiments with this method [41] show that generally, average colors of frames not only ignore luminance but are ultimately not rich enough features to capture change or motion.

As further detailed in [5, 41, 31], the second approach is to detect and model the motion of objects in between two frames. This can be done by either estimating the optical flow of the object in the frames or by estimating the physical, real-world motion of the object. Estimating the optical flow of the object relies on comparing pixels between frames, whereas estimating the real motion of the object involves developing a two-dimensional affine motion. Both these approaches are sensitive to noise.

Thirdly and lastly, [41, 18] discuss change detection using MPEG compressed data. Compressing data into MPEG or MPEG-2 formats allows one to extract higher level semantic information from the videos' hyper parameters that allow to infer information about change between images.

2.5 Action Recognition

With the recent success of neural networks and deep learning, a whole host of new techniques came about that made solving the problem of action recognition a feasible task. Action recognition is usually the task of analyzing not only the contents of a video, but also how this content changes over time. Since 2010, a number of convolutional neural networks-based approaches have been developed. We will discuss some of more pertinent ones of those described in [33] below (Fig. 2-4).

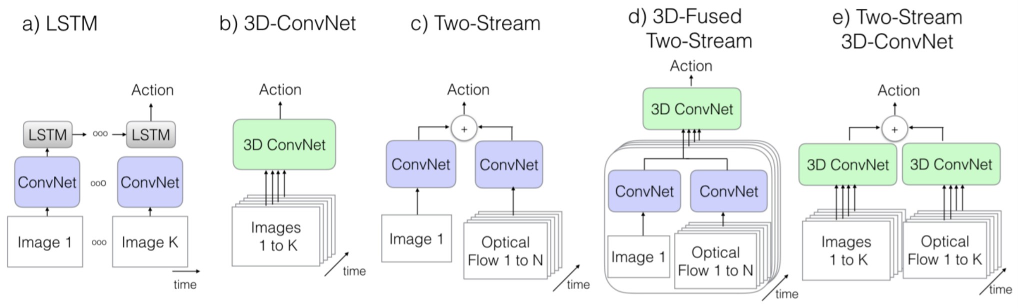


Figure 2-4: Action Recognition approaches. K represents the number of frames in a video and N represents the N neighboring frames in a video [6].

The earliest approach is discussed in [16] and a more generalized version of [16] is presented in [17]. This approach involves a single stream of 2-dimensional recurrent convolutions on multiple clips sampled from a video. The average of the predictions on each clip is taken to be the prediction for the whole video. This method failed to classify motion features of objects in the video as it was unable to capture the temporal aspects of an action well. Additionally, the sampling of clips led to false label assignments to each clip.

Evolving from [17], [7] and [40] attempt to solve [17]'s problem of failing to capture temporal features by introducing a two-stream architecture of 2-dimensional convolutions, where one stream is used for spatial features and the other for temporal features. The two streams are then fused together in the last convolutional layer either using a SVM or a 3-dimensional convolutional layer. This performed slightly better than [17].

The next approach, as introduced by [39] and further developed by [10], is similar to [17]. Instead of 2-dimensional convolutions, the authors utilized 3-dimensional convolutions. The proposed architecture is based on 3 by 3 by 3 convolution kernels at each layer performed quite well while maintaining compact features. This approach captured long-term temporal features quite well despite struggling to classify more complex actions.

Finally, [6] reinvents the work done by [7] and inflates the state-of-the-art 2-dimensional CNNs trained on ImageNet such that the two streams use 3-dimensional convolutions instead of 2-dimensional ones. These two new streams are then fused together using a SVM or another linear classifier. This approach has achieved very good precision and recall metrics for action recognition.

Chapter 3

Technical Approach

The problem is about detecting dynamic obstacles out of the direct line of sight of a moving ego vehicle based on shadows. Conceptually we aim to increase safety by increasing the situational awareness of the human driver when ShadowCam is used an additional ADAS or of the autonomous vehicle when ShadowCam is used as an additional perception module. In this chapter we highlight the specific challenges of this problem and explain our technical approach to address these. The core module of the ShadowCam pipeline is the classification of frame sequences (Alg. 4) which enables the human driver or the autonomous vehicle to avoid potential collision with dynamic obstacles out of the direct line of sight.

It is challenging to come up with a general solution for the described NLoS problem. At some corners moving objects are physically not able to cast a shadow, at others and in most cases the shadow is a low SNR signal. This signal highly depends on various nuisance factors — these include size of the object, speed of the movement, lighting, reflection properties of the floor, color of the floor, ego motion, among others. Despite these difficulties (as shown in [3]) it is possible to create a signal of a moving obstacle behind a corner from a static camera. But to actually make use of the shadow signal in a practical way, e.g. as a safety feature for autonomous vehicles, it needs to work on a moving platform. This requirement (1) adds even more noise to the system and thus makes it harder to achieve a reliable detection accuracy and (2) the implementation of the algorithm needs to be real-time capable.

We are presenting a motion detection algorithm based on shadows as features from the viewpoint of a moving vehicle. We look at corners where moving objects are physically able to cast a shadow. The algorithm runs on a cyclic buffer and in a pre-processing step projects all images of the buffer to the same viewpoint (Sec. 3.1). On these registered image sequences, we run the ShadowCam algorithm to detect dynamic obstacles (Fig. 3-1). For the registration step we compare two techniques:

- Visual fiducial markers (i.e. AprilTags) placed on the ground plane (Sec. 3.2)
- Visual odometry method (i.e. DSO) to get the rotation matrix and the translation vector between each frame for the projection into the same coordinate system (Sec. 3.3)

The ROI could be determined using the map that the autonomous wheelchair uses to localize itself, other place recognition algorithms [42], or a deep-learning-based detector, but determining the ROI is not the focus of this work. Instead we use hand annotations for each corner to crop the ROI where we expect to see a shadow.

Fig. 3-1 and Alg. 3 show how we embedded ShadowCam in the perception, planning and control cycle of the autonomous vehicle, which is in our case a wheelchair. The ShadowCam pipeline consists out of five steps. First, we run a cyclic buffer with the image stream from the camera. Then we run two different image registration methods (either based on AprilTags (Sec. 3.2) or on DSO (Sec. 3.3)) on this buffer. In Sec. 3.1 we introduce more details about the image registration process. This step also includes the ROI selection based on the annotations. The output of the second step is a registered buffer (i.e. frame sequence) with ROI selection. During the third step (i.e. pre-processing step) we compute the mean image of the current sequence, resize and amplify the signal. The output of the third step is a frame sequence which has the same size for both image registration method. This allows us to interchange the image registration methods seamlessly. The classification algorithm of the ShadowCam pipeline (Alg. 4) in the fourth step decides based on the pre-processed image sequence whether it is safe to continue along the path. The vehicle interface in the fifth and last step then executes this decision.

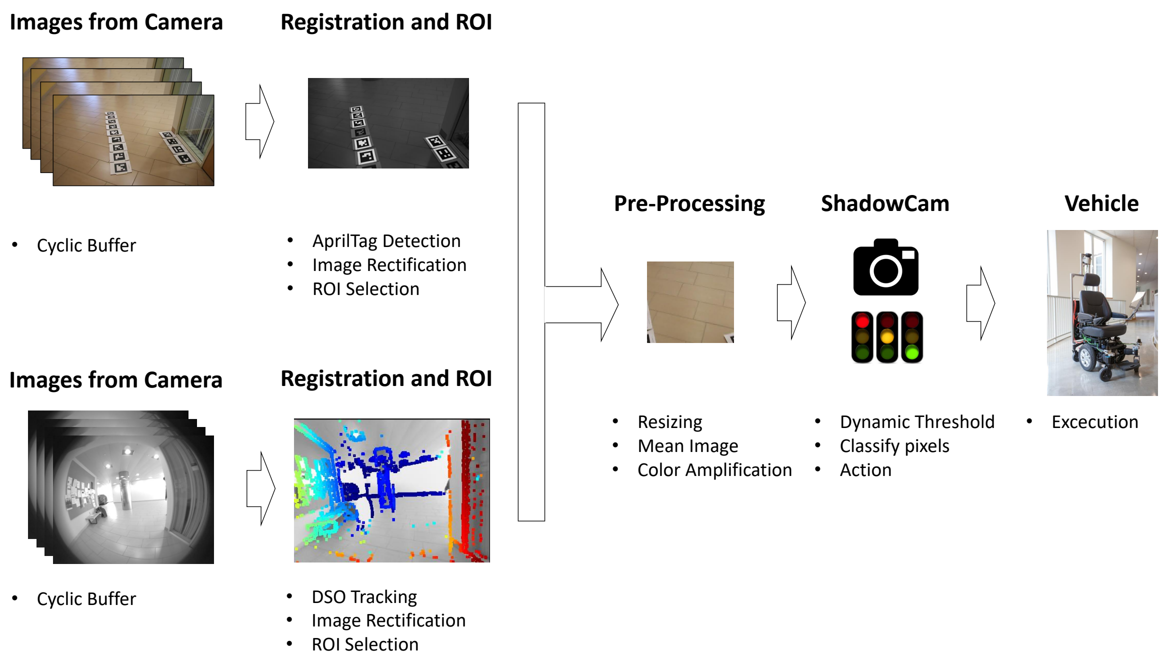


Figure 3-1: Overview of the ShadowCam Pipeline (Alg. 3). We run a cyclic buffer over the frames of the camera mounted on top of the autonomous vehicle. After the pre-processing steps – including image registration and ROI selection (Sec. 3.2 and 3.3) – we classify sequences (Alg. 4) and ensure that the vehicle avoids collisions with unseen obstacles based on shadows.

3.1 Image Registration

In the literature image registration is usually referencing to the process of transforming multiple images into the same coordinate system. This process can be split into four steps [47]:

- Feature detection (e.g. Oriented FAST and rotated BRIEF (ORB), scale-invariant feature transform (SIFT) or speeded up robust features (SURF)-features)
- Feature matching
- Estimating the homography based on the matched feature points
- Resampling and transformation of the image with an appropriate interpolation technique

In this case (Eq. 3.1) the homography H transforms points of two planes (up to a scale-factor s) with 8 Degrees of Freedom (DOF):

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.1}$$

Fig. 3-2 visualizes a planar surface from the view point of two cameras and shows how H projects *image 1* to the coordinate frame of *image 2*. This allows to overlay two or more images from the same environment but shot from different angles. We show in the following the two methods we chose for image registration.

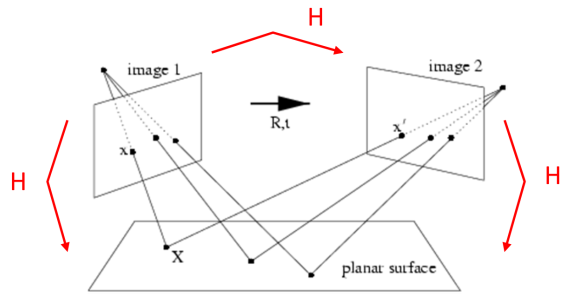


Figure 3-2: Planar World. Rotation and Translation of Camera Frames [15].

3.2 Visual Fiducial System: AprilTags

To be able to focus on the algorithmic development in a first iteration we sought for a straight forward way to register images reliably. Placing visual fiducial markers on the ground plane was the favorable option.

We use AprilTags [28, 45] to provide features for sequence registration. April-Tags are a visual fiducial system. The tags can be created from a normal printer, and the open-source “AprilTag detection software computes the precise 3D position, orientation, and identity of the tags relative to the camera”[1]. The open-source implementation is real-time capable. We placed 13 AprilTags on the same plane at which

[1]`https://april.eecs.umich.edu/software/apriltag`

we expect to detect a shadow. In theory one tag would be enough, but most of the time only a subset of the AprilTags gets accurately detected and by adding more tags we increase the numerical stability. The unique IDs of the tags can be used for cropping the ROI where we expect to see a shadow. We take the ID of the furthest tag from the ego vehicle and crop a dynamic rectangle using the size of that tag, which results in a rectangle that is larger the closer we are to the corner.

As described in [45] the detection steps are (1) binarizing the input image with an adaptive threshold (2) segmentation based on connections between black and white regions (3) fitting of squads to cluster of border pixels (4) outputting valid tag detections.

Relying only on the AprilTags on the ground plane as features for the image registration process has two major advantages:

- Feature matching is fast since every tag has a unique ID
- Since all tags are placed on the ground plane, we only consider matched points on the ground plane to compute the homography

ORB or SIFT features would not only be on the ground plane but all over the scene and thus would have a possibly negative impact on the resulting homography. Solving for the homography in this case would mean to get a good overall transformation result, but in our case, we care mostly about an accurate homography for the ground plane where we expect to detect a shadow.

Algorithm 1 gives an overview of the sequence stabilization process with AprilTags. For all frames in the cyclic-buffer we find the maximum set of commonly detected tags (step 3) and compute homographies (step 5) based on the matched points (step 4). This homography then transforms all frames f_i in the buffer to the view point of the first camera frame (transformation from c_i to c_0 in step 6).

Algorithm 1 AprilTag Image Registration Algorithm

```
1: $d_0 \leftarrow$ tagDetection(0)
2: for all i=1; i< buffer.length; i++ do
3:     $d_i \leftarrow$ tagDetection(i)
4:     $m_i \leftarrow$ findMatchingPoints()
5:     $H_{c_i}^{c_0} \leftarrow$ computeHomography()
6:     $f_0 \leftarrow$ warpPerspective($f_i$)
```

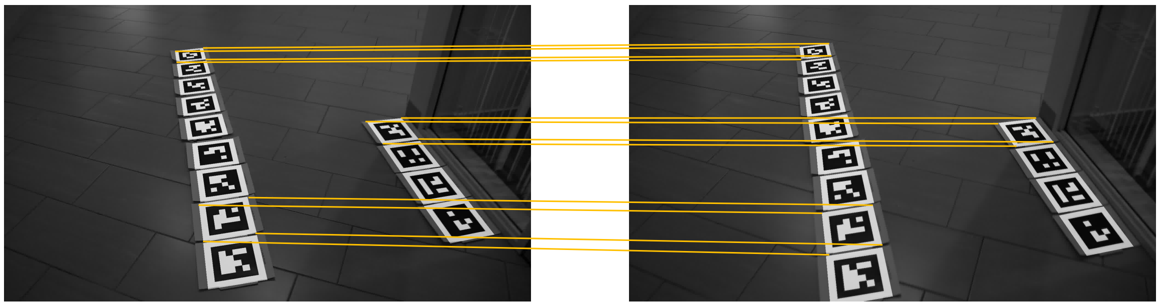


Figure 3-3: AprilTag Matches. Example matches of the AprilTags on the ground plane where the image on the right is closer to the corner. Each tag has a unique ID which allows finding corresponding points fast and easily.

3.3 Visual Odometry Method: DSO

Many different visual odometry methods have been developed for the past 15 years. Various methods found their way into wide ranging applications in robotics and augmented reality. On a higher level the literature separates this line of work based on the data association design choice [46] (Fig. 3-4). Some of the most recent examples are shown in Fig. 3-5. Our choice for DSO is mainly driven by two requirements:

- The code is open-source, works and real-time capable (i.e. ca. 20Hz)
- The visual odometry method should also perform reliably in hallways and areas where only very few textural features exist

Specifically, we looked at the open-source implementations of ORB-SLAM [25] and DSO. But since ORB SLAM is a feature-based method it works better in feature richer environments. We run our experiments mainly in hallways without many textural features. In theory DSO performs in this setting more reliably. DSO is a sparse and direct method for monocular visual odometry. It "jointly optimizes full likelihood for all involved model parameters, including camera poses, camera intrinsics, and geometry parameters (inverse depth values)" [11]. We tried the open-source code of DSO and ORB-SLAM. We ran initial tests which could confirm that DSO performs better in our experiment settings.

After this initial review and evaluation, we moved forward with DSO and modified the open-source code so that it integrates seamlessly the ShadowCam pre-processing pipeline.

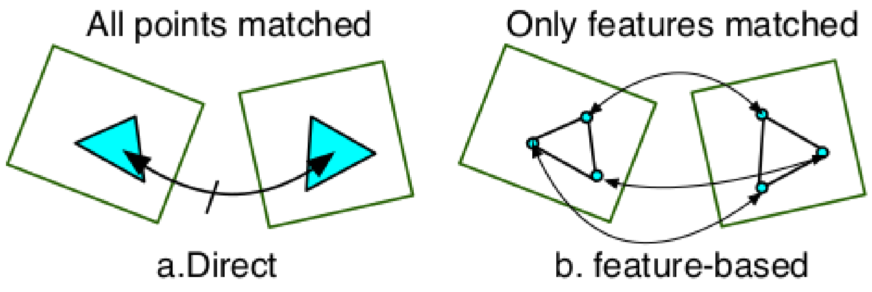
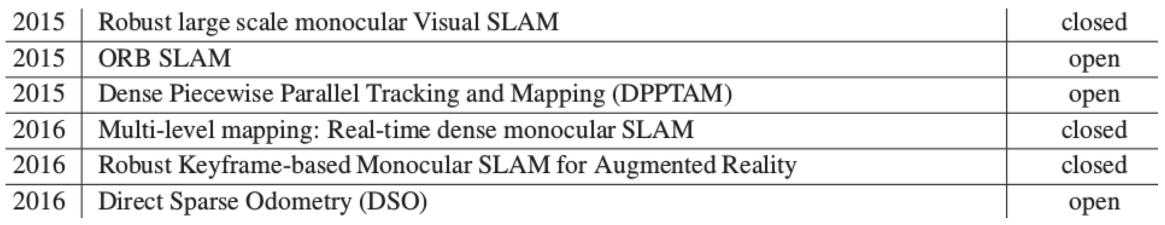


Figure 3-4: Design choices for data association. Direct (a) vs. feature based (b) methods [46].



2015	Robust large scale monocular Visual SLAM	closed
2015	ORB SLAM	open
2015	Dense Piecewise Parallel Tracking and Mapping (DPPTAM)	open
2016	Multi-level mapping: Real-time dense monocular SLAM	closed
2016	Robust Keyframe-based Monocular SLAM for Augmented Reality	closed
2016	Direct Sparse Odometry (DSO)	open

Figure 3-5: Visual SLAM Methods. Recent keyframe-based visual SLAM system with either open or closed source code [46]. Column 1: Publication year. Column 2: Name of the method. Column 3: Open our closed source code.

The open-source implementation of DSO[2] computes the position for each frame (M_w^c) which consists out of the rotation matrix R and the translation vector t. We adapted the source code and integrated it into our pre-processing pipeline required to run ShadowCam algorithm.

[2]`https://github.com/jakobengel/dso`

In the following we describe how we obtain the homography H mathematically from R and t. As indicated in Fig. 3-2 the homography is proportional to the information given by the *planar surface* equation, the rotation matrix R and the translation vector t between two image frames

$$H \propto R - tn^T \tag{3.2}$$

where n designates the normal of the local planar approximation of the scene [34]. Symbols used in this section are described in Table 3.1.

Algorithm 2 gives an overview of how the following equations are connected to get H from R and t for each frame. We obtain R and t for the first frame in the buffer and register all following frames with respect to the first frame (in Alg. 2 denoted as c_2). This essentially means that all frames are projected into the same coordinate system.

We annotate three points on the ground plane and in the world frame w. This results in reasonable transformations for most pixels on the ground plane. This is important since later we want to classify shadows close to a corner on this plane.

After we obtain R and t of frame f_i we can transform the points on the plane and in the world frame w to the camera frame c_1. M_w^c in homogeneous form[3] transforms points from the world frame (denoted as w) into the camera frame (denoted as c):

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = M_w^c \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} R_w^c & t_w^c \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.3}$$

Given K the camera's intrinsic matrix and M_w^c the camera's pose we can obtain the

[3]`https://docs.opencv.org/3.4.1/d9/dab/tutorial_homography.html`

Algorithm 2 DSO Image Registration Algorithm

```
1: $planePoints_w \leftarrow parametersFromFile()$
2: $R_{c_2} \leftarrow getRotationMatrix(0)$        ▷ Rotation matrix of first frame in cyclic buffer
3: $t_{c_2} \leftarrow getTranslationVector(0)$  ▷ Translation vector of first frame in cyclic buffer
4: for all i=1; i< buffer.length; i++ do
5:     $R_{c_1} \leftarrow getRotationMatrix(i)$
6:     $t_{c_1} \leftarrow getTranslationVector(i)$
7:     $planePoints_{c_1} \leftarrow$ Eq. 3.3          ▷ Transformation of world plane points to $c_1$
8:     $R_{c_1}^{c_2} \leftarrow$ Eq. 3.6                 ▷ Obtaining rotation matrix from $c_1$ to $c_2$
9:     $t_{c_1}^{c_2} \leftarrow$ Eq. 3.7                 ▷ Obtaining translation vector from $c_1$ to $c_2$
10:    $n_{c_1} \leftarrow computeNormal(planePoints_{c_1})$
11:    $d_{c_1} \leftarrow computeDistance()$
12:    $H_{c_1}^{c_2} \leftarrow$ Eq. 3.8                 ▷ Calculating homography matrix
13:    $f_{c_2} \leftarrow warpPerspective(f_{c_1})$
```

image points directly from world points in the following way:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K\, M_w^c \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.4}$$

With both positions of the camera ($M_w^{c_1}$ and $M_w^{c_2}$, where c_2 is the camera frame of the first image in the cyclic-buffer) we can find the transformation for a 3D point from camera frame c_1 to c_2:

$$M_{c_1}^{c_2} = M_w^{c_2} \cdot (M_w^{c_1})^{-1} = \begin{bmatrix} R_w^{c_2} & t_w^{c_2} \\ 0_{3\times1} & 1 \end{bmatrix} \cdot \begin{bmatrix} (R_w^{c_1})^T & -\,(R_w^{c_1})^T \cdot t_w^{c_1} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{3.5}$$

This allows us to specify the rotation matrix R

$$R_{c_1}^{c_2} = R_w^{c_2} \cdot (R_w^{c_2})^T \tag{3.6}$$

and the translation vector t between two frames

$$t_{c_1}^{c_2} = R_w^{c_2} \cdot \left(- \left(R_w^{c_1}\right)^T \cdot t_w^{c_1} \right) + t_w^{c_2} \tag{3.7}$$

with the distance d as the dot product between the plane normal and a point on the plane. This leads to the homography H from c_1 to c_2

$$H_{c_1}^{c_2} = R_{c_1}^{c_2} - \frac{t_{c_1}^{c_2} \cdot \left(n_{c_1}\right)^T}{d_{c_1}} \tag{3.8}$$

which is the same as Eq. 3.2 including scaling. The implementation of these equations can be found here `https://github.com/fnaser/dso`.

M_c^w	Camera pose, transformation from camera c to world w frame (4x4 matrix)
R_c^w	Rotation matrix, rotation from camera c to world w frame (3x3 matrix)
t_c^w	Translation vector, translation from camera c to world w frame (3x1 matrix)
$H_{c_1}^{c_2}$	Homography matrix, projection from camera c_1 to c_2 frame (3x3 matrix)
K	Camera intrinsics (3x3 matrix)
n_{c_1}	Plane normal in camera frame c_1 (3x1 matrix)
d_{c_1}	Distance between camera c_1 and plane (skalar)

Table 3.1: Symbol Table. Description of variables used in this section.

3.4 Dynamic Threshold

Algorithms 3 and 4 sketch out how we implemented the ShadowCam. For the actual C++ code, we refer to the open-source ROS package `https://github.mit.edu/fnaser/dso_ros`. Algorithm 3 first gives an overview of the main loop. The classification procedure, Algorithm 4, shows how we determine "dynamic" or "static" which is our core algorithmic contribution.

Algorithm 3 ShadowCam Pipeline

```
1:  list ← global var
2:  while true do
3:      f ← getFrame()
4:      r_1 ← ATregistration(f)          ▷ Image registration based on AprilTags (Alg. 1)
5:      or
6:      r_2 ← DSOregistration(f)             ▷ Image registration based on DSO (Alg. 2)
7:      if checkImageRegistration(r) then
8:          s ← createSequence(f, list)                          ▷ Running cyclic buffer
9:          c ← classifySequence(s)                      ▷ Classifying sequence (Alg. 4)
10:         vehicleInterface(c)
```

Our system uses a cyclic frame buffer approach to achieve real-time performance. This means we can output a detection result whenever a new image fulfills the requirements to get appended to the sequence, e.g. we require a maximum number of detected AprilTags above a defined quality level.

Once a new image gets appended to the sequence, we either identify corresponding tags or use DSO to compute a homography H for each frame to get projected to the viewpoint of the first frame in the sequence. In other words, we apply image registration to all frames i in the current sequence j according to

$$f_{j,i}(x,y) = f'_{j,i}(h(x), h(y)) \ . \tag{3.9}$$

After the image registration step, we crop according to the tags the ROI. To reduce

Algorithm 4 Classify Sequence

```
1:  procedure CLASSIFYSEQUENCE($\mathcal{S}$)
2:      $\bar{f} \leftarrow mean(\mathcal{S})$            ▷ Calculating the mean image of the sequence $\mathcal{S}$
3:      $sum \leftarrow 0$            ▷ Initializing the sum over the whole frame sequence
4:      $c \leftarrow 0$            ▷ Initializing the classification output (0 = "static")
5:      for all $f \in \mathcal{S}$ do
6:          $f \leftarrow colorAmplification(\bar{f}, f)$        ▷ Amplifying weak signals (Eq. 3.12)
7:          $f \leftarrow temporalFilter(f)$
8:          $f \leftarrow dynamicThreshold(f)$        ▷ Thresholding pixels based on mean and
    standard deviation (Eq. 3.14)
9:          $f \leftarrow morphologicalFilter(f)$
10:         $sum \leftarrow sum + sumPixels(f)$
11:     if $sum >= camThreshold$ then:
12:         $c \leftarrow 1$            ▷ Classifying the sequence (1 = "dynamic")
13:     return $c$.
```

noise, we down-sample each cropped and registered image using bilinear interpolation to a 100×100 patch,

$$f_{j,i} = \mathrm{resize}(f_{j,i}, (w, h)) \ . \tag{3.10}$$

Then we compute the mean image over all down-sampled images, i.e.,

$$\bar{f}_j = \frac{1}{n} \sum_{i=1}^{n} f_{j,i} \ . \tag{3.11}$$

We subtract the mean image from each frame in the current sequence and apply a Gaussian blur before we amplify the difference to the mean [3]. That is, we compute

$$d_{j,i} = |G\big((f_{j,i} - \bar{f}_j), k, \sigma\big)| \cdot \alpha \tag{3.12}$$

where G is a linear blur filter of size k using isotropic Gaussian kernels with covariance matrix diag(σ^2, σ^2). We chose σ depending on k according to $\sigma = 0.3 \cdot ((k-1) \cdot 0.5 - 1) +$

0.8 as in [4]. We call α the amplification parameter since it amplifies the difference to the mean image. (Based on empirical observations, k has been set to 3 and α has been set to 5 for all experiments.) This process serves as color amplification and helps to improve the detectability of a shadow (sometimes even if the original signal is invisible to the human eye). In other words, this process increases the signal-to-noise ratio. Fig. 3-6 depicts an example of an image before and after the color-amplification process. As it can be observed the shadow is much better detectable after the color amplification. After the frame is color amplified, we run a temporal low-pass filter

$$t_{j,i} = d_{j,i} \cdot t + d_{j,i-1} \cdot (1 - t) \tag{3.13}$$

where $t_{j,i}$ is the filtered result of the difference images $d_{j,i}$. To become more robust against different corners and light conditions, we take inspiration from [19] and apply a "dynamic" threshold. We take the difference from the mean of each channel of the filtered frame as a criterion to determine motion with respect to the standard deviation of the filtered image,

$$c_{j,i} = \begin{cases} 0, & \forall |t_{j,i} - \bar{t}_{j,i}| < w \cdot \sigma(t_{j,i}) \\ 1, & \forall |t_{j,i} - \bar{t}_{j,i}| \geq w \cdot \sigma(t_{j,i}) \end{cases} \tag{3.14}$$

where w is a tune-able parameter that depends on the noise distribution. We set $w = 2$ for all our experiments. The underlying assumption here is that "dynamic" pixels are further away from the mean, since they change more drastically. A combination of dilation and erosion is used to first connect pixels which got classified as motion and then erosion is used to reduce noise. We are applying morphological ellipse elements with two different kernel sizes [4], i.e.,

$$c_{j,i} = \mathrm{dilate}(c_{j,i}, 1)\ , \qquad\qquad c_{j,i} = \mathrm{erode}(c_{j,i}, 3)\ .$$

At the end, we sum up all pixels under the intuitive assumption that more movement in between frames will result in a higher sum

$$s_j = \sum_{i=1}^{n} c_{j,i}(x,y) \ . \tag{3.15}$$

To classify the whole sequence as either "dynamic" or "static" we then apply a camera-specific threshold. We show in Sec. 5 how the threshold can be determined. The data appears to prove what sounds intuitively correct: A less noisy image results in fewer miss-qualified pixels which results in a lower threshold. This implies that a better camera (frame rate and resolution) and a better image registration quality lead to a smaller threshold.

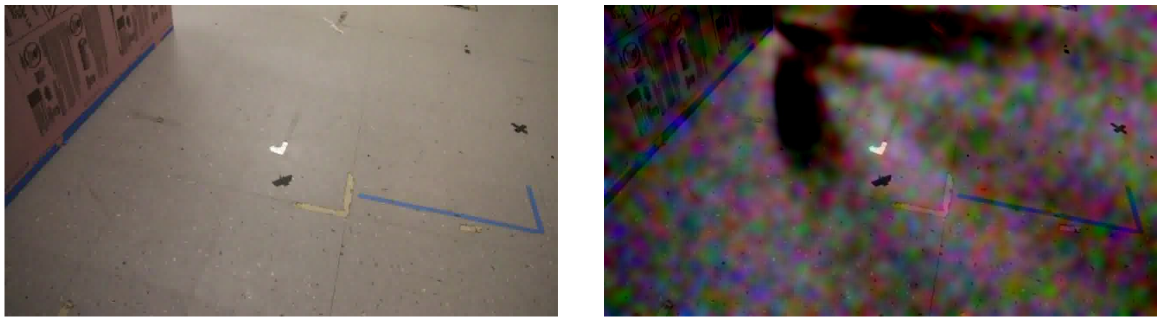


Figure 3-6: Color Amplification. On the left side is an original Canon camera frame and on the right side the corresponding color-amplified frame.

On the vehicle interface side, we run a temporal filter on the detection results to further smooth the signal. Once the ShadowCam detects an obstacle behind the corner the autonomous vehicle stops until it is safe to continue. An example of how this process could look like on real-world data is visualized in Fig. 3-9 and Fig. 3-10.

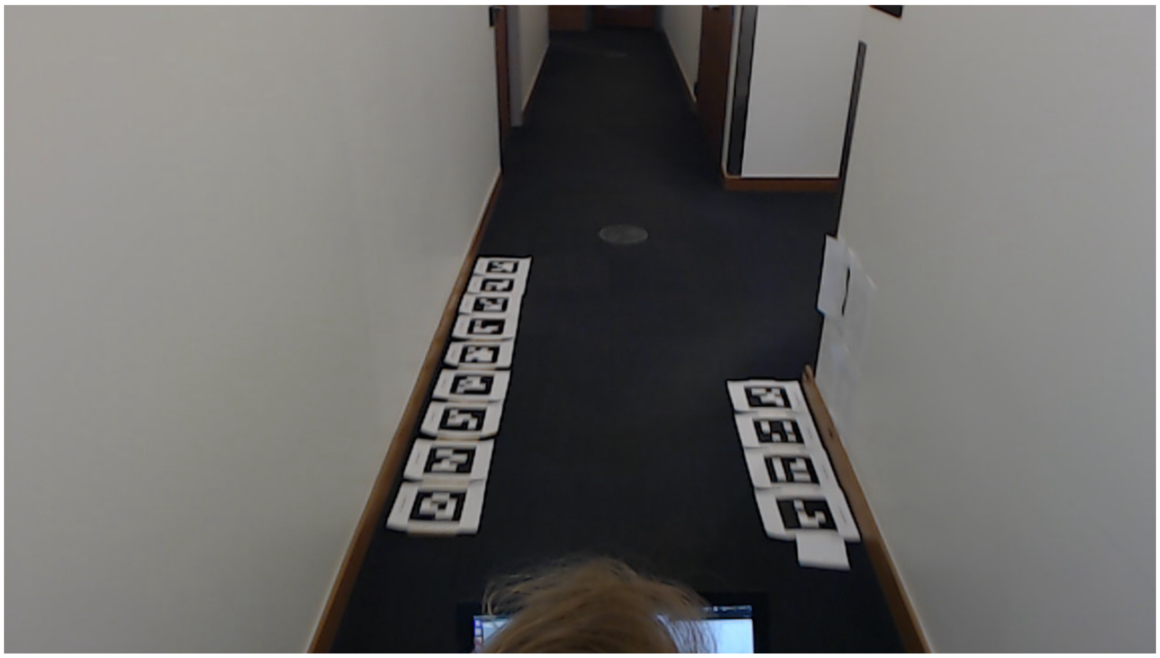


Figure 3-7: Ego-view. The autonomous wheelchair approaches a corner without direct sight of what is going on behind the corner.

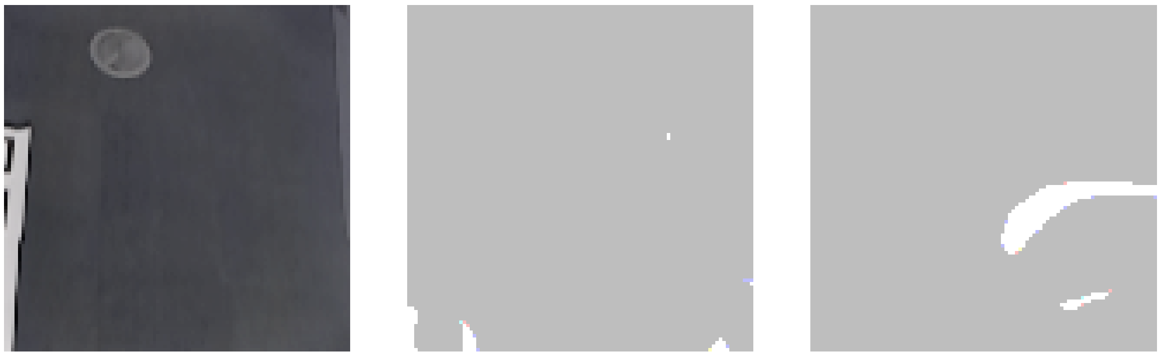


Figure 3-8: Image Analysis. Left: The cropped, re-sized and registered image. Middle: Example of a frame which contains no moving obstacle behind the corner. Right: Example of a frame which contains movement. The white areas correspond to a shadow signal.

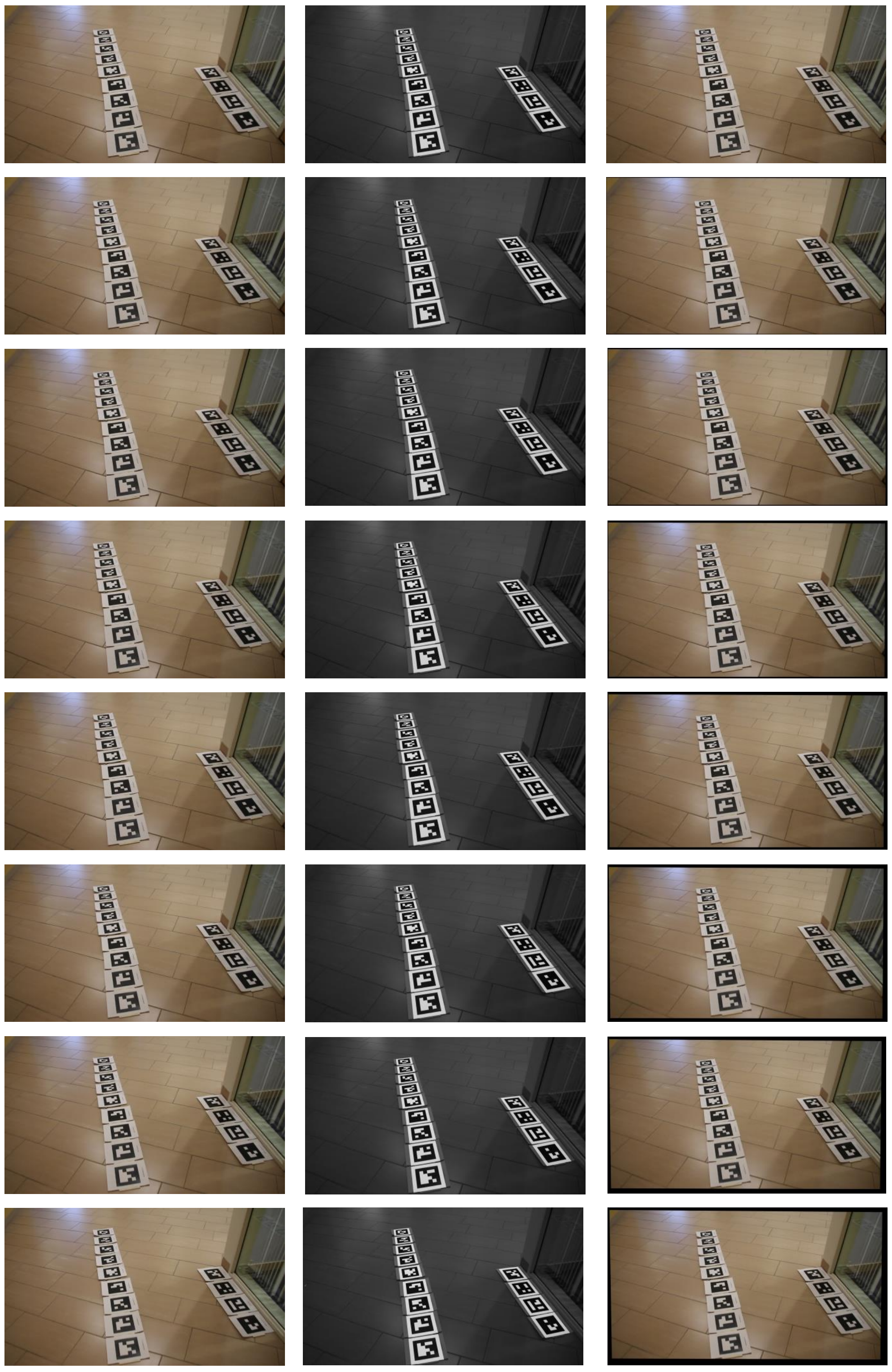


Figure 3-9: Algorithm Steps with AT (1). The first column is representing the input stream of images from the camera in the cyclic buffer. The second column shows which tags got detected by the AprilTag detection software. The third and last column shows how the tag detections were used to register all images to the view point of the first image in the first row.

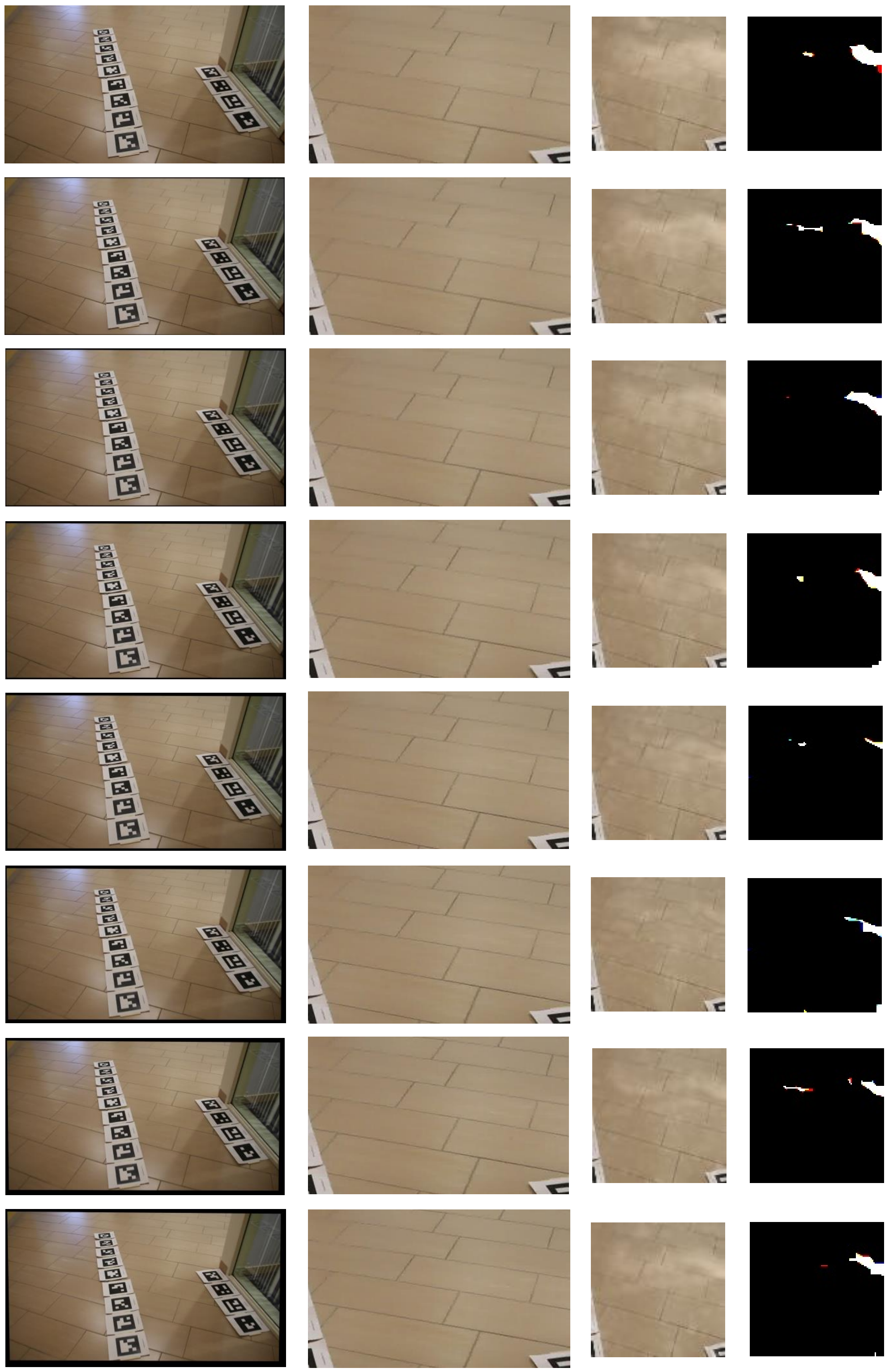


Figure 3-10: Algorithm Steps with AT (2). In the first column we see again the registered images. The second column shows the cropped region of interest. The third column the re-sized and color-amplified version of the cropped images. The fourth and last column displays in which areas the algorithm detected movement based on a dynamic threshold.

3.5 Summary

In this chapter we present our technical approach to tackle the problem of detecting moving obstacles out of the direct line of sight from the view point of the ego vehicle based on shadows. We incorporated two image registration methods in the same pipeline. During pre-processing we amplify the sometimes-weak shadow signal. The decision whether it is safe to move ahead is based on a pixel sum per sequence and a threshold.

Chapter 4

Experimental Setup and Data Collection

This chapter gives an overview under which circumstances and how we collected the dataset to evaluate the technical approach from Chapter 3. Chapter 5 presents then the performance of our technical approach on the dataset we present in this chapter. In general, we want to compare the classification accuracy between AprilTags and no AprilTags and between "dynamic" and "static" sequences. Thus, we are interested in collecting data in the real-world under four main circumstances:

- AprilTags with dynamic obstacle around corner (i.e. "dynamic" sequence)
- AprilTags without dynamic obstacle around corner (i.e. "static" sequence)
- No AprilTags with dynamic obstacle around corner (i.e. "dynamic" sequence)
- No AprilTags without dynamic obstacle around corner (i.e. "static" sequence)

To evaluate the algorithms of the ShadowCam pipeline (Fig. 3-1) in different scenarios and to analyze the performance statistically, we collected data in a motion-capture room (referred to as the Holodeck), created synthetic corners with different properties in a simulation (e.g. Blender[1] and Sec. 4.2), and collected real-world data

[1]`https://www.blender.org/`

using different cameras. We composed the entire dataset to cover a broad range of the mentioned nuisance factors, such as size of the object, speed of the movement, lighting, reflection properties of the floor, color of the floor, ego motion, among others.

The experiments in the Holodeck and in simulation allow us to analyze the spatial performance of the algorithm, since we know the exact position of the moving obstacle behind the corner. Additionally, we can label each sequence based on the ground-truth. For the real-world data collection, we compare two labeling techniques:

- Complete videos are labeled as "dynamic" or "static" depending on whether or not a person was instructed to move behind the corner during video recording. This leads to potentially mislabeled sequences in the complete videos, since the person behind a corner does not always move and the shadow is sometimes not visible on the ground plane close to the corner from the viewpoint of the moving ego vehicle.

- Single frames are labeled with the annotation tool we developed (Sec. 4.5). As will be shown in the results (Chap. 5) this improves the overall accuracy on average.

4.1 Holodeck

The controlled test setup in the Motion Capture environment (i.e. Holodeck) enables labeling of each sequence automatically. There, we collected data with a stationary Canon EOS 70D camera using a EFS $17 - 58$ mm lens. The frame-rate was set to 30 fps, using codec H.264 at a resolution of 1920×1080. The motion capture system tracks the moving object behind the corner at around 100 Hz. We synchronize the video stream[2] with the motion-capture data and label a sequence as "dynamic" when more than half of the frames contain a moving person behind the corner, otherwise it is labeled as "static".

In the Holodeck we were also able to vary a few scene parameters, such as the size

[2]Data synchronization in the Holodeck `https://youtu.be/GCAjBHeh744`

of the person behind the corner, the light conditions produced by switching on and off different ceiling lights and the material of the ground on which the ShadowCam tries to detect shadows. Intuitively, we would expect a better signal when the moving object is closer to the corner, which is confirmed by our spatial analysis of the signal in the Holodeck test setup shown in Fig. 4-1 and 4-2.

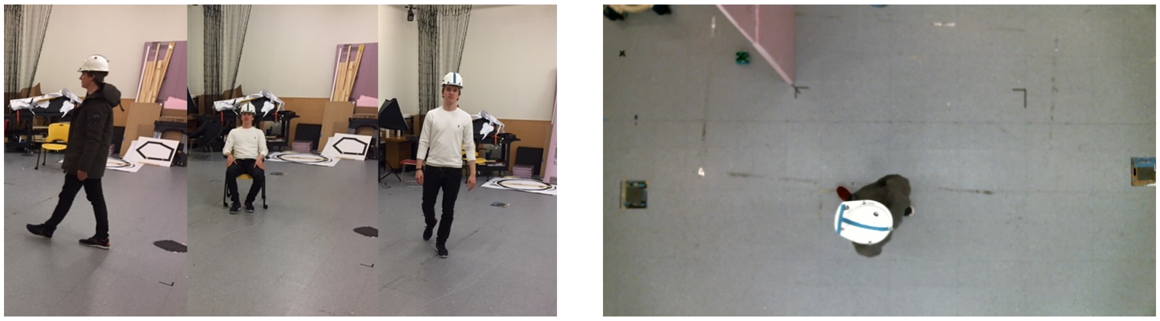


Figure 4-1: Holodeck Setup. The left image shows how we varied the size of the moving object behind the corner: Wearing a winter jacket, sitting on a chair with rollers and walking normally. The right image shows the corner wall and a person walking from the top view.

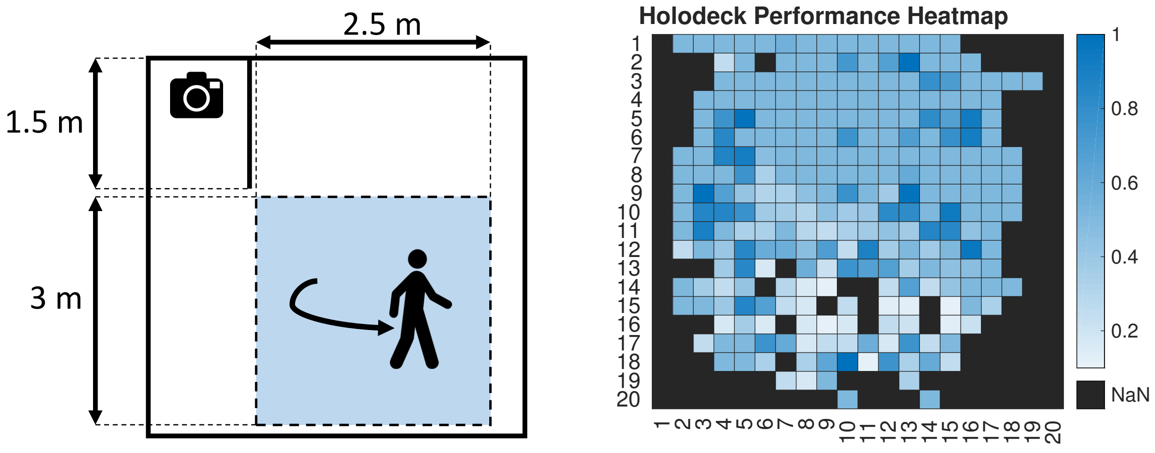


Figure 4-2: Holodeck Performance. The left plot shows the region in which the person behind the corner walked randomly from the top-view. The right plot shows the spatial performance of the algorithm (with regard to the mean position of the moving person in the blue area) to correctly classify a sequence as “dynamic” or “static”. The darker the blue, the higher the classification accuracy for both classes.

4.2 Simulation in Blender

We created a synthetic dataset with Blender, to test the algorithm under a greater variety of lighting conditions, textures, person sizes, and material properties. “Blender is a free and open source 3D creation suite. It supports the entirety of the 3D pipeline-modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation”[3]. With a Python script we access Blender’s API to change the scene parameters dynamically and get ground-truth data for image registration and object motion. We use Blender’s “Cycles Renderer” with only 10 samples to create one frame of size 960×540. This results in more noisy images making it harder for the algorithm to detect a shadow and thus providing more realistic data. In the real-world we are also facing noisy images, e.g. due to motion blur. The chosen area light casts shadows with soft edges based on ray tracing.

We change the textures of the scenes randomly. Floor and walls are changed independently which results in more combinations. We chose from 30 texture images (Fig. 4-3) to create 30 different corners. We render for each corner $1,000$ frames for both classes (“dynamic” and “static”) which sums up to $50,000$ synthetic images and around 6.2 GB.

In addition to the change of the texture, we change the material properties such as surface quality (e.g. roughness) and reflection strength (e.g. mirror or carpet) randomly. The position and color of the light, the path of the moving platform with the camera, and the scale and texture of the person behind the corner are randomly modified within certain boundaries. E.g. the height of the occluded person ranges from 0.75m to 2.2m (see Fig. 4-4 and Fig. 4-5). For both camera and person, the speed of motion changes independently for each corner randomly within the range $1 - 3$ m/s.

[3]https://www.blender.org/

Figure 4-3: Examples textures. Textures we use to create different scenes with Blender. We choose various textures ranging from dark to bright colors.

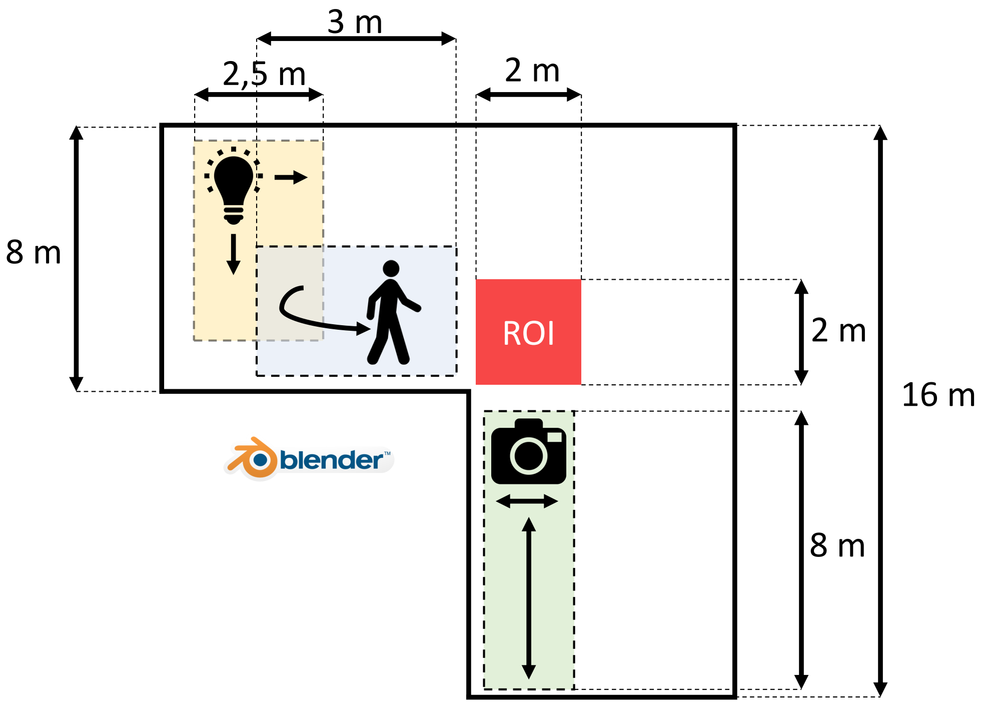


Figure 4-4: Blender Scene. In addition to material changes (such as texture or reflection properties) we change the position of the light and the path of the camera. The person walks randomly at different speeds within the blue box.

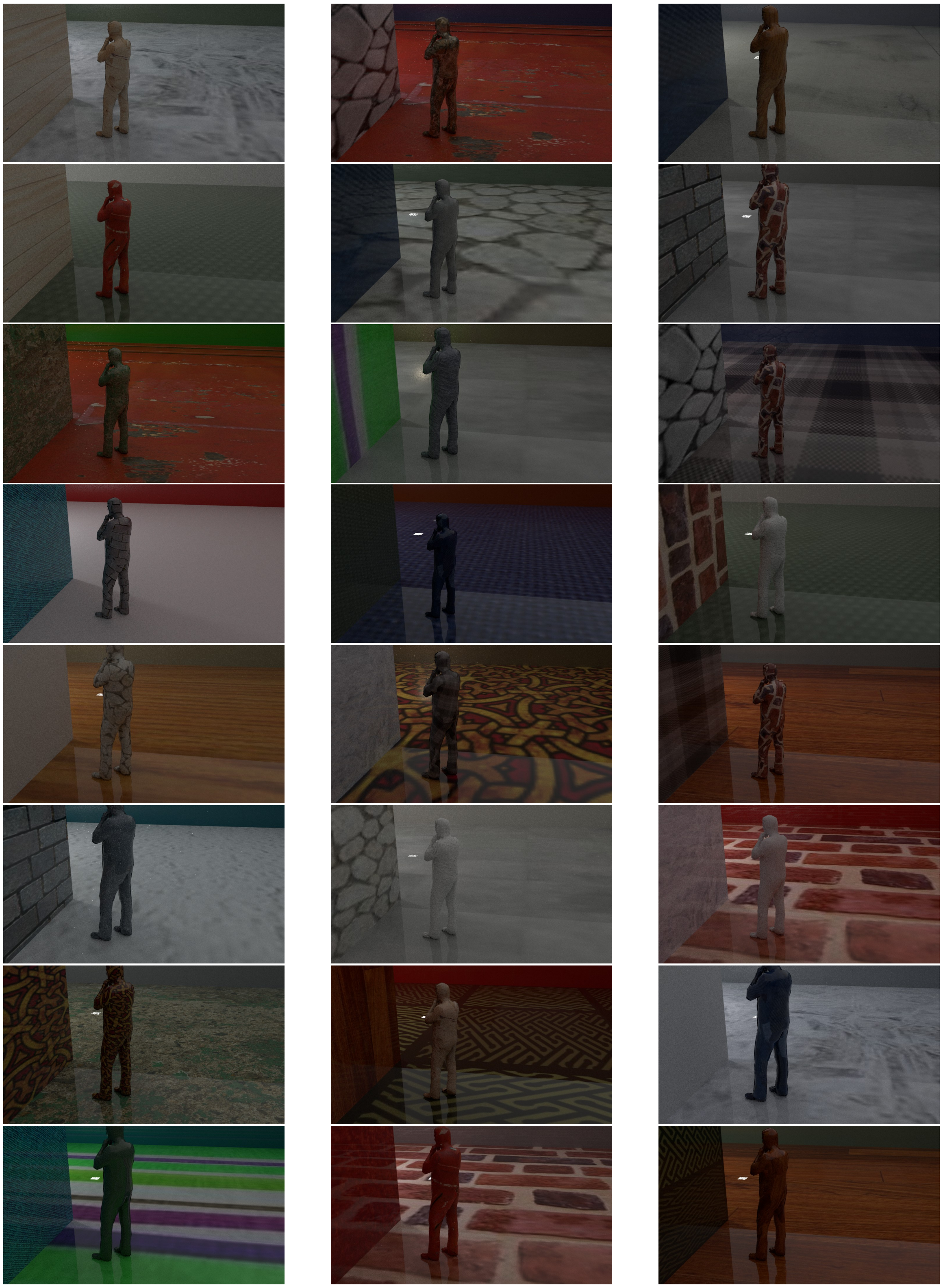


Figure 4-5: Simulation example corners. Examples of how we create random corners in the simulation.

4.3 Real-World Corner Data Collection

Besides the controlled environments in the Holodeck and Blender we evaluated how the algorithm performs “in the wild”. We created a real-world dataset with 4 different cameras. With the Canon and the webcams, we collected $85,000$ images of around 1 hour of data resulting in ca. 7.4 GB in total. With the global shutter camera from IDS[4] we collected around $42,000$ images at around 20Hz resulting in ca. 73.4 GB in total. To not only cover different types of corners, but also different image qualities, we chose the following cameras:

- **Canon** EOS 70D and the EFS $17-58$ mm lens (single-lens reflex (SLR) camera) for AprilTags as image registration step: The frame-rate was set to 30 fps, with codec H.264 and image dimensions 1920×1080.

- **Webcam** Logitech HD Webcam C525 (low-end webcam) for AprilTags as image registration step: The frame rate was set to 24 fps, with codec VP8 and image dimensions 1280×720.

- **Webcam** Logitech HD Webcam C925-e (high-end webcam) for AprilTags as image registration step: The frame rate was set to 20 fps, with codec VP8 and image dimensions 1280×720.

- **IDS uEye** UI-3241LE-M-GL (monochrome, global shutter CMOS) for DSO as image registration step: The frame rate was set to 20 fps, with png-compression and image dimensions 1280×1024.

We chose mainly corners where it is physically possible for a moving object behind a corner to cast a shadow. For these corners humans, if they pay close attention, might be able to see a shadow of an approaching person on the ground. We collected data ranging from high reflection floors and stone to dark carpet (see Fig. 4-6). In all real-world videos, we (1) label each complete video as “dynamic” or “static” depending on whether or not a person was asked to walk behind the corner or (2) with an

[4] `https://www.ids-imaging.us/home.html`

annotation tool on a frame by frame basis. The camera is moving in a range of 1 to 3 meters back and forth at around 3mph, whereas the person behind the corner moves randomly in a similar range and pace.

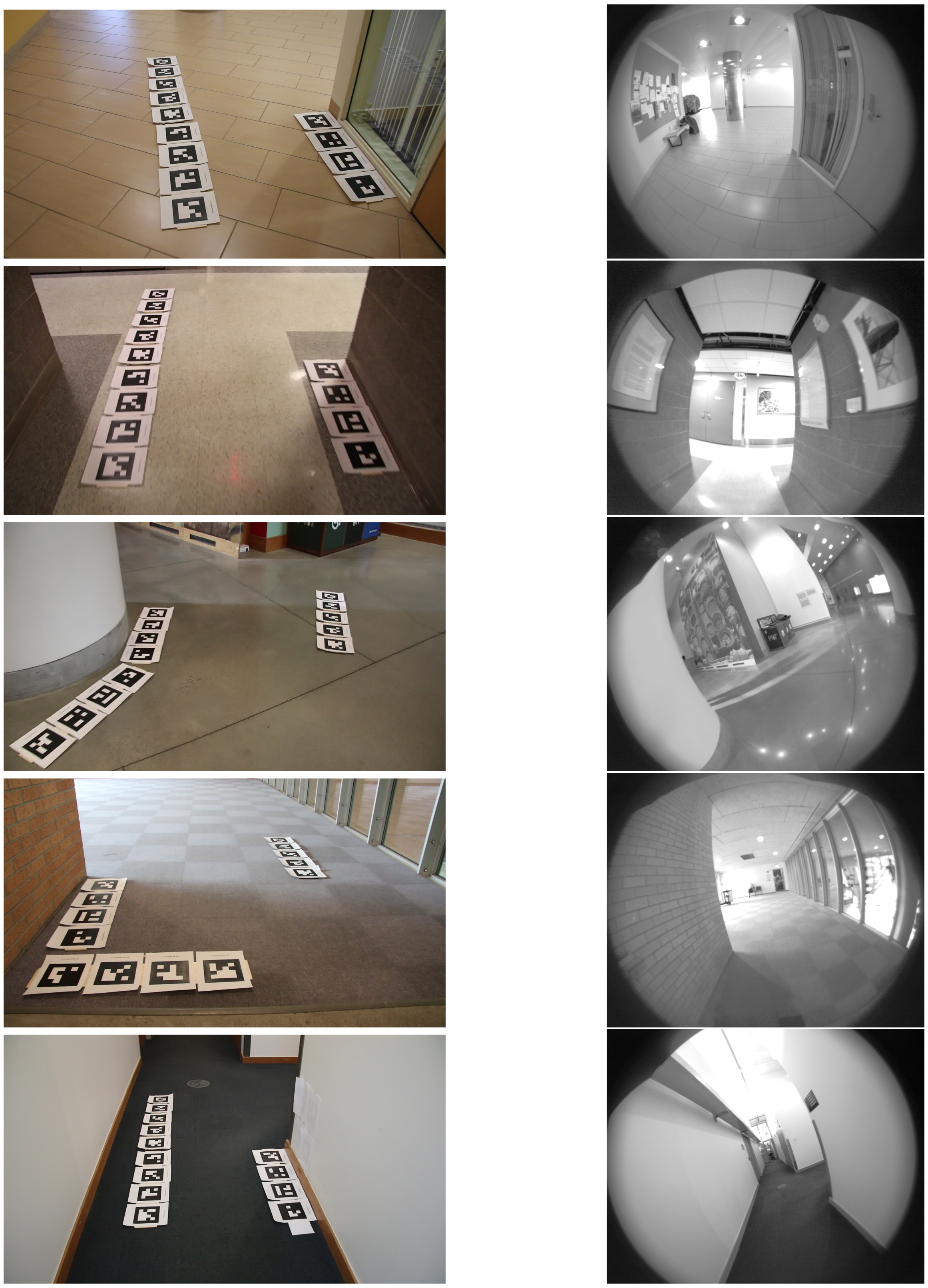


Figure 4-6: Real-world corner examples. On the left side images from videos recorded with the Canon and AprilTags. On the right side images of the same corners from videos recorded with the IDS uEye and DSO.

4.4 Autonomous Vehicle Setup

The algorithm was also tested in motion on an autonomous wheelchair (see Fig. 4-7) which has been designed as an indoor counterpart to the autonomous car presented in [26]. Therefore, the wheelchair has a very similar sensor configuration and a software stack based on ROS. This enables us to run (besides vehicle specific software parts, such as the low-level control) the same software packages on different vehicle types. This setup enables easily deploying a similar functionality to a real car. For the experiments, we added a (1) Logitech HD Webcam C925-e and (2) IDS uEye on top of the Wheelchair's top laser scanner to increase the look-ahead distance and improve the angle at which the cameras perceive the environment. Once the ShadowCam detects movement behind the corner we adjusted the control algorithm of the wheelchair so that it stops if movement is detected. As soon as the way is clear again the wheelchair resumes the forward motion.

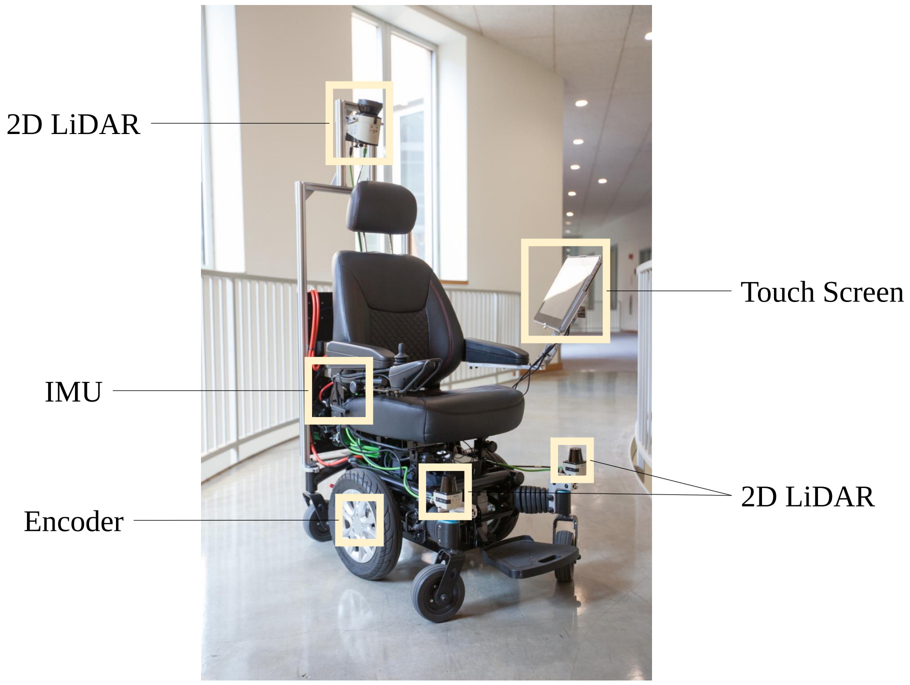


Figure 4-7: Autonomous wheelchair and the main sensors. We mounted the camera for the ShadowCam algorithm on top of the top LiDAR. Only the camera is required to run the ShadowCam algorithm.

The autonomous systems operate on a given map and with pre-defined path. The localization approach is based on laser scan matching, for re-planning in case of a moving obstacle we are using an RRT* variant (rapidly exploring random tree) and for path following a pure pursuit controller implementation.

4.5 Annotation Tool

The annotation tool allows us to hand label images on a frame-by-frame basis. We observed that by labeling complete videos as “static” or “dynamic” the labels are not accurate sometimes.

The user can load a csv file with labels per complete video and modify the label by stepping through all images. On the bottom a progress bar displays how many of the images in the video have been annotated. The top row contains our labeling options. The most important being “static” or “dynamic”. Next to this classification are the other labeling options for (1) indoor or outdoor (2) in which mode the video was recorded (e.g. walking, wheelchair, car) (3) the camera type (e.g. Canon, Webcam, IDS uEye) and (4) date of the recording.

The output of the annotation process is a new csv file with the same structure but appended columns containing the updated label information.

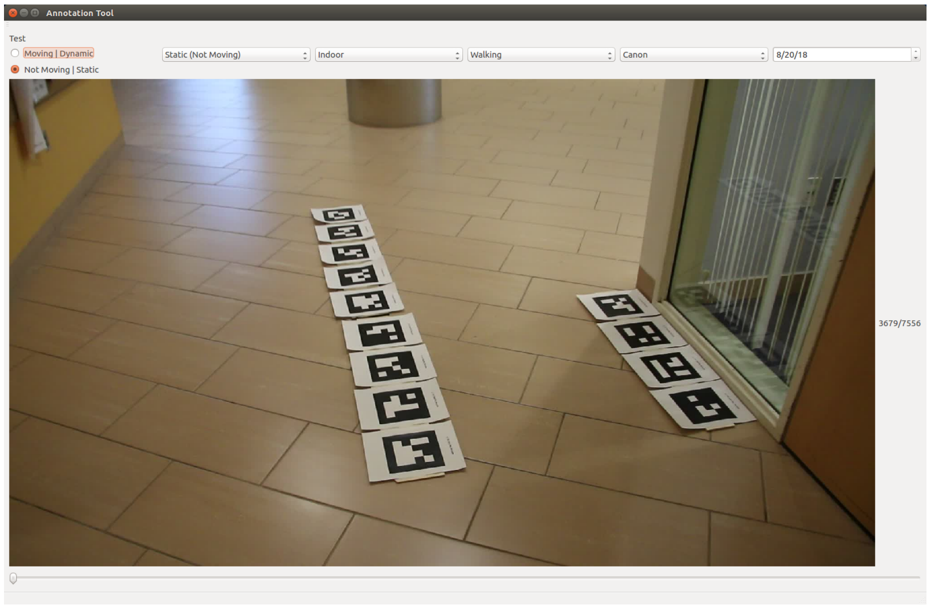


Figure 4-8: Annotation Tool. The user interface for hand labeling each image of a video on a frame per frame basis.

4.6 Summary

This chapter presents the data collection modalities which are required to provide evidence to the hypothesis that (1) the classification accuracy is in a similar range with and without AprilTags. (2) The classification accuracy is for both sequences "dynamic" and static" better than random. (3) The algorithm should generalize to different corners, materials and light conditions.

Chapter 5

Results

We quantitatively analyze the classification accuracy, real-time capability of the algorithm and demonstrate the use of ShadowCam integrated into an autonomous wheelchair. Specifically, we are evaluating the performance of two image registration methods (AprilTags from Sec. 3.2 and DSO from Sec. 3.3), the effect of new labels from the annotation tool (Sec. 4.5) and compare the classification accuracy of "dynamic" and "static" sequences. The success metric is as follows e.g.: When the ShadowCam pipeline classifies 7 out of 10 "static" sequences as "static" the classification accuracy would be 70%. The higher the classification accuracy the better the system performs.

Boxplots, histograms and ROC analysis are visualizing the performance of the ShadowCam algorithm on the respective datasets in Fig. 5-1, 5-2, 5-3, 5-4. The ShadowCam algorithm computes one value for each sequence which represents the sum over all "dynamic"-classified pixels (Eq. 3.15). The better the distributions of this value (for the cases with and without a moving obstacle) can be separated, the higher the classification accuracy can be once the threshold is set to the optimal point. For example in a "static" sequence with 1% as "dynamic" miss-classified pixels, the method would yield a threshold value of $255,000$ (= 100 px $\times$ 100 px $\times$ 10 images $\times$ 1% $\times$ 255 pixel value, where the image dimensions are 100, the length of the sequence is 10 and the maximum pixel value is 255). All sequences with values less than the specified threshold are classified as "static" and all sequences with values greater (or equal) than the specified threshold are classified as "dynamic" (Alg. 4).

In order to determine the threshold value upon which we classify a sequence as "dynamic" or "static", we examine the histograms over all corners of a specific recording setup (e.g. simulation or real-world) and found the noise of the camera to be correlated with the choice of the threshold. This confirms the intuition that higher noise levels lead to higher miss-classification rates. Table 5.1 gives an overview of the final thresholds we chose to create the mean classification accuracy plots in Fig. 5-1.

For the evaluation on the wheelchair, we implemented the ShadowCam algorithm in C++. For the two image registration methods we analyze the real-time performance:

- AprilTag image registration (Sec. 3.2): The implementation of a cyclic frame buffer enables us to compute a classification output at around 30 Hz for a sequence length of 10 frames. But since the camera we use on the wheelchair only runs at 20 Hz and only in around 1/3 of the images enough AprilTags get detected, the rate on the real system is around **7 Hz**. The low rate of AprilTag detection is mostly due to motion blur. However, for the speed of the wheelchair 7 Hz is fast enough, and the performance could be easily increased by switching to a better camera with a higher frame rate and/or image registration method. Our experiments with the autonomous wheelchair show that even with consumer grade cameras (such as the Logitech Webcam) the signal can be detected reliably.

- DSO image registration (Sec. 3.3): To enable DSO we up-graded the camera on the wheelchair to a global shutter camera which can run up to 60 fps. For the experiments we run it at 20 fps. In a distributed setup where one laptop runs the autonomous software and the other laptop runs the ShadowCam algorithm we can output classification results at around **20 Hz**.

Because our system is designed as an additional safety feature, we aim for a low rate of false "dynamic" classifications to provide a smooth driving experience (Fig. 5-1) without unnecessary interruptions. That is, when the algorithm detects movement then it is very likely someone is actually moving behind the corner. Thus, the majority of sequences get classified as “static” even if they sometimes contain a moving obstacle. It requires a strong movement behind the corner for the sequence to get classified as “dynamic”.

We compare the classification accuracy with two labelling techniques (1) based on videos which got labeled as a whole and (2) hand labels for each frame (new label from hand annotation = NL). We are expecting better results with the same algorithm if the labels are more accurate, since “dynamic” labeled videos contain also “static” sequences where e.g. the person behind the corner is not moving or the shadow is physically not visible.

Camera Type	Threshold	Percent of Pixel	Number of Corners
Holodeck	200000	$\approx 1\%$	1
Blender	220000	$\approx 1\%$	30
Canon	500000	$\approx 2\%$	11
Webcam	650000	$\approx 2.5\%$	3
IDS Ueye	700000	$\approx 2.5\%$	7

Table 5.1: Dataset Overview. Depending on the noise level of the image registration method and the camera recording quality we set a different threshold for the sum of the sequence (Eq. 3.15) for each camera type.

Our results show that it is possible to detect moving obstacles based on shadows out of the line of sight from a moving platform at indoor corners provided that the object can physically cast a shadow, we have access to reliable image registration, and the region of interest is known a priori (e.g. through a specific detector, map or annotation). Our algorithm is easy to deploy (it has only a few tune-able parameters) and generalizes to different corner settings when the image registration method is set to DSO. Importantly, we analyzed performance on different floors, light conditions and object sizes.

5.1 Overall Performance

With the threshold per camera type from Table 5.1 we run the ShadowCam algorithm on 7 corners for AT (= AprilTags) (Sec. 3.2) and DSO (= Direct Sparse Odometry) (Sec. 3.2) as image registration steps. Our experiments give further evidence that it is possible to detect moving obstacles from a moving platform at indoor corners where it is physically possible for a dynamic obstacle to cast a shadow at low speeds (e.g. 3mph). Fig. 5-1 indicates the mean classification accuracy for each data collection mode. Importantly we can observe that for both classes "static" and "dynamic" the accuracy is well above *random* 50%. Additionally, even when we remove AprilTags and rely instead on DSO as the image registration method we can maintain a classification accuracy of around 70%. Overall, we can also observe – which makes intuitively sense – that it's easier to detect "static" sequences than it is to detect moving shadows. As expected, we are able to perform very well on the simulated frames from Blender with a mean accuracy of above 85%.

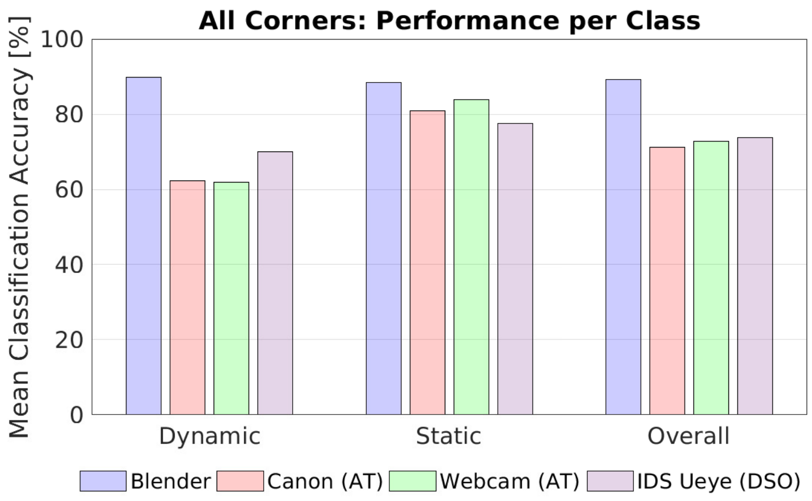
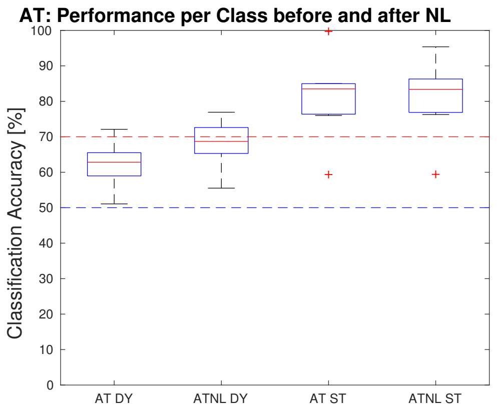


Figure 5-1: Classification Performance per class and method. As the distributions of the histogram (Fig. 5-3) suggest, we observe a low number of false positives. The classification accuracy for "static" sequences is high, whereas it is harder to detect movement based on shadows (i.e. it is harder to classify "dynamic" sequences correctly). The mean performance of the algorithm on real-world data acquisition methods is for both classes above 60% and overall even around 70%.

5.2 Effect of new Labels

We are also interested in the impact of the new labels (NL) from the annotation tool (Sec. 4.5). Fig. 5-2 compares the per class performance (“static” = ST and “dynamic” = DY) for both image registration methods before and after NL.



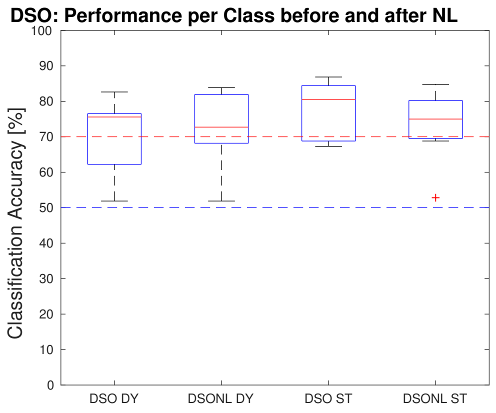


Figure 5-2: Effect of NL. On the left: For frames registered with AT we can observe that the mean accuracy for “dynamic” sequences increases around 5% and stays almost the same for “static” sequences. On the right: For frames registered with DSO we can observe that mean accuracy decreases which implies that e.g. some previously as “dynamic” labeled sequences did not contain movement from a shadow.

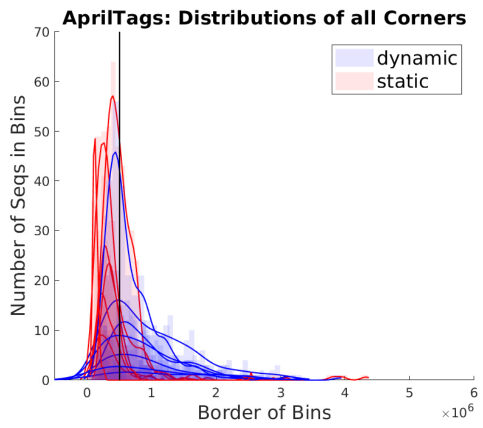
For the AT image registration we can observe on the left side of Fig. 5-2 that after NL the performance increases for DY. This effect can be explained by the observation that a label per complete video is likely to lead to miss-labeled frames (e.g. a sequence doesn’t contain a moving shadow but gets labeled as “dynamic”). Therefore, it makes sense that a dataset with cleaner labels boosts especially the performance on the DY side. For ST experiments most sequences – even with a label per complete video – are labeled accurately, since we made sure that most of the time no one would interfere the data collection and cast a moving shadow.

For the DSO image registration we can observe on the right side of Fig. 5-2 that the mean classification accuracy for both classes is reduced after NL. At first this might sound counter-intuitive, but it makes sense after a closer look at the underlying data. The image registration method with DSO is noisier which causes texture on the ground to change over time and to be detected as movement even though there is no “dynamic” shadow. After NL, more sequences from a video labeled completely as “dynamic” are then correctly labeled as “static”. These sequences are then miss-classified due to the texture movement on the ground caused by a noisier image registration in comparison to AT. Despite this effect the mean classification accuracy with DSO is well above 70%.

5.3 Performance on real-world Data

In this section we have a closer look at how the two image registration methods perform on the real-world data (i.e. we zoom in Fig. 5-1 on “Canon (AT)” and “IDS Ueye (DSO)”). As introduced in Sec. 4.3 the dataset equals for the AprilTag case to around 60 mins and around 4000 sequences and for the DSO case to around 40 mins and 1500 sequences (where each sequence consists out of 10 frames). This sums up to around 100 mins and 5500 sequences of real-world experiment data. With a mean classification accuracy of around 70% for both image registration methods this means that ShadowCam classifies 3850 sequences or 70 mins correctly into the categories “dynamic” or “static” depending on whether a dynamic obstacle was moving behind the corner. This helps to prevent a potential collision with a “dynamic” obstacle out of the direct line of sight. Since we aim for an algorithm parametrization which allows a smooth driving experience ShadowCam only outputs a “stop” signal when the movement behind the corner is relatively strong. Thus, the classification accuracy for both image registration methods is higher for “static” sequences.

The plots (Fig. 5-2, 5-3, 5-4) indicate coherent trends. DSO is weaker on ST sequences and stronger on DY than AT. This is for example reflected in the heatmap (Fig. 5-4) with the center of AT being higher and more to the right, where higher means a higher true positive rate for ST sequences and further to the right a higher false positive rate.



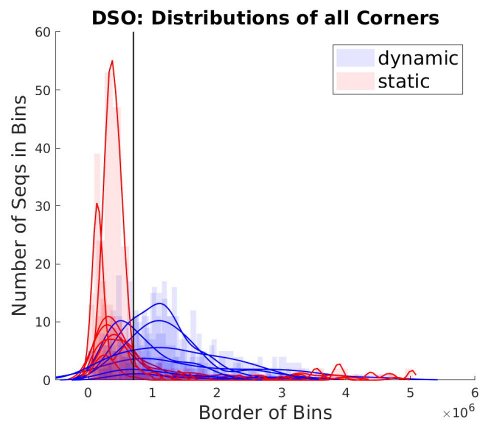
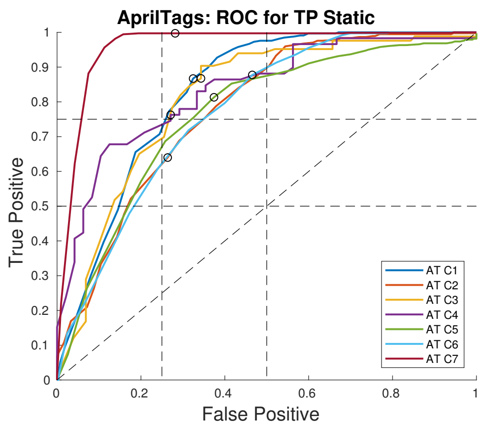
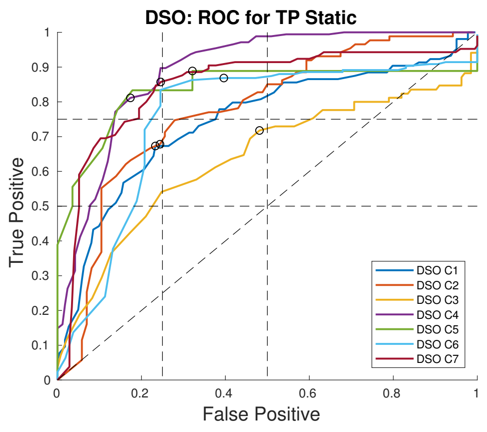
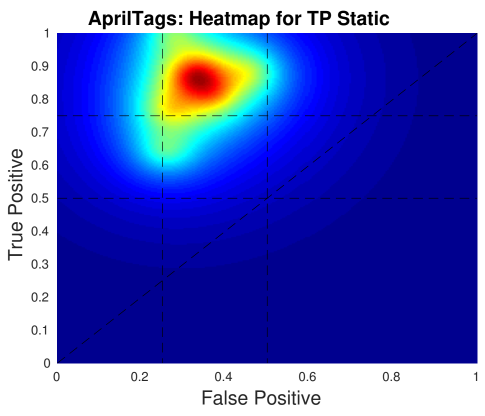


Figure 5-3: Signal Distributions. Histograms of the sum of the sequences (Eq. 3.15) for various corners with AprilTags on the left and without AprilTags and DSO as the image registration method on the right. The distributions of "dynamic" and "static" sequences explain the mean classification accuracy of around 70% when the threshold is set as the black vertical line indicates.







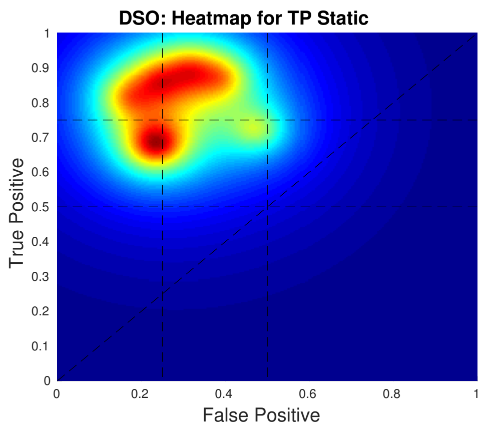


Figure 5-4: Receiver-Operating-Characteristic (ROC). Curves of data collected with AprilTags on the left and without on the right with the thresholds from Table 5.1. TP stands for true positive. If the heatmaps would have their center in the top left corner the classification accuracy would be 100%.

5.4 Evaluation of AprilTags as the Image Registration per Corner

In this section we show the classification performance for each individual corner in the dataset with AprilTags as the image registration methods (Sec. 3.2). For each corner we collected data with and without a “dynamic” obstacle.

For each corner we show (1) an example image from the input video stream on the top left corner (2) the distribution of “dynamic” and “static” sequences on the top right (3) the cropped ROI in the bottom left and (4) the mean classification performance per class before and after hand labelling the data on the bottom right.

For most corners the new labels increased the classification accuracy. The corners in the dataset cover a broad range of light conditions and floor materials.

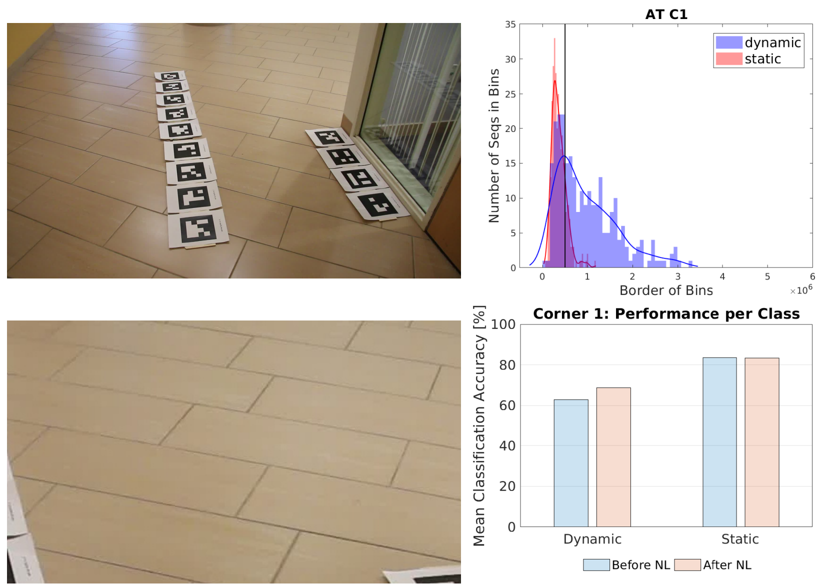


Figure 5-5: AT Corner 1. A bright environment with a lot of natural light from windows and a stone floor. The relatively strong shadow signal makes a good distinction between DY and ST possible.

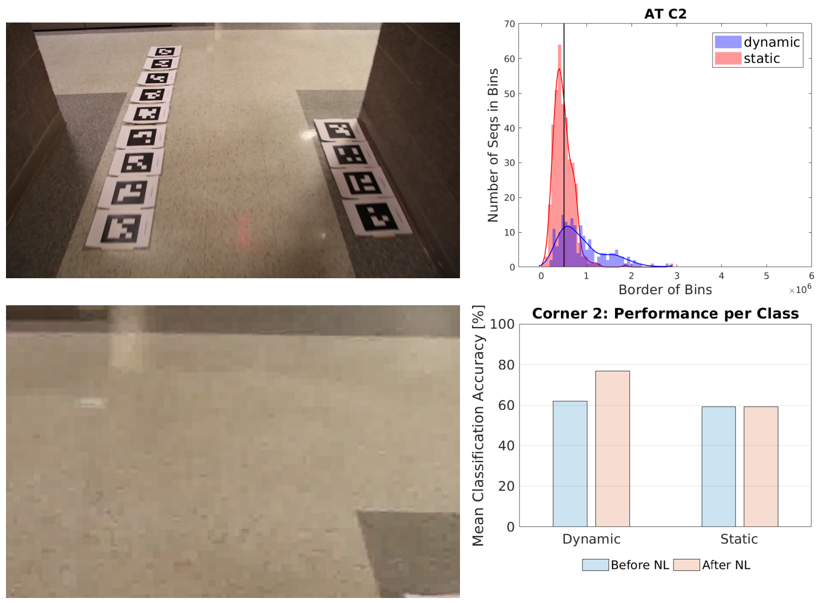


Figure 5-6: AT Corner 2. A reflective industry style floor with no natural light sources. Due to the ego motion the reflection of light on the ground leads to more miss-classifications on the ST side and thus a lower accuracy.

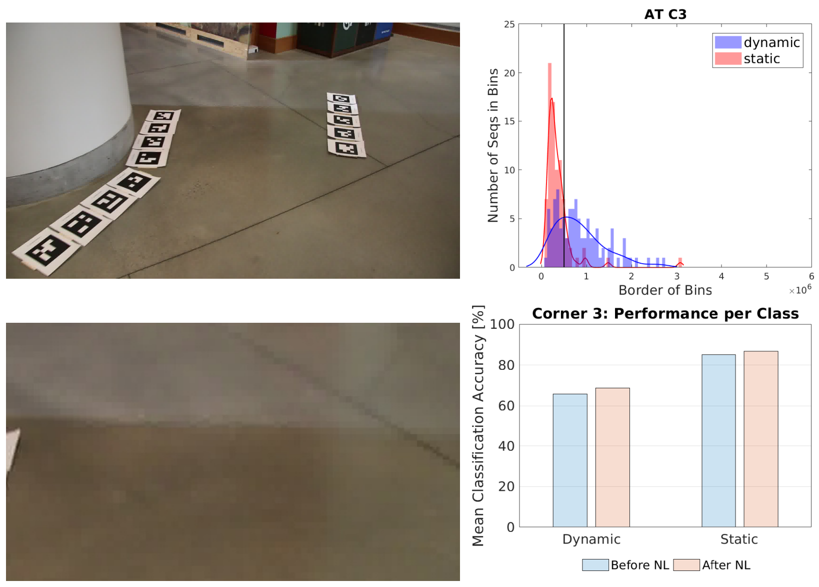


Figure 5-7: AT Corner 3. Concrete style floor with a strong natural light source from the left side. A relatively distinct distribution leads to a classification accuracy on the DY side in the high 60ties.

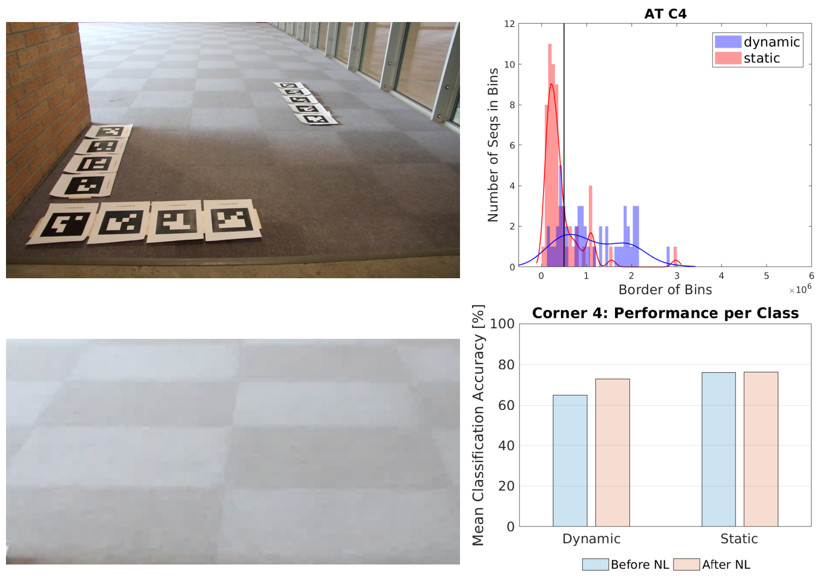


Figure 5-8: AT Corner 4. Carpet style floor with a strong natural light source from the left. Due to the placement of the tags only relatively few sequences can be registered.

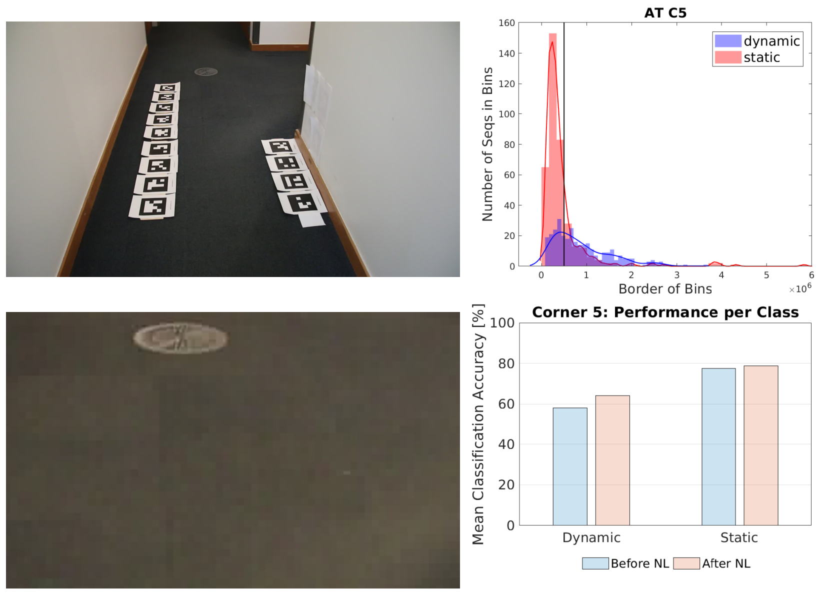


Figure 5-9: AT Corner 5. Carpet style floor and almost no natural light, mostly lamps. The prominent feature on the ground would sometimes be detected as movement.

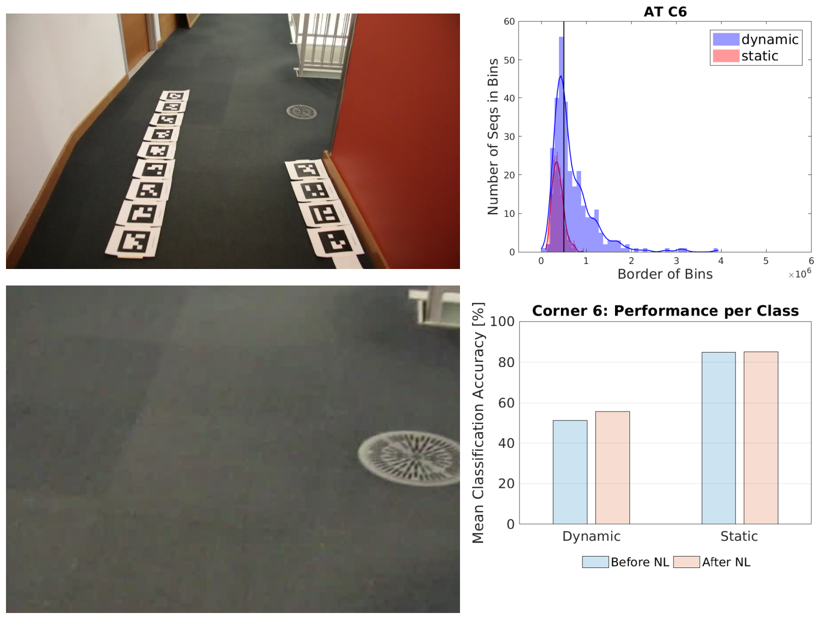


Figure 5-10: AT Corner 6. Carpet style floor with natural light from the ceiling. This leads to a physical shadow cast in a bad angle for detection.

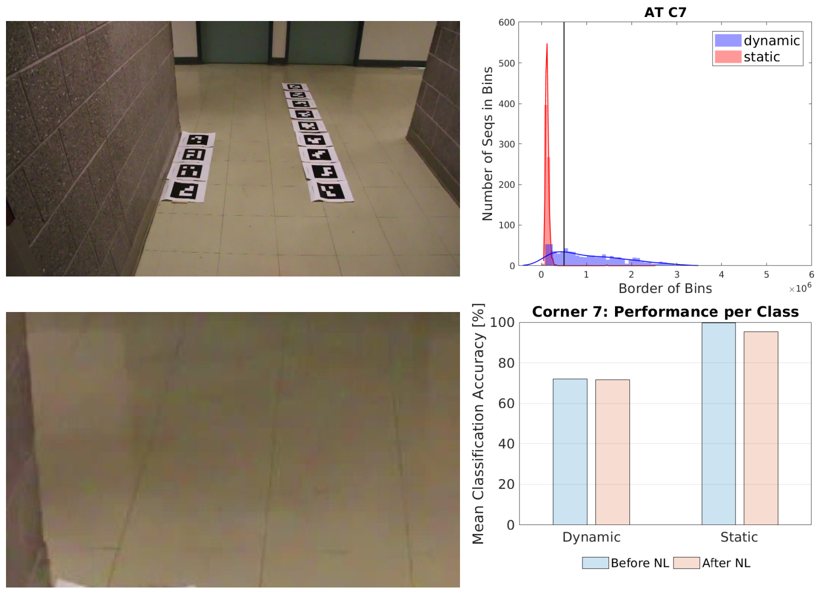


Figure 5-11: AT Corner 7. Industry style floor with no natural light and low reflection properties. Very good constellation for classification (both ST and DY).

5.5 Evaluation of DSO as the Image Registration per Corner

To complement Sec. 5.4 this section includes the detailed analysis of the same corners but without AprilTags and instead with DSO as the image registration method (Sec. 3.3). For each corner we collected data with and without a "dynamic" obstacle. We can observe that the image registration with DSO is noisier and can make it harder to distinguish pixel changes over time induced either by a moving shadow or by a noisy registration. Overall (as for AT Sec. 5.4) the performance for ST sequences is higher than for DY.

The structure per corner out of the dataset follows the one from AT where we show an example image from the input video stream, the distribution, ROI and performance before and after NL. The four red points in the bottom left images mark the annotations for the ROI.

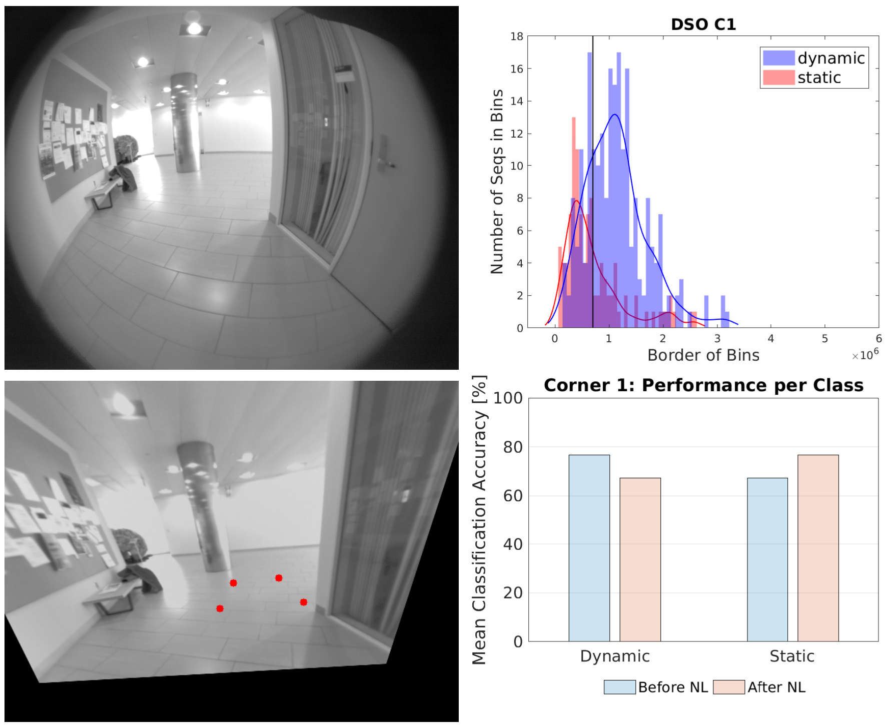


Figure 5-12: DSO Corner 1. A bright environment with a lot of natural light from windows and a stone floor with strong contrasting gaps. The gaps in the floor also cause some miss-classification on the DY side before NL.

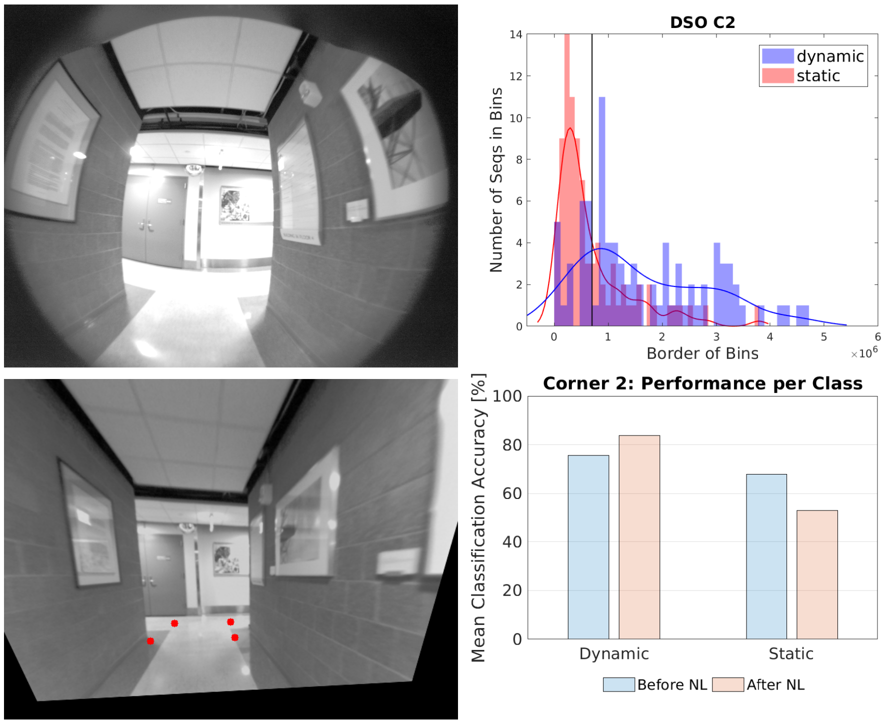


Figure 5-13: DSO Corner 2. A reflective industry style floor with no natural light sources and contrasting floor structures. Due to the floor structures it's harder to classify "static" sequences.

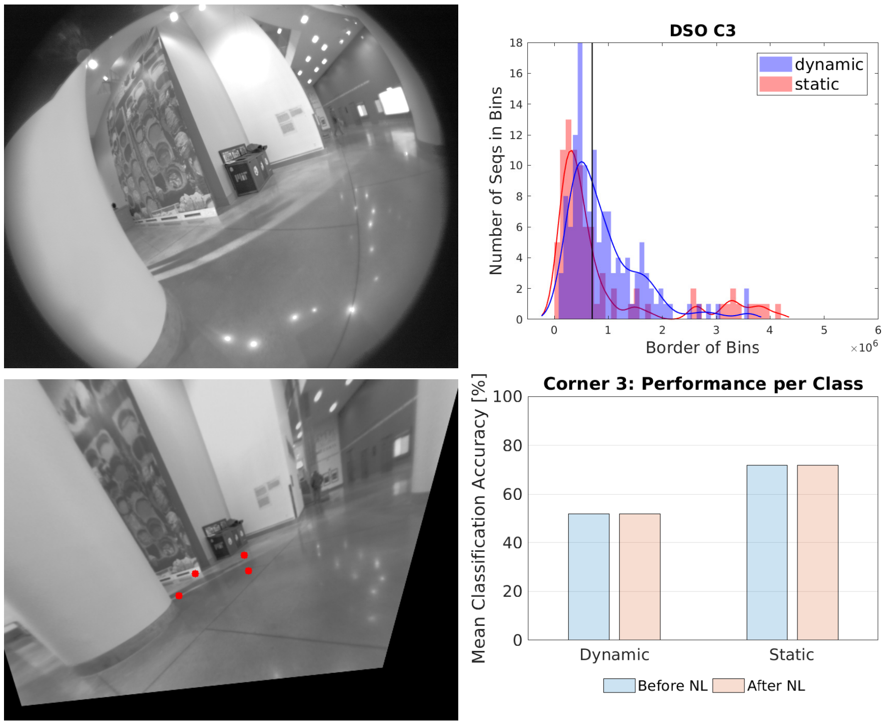


Figure 5-14: DSO Corner 3. Concrete style floor with a strong natural light source from the left side which also causes a strong edge on the ground. This makes it harder to distinguish DY from ST.

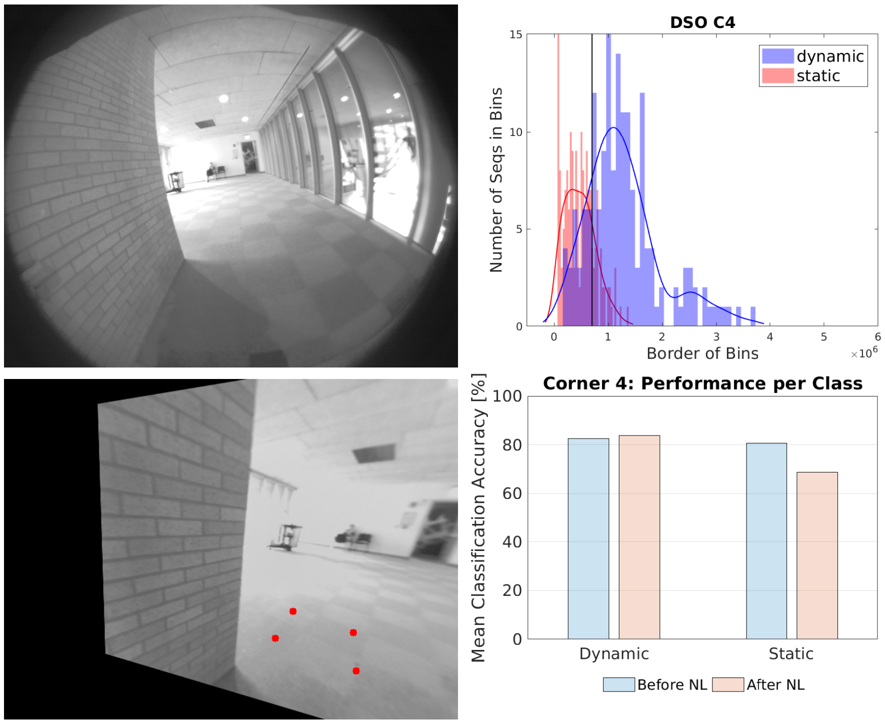


Figure 5-15: DSO Corner 4. Carpet style floor with a strong natural light source from the left (depending on the time of day). This results in a relatively strong shadow signal.

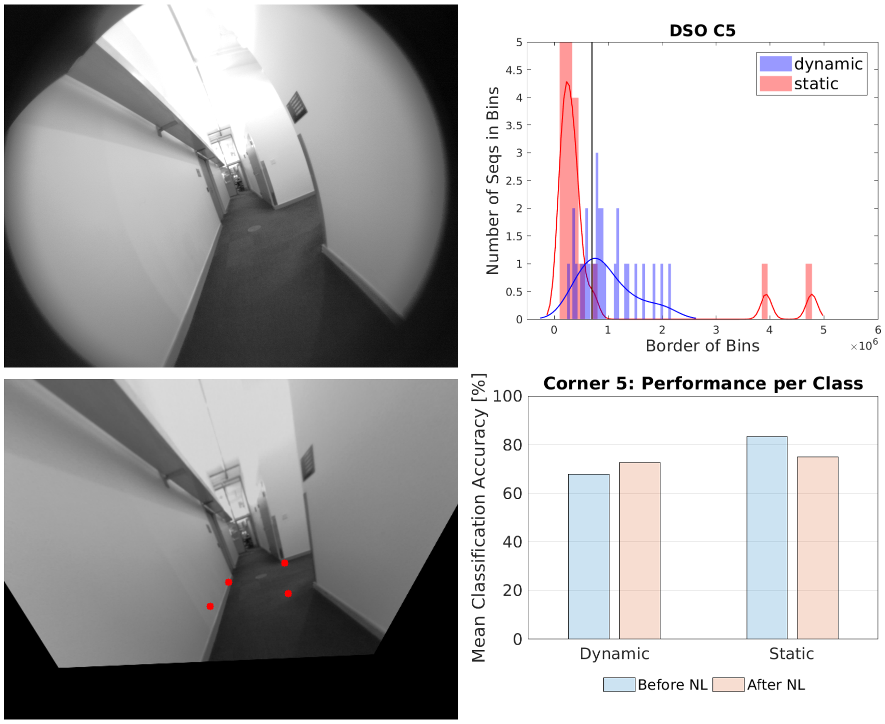


Figure 5-16: DSO Corner 5. Carpet style floor and almost no natural light, mostly lamps from the ceiling. After NL the accuracy for DY increases and decreases for ST.

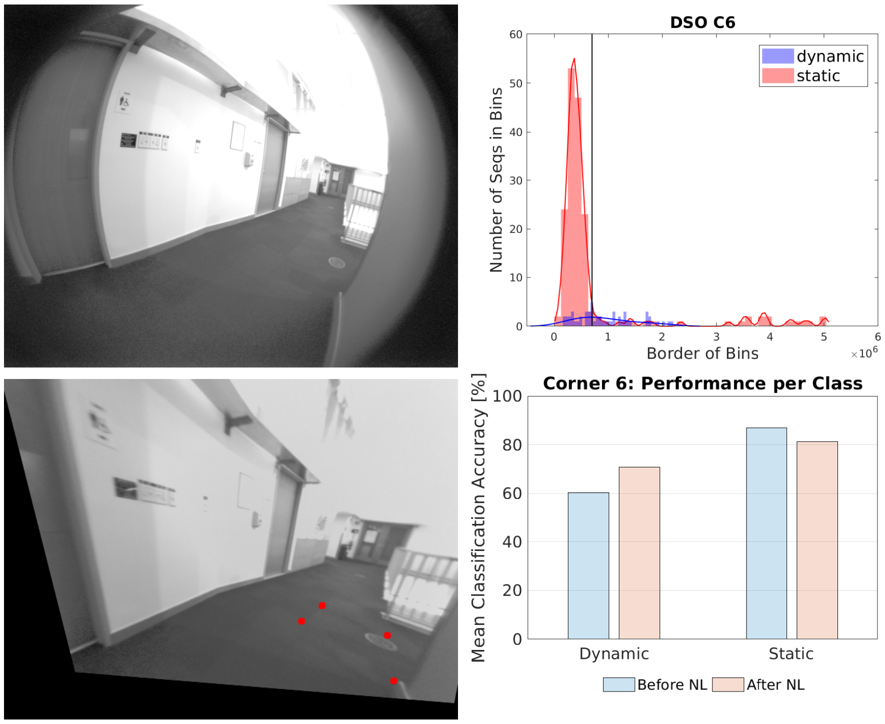


Figure 5-17: DSO Corner 6. Carpet style floor with natural light from the ceiling and a strong feature on the ground. For this corner exists more “static” sequences in the database than “dynamic”.

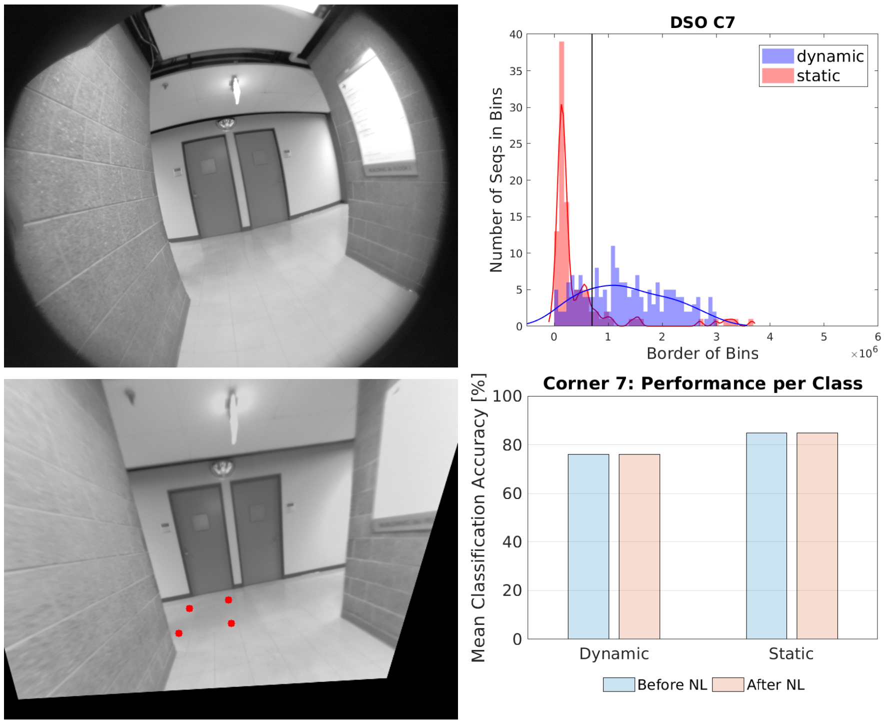


Figure 5-18: DSO Corner 7. Industry style floor with no natural light and low reflectively. This setting allows relatively good classification.

5.6 Summary

In this chapter we applied the technical approach (Chap. 3) to the collected and labeled dataset (Chap. 4). We demonstrate that our hypotheses hold. (1) The classification accuracy is in a similar range with and without AprilTags of around 70% when both the obstacle's and the ego vehicle's speed is at around 3mph and the ROI is known. (2) The classification accuracy is for both sequences "dynamic" and static" better than random. (3) The algorithm is generalizing to different corners, materials and light conditions for both image registration methods (Sec. 5.4 and Sec. 5.5).

Chapter 6

Conclusions

From a high-level point of view, we look at a low SNR signal and reduce the SNR even further by adding camera movement. By making the assumption that the ROI is known we reduce our problem to the core research problem of motion detection based on shadows from a slow-moving platform (around 3mph). For the image registration step we compare AprilTags with DSO and were able to show that even with a visual odometry method in feature poor indoor environments the signal can be detected with our ShadowCam algorithm. This algorithm is a real-time capable motion detection algorithm which is robust against noise but is still able to detect a low SNR signal. To the best of our knowledge we present the first autonomous wheelchair which can detect and react to out of the line of sight moving obstacles based on their shadows.

6.1 Lessons Learned

The most important learning – from the initial idea for this work until we were able to detect first shadows from moving obstacles – was to iterate often and to learn from mistakes, in other words changing the plan, but not the goal. Focusing on the core research problem also helped. In a first step we assumed we could rely on an almost perfect image registration method and placed AprilTags on the ground plane. We first had to give evidence to the hypothesis that it is possible to detect shadows from a moving platform. In a second iteration we integrated a visual odometry method into the ShadowCam pipeline. Since visual odometry is an active research area itself this approach was riskier, but also enabled us to generalize the ShadowCam pipeline to any corner without AprilTags.

For rapid prototyping we first developed in Python and then ported the algorithms to C++ based on OpenCV and ROS to achieve real-time performance.

On the one hand, working with real-world data, complex hardware and software systems such as the autonomous wheelchair adds another challenging dimension to the problem. On the other hand, mastering this challenge and demonstrating the integration of a real-time capable implementation of the ShadowCam pipeline into an autonomous system validates our results even more.

6.2 Future Work

We are excited to further explore more application areas for our motion detection algorithm and ways to improve accuracy and robustness.

So far we only looked at indoor corners where it is physically possible to cast a shadow for a moving obstacle. Another major challenge will be to come up with a way of detecting whether it is physically possible to cast a shadow e.g. from behind a corner or a parked car. When it is physically not possible to cast a shadow, the autonomous car should not rely on ShadowCam to detect a moving shadow out of the line of sight of the ego vehicle. One idea to come closer to a solution for this problem might be a prediction of the light source and direction. This would allow the system to predict from which side it would be possible to cast a shadow and which height the moving obstacle needs to reach to be able to cast a detectable shadow.

Another option worth exploring might be a more data and deep learning driven approach for the classification into “static” or “dynamic” sequences. This could help to boost accuracy. One could also explore the possibility of not only classifying into “static” or “dynamic” sequences but also predicting e.g. the speed of the movement and the direction. This would allow to continue driving for example when the obstacle is moving away from the ego vehicle’s direction of driving. Without this added functionality the system would detect movement and stop or slow down to prevent a collision even when the obstacle is walking away.

Reflection of movement on windows or on parked cars might be another way to detect obstacles out of the direct line of sight. This might be possible with a motion detection algorithm like ShadowCam in combination with a system that detects and registers not only areas close to a corner but more relevant areas such as mirror like surfaces.

Continuing with a stereo vision setup might also help to stabilize frame sequences. We are also planning to collect more data and to test the algorithm at higher speeds. This might pave the way to bring the ShadowCam algorithm as an additional safety feature to autonomous cars.

6.3 Summary

This thesis is mostly based on the work we presented at ITSC 2018 "ShadowCam: Real-Time Detection Of Moving Obstacles Behind A Corner For Autonomous Vehicles". In essence, we were able to show that a shadow cast by a moving obstacle out of the direct line of sight shouldn't only be treated as unwanted noise in an image but can actually provide safety relevant features. We hope this research will lead to safer driving and inspire others to treat shadows more like an additional signal than as unwanted noise. The new dataset can be used for further research and in classes related to signal processing and computer vision.

Appendix A

Source Code

- https://github.mit.edu/fnaser/dso_ros
- https://github.mit.edu/fnaser/dso
- https://github.mit.edu/fnaser/shadow_cam
- https://github.mit.edu/fnaser/cornercam
- https://github.mit.edu/fnaser/ueye
- https://github.mit.edu/fnaser/five-video-classification-methods
- https://github.com/fnaser/usb_cam
- https://github.com/mit-drl/knightrider-mobility
- https://github.com/mit-drl/knightrider-common
- https://github.com/mit-drl/pub-2018-naser-cornernet
- https://github.com/mit-drl/deepknight

Bibliography

[1] Fadel Adib and Dina Katabi. See Through Walls with WiFi! In *Proceedings of the ACM SIGCOMM Conference*, 2013.

[2] Nijad Al-Najdawi, Helmut E. Bez, Jyoti Singhai, and Eran. A. Edirisinghe. A survey of cast shadow detection algorithms. *Pattern Recogn. Lett.*, 33(6):752–764, April 2012.

[3] Katherine L Bouman, Vickie Ye, Adam B Yedidia, Frédo Durand, Gregory W Wornell, Antonio Torralba, and William T Freeman. Turning Corners Into Cameras: Principles and Methods. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[4] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* O'Reilly Media, Inc., 2008.

[5] ZHU Yun-fang BWU Jing, DU Xin and GU Wei-kang. Adaptive Fuzzy Filter Algorithm for Real-time Video Denoising. In *2008 9th International Conference on Signal Processing*, 2008.

[6] Joao Carreira and Andrew Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[7] Axel Pinz Christoph Feichtenhofer and Andrew Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] Peter Corke, Rohan Paul, Winston Churchill, and Paul Newman. Dealing with Shadows: Capturing Intrinsic Scene Appearance for Image-Based Outdoor Localisation. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[9] Vikas Dhiman, Quoc-Huy Tran, Jason J. Corso, and Manmohan Chandraker. A Continuous Occlusion Model for Road Scene Understanding. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] Rob Fergus-Lorenzo Torresani Du Tran, Lubomir Bourdev and Manohar Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[11] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *CoRR*, abs/1607.02565, 2016.

[12] Graham D. Finlayson, Mark S. Drew, and Cheng Lu. Entropy Minimization for Shadow Removal. *International Journal of Computer Vision*, 85(1):35–57, 2009.

[13] Thomas Frank, Michael Haag, Henner Kollnig, and Hans-Hellmut Nagel. Tracking of Occluded Vehicles in Traffic Scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 1996.

[14] Genevieve Gariepy, Francesco Tonolini, Robert Henderson, Jonathan Leach, and Daniele Faccio. Detection and Tracking of Moving Objects Hidden from View. *Nature Photonics*, 10(1):23–26, 2016.

[15] Christiano Gava. 2D projective transformations (homographies). `https://ags.cs.uni-kl.de/fileadmin/inf_ags/3dcv-ws11-12/3DCV_WS11-12_lec04.pdf`. Accessed: 2018-12-10.

[16] Georgia Gkioxari, Ross Girshick, and Jitendra Malik. Contextual action recognition with r*cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[17] Marcus Rohrbach Subhashini Venugopalan Sergio Guadarrama Kate Saenko Jeff Donahue, Lisa Anne Hendricks and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] Jesus Bescos. Real-time shot change detection over online MPEG-2 video. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(4):475–484, 2004.

[19] Wu Jing, Du Xin, Zhu Yun-fang, and Gu Wei-kang. Adaptive Fuzzy Filter Algorithm for Real-Time Video Denoising. In *Proceedings of the International Conference on Signal Processing (ICSP)*, 2008.

[20] Achuta Kadambi, Hang Zhao, Boxin Shi, and Ramesh Raskar. Occluded Imaging with Time-of-Flight Sensors. *Transactions on Graphics (ToG)*, 35(2):15, 2016.

[21] Salman H. Khan, Mohammed Bennamoun, Ferdous Sohel, and Roberto Togneri. Automatic Shadow Detection and Removal from a Single Image. *Transactions on Pattern Analysis and Machine Intelligence*, 38(3):431–446, 2016.

[22] Martin Laurenzis, Andreas Velten, and Jonathan Klein. Dual-mode Optical Sensing: Three-Dimensional Imaging and Seeing Around a Corner. *Optical Engineering*, 56(3), 2017.

[23] A. Leone and C. Distante. Shadow Detection for Moving Objects based on Texture Analysis. *Pattern Recognition*, 40(4):1222–1233, 2007.

[24] Will Maddern, Alexander D. Stewart, and Paul Newman. LAPS-II: 6-DoF day and night visual localisation with prior 3D structure for autonomous road vehicles. In *Proceedings of the Intelligent Vehicles Symposium (IV)*, 2014.

[25] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

[26] Felix Naser, David Dorhout, Stephen Proulx, Scott Drew Pendleton, Hans Andersen, Wilko Schwarting, Liam Paull, Javier Alonso-Mora, Marcelo H Ang, Sertac Karaman, Russ Tedrake, John Leonard, and Daniela Rus. A Parallel Autonomy Research Platform. In *Proceedings of the Intelligent Vehicles Symposium (IV)*, 2017.

[27] Eshed Ohn-Bar and Mohan Manubhai Trivedi. Learning to Detect Vehicles by Clustering Appearance Patterns. *Transactions on Intelligent Transportation Systems*, 16(5):2511–2521, 2015.

[28] Edwin Olson. AprilTag: A Robust and Flexible Visual Fiducial System. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2011.

[29] Wanli Ouyang and Xiaogang Wang. A Discriminative Deep Model for Pedestrian Detection with Occlusion Handling. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[30] Rohit Pandharkar, Andreas Velten, Andrew Bardagjy, Everett Lawson, Moungi Bawendi, and Ramesh Raskar. Estimating motion and size of moving non-line-of-sight objects in cluttered environments. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[31] Marc Gelgon Patrick Bouthemy and Fabrice Ganansia. A Unified Approach to Shot Change Detection and Camera Motion Characterization. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(7):1030–1044, 1999.

[32] Rishi Ramakrishnan, Juan Nieto, and Steve Scheding. Shadow compensation for outdoor perception. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2015.

[33] Rohit Ghosh. Deep Learning for Videos: A 2018 Guide to Action Recognition. `http://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review`. Accessed: 2018-12-03.

[34] Guy Rosman, Shachar Shem-Tov, David Bitton, Tal Nir, Gilad Adiv, Ron Kimmel, Arie Feuer, and Alfred M. Bruckstein. Over-parameterized optical flow using

a stereoscopic constraint. In Alfred M. Bruckstein, Bart M. ter Haar Romeny, Alexander M. Bronstein, and Michael M. Bronstein, editors, *Scale Space and Variational Methods in Computer Vision*, pages 761–772, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[35] Andres Sanin, Conrad Sanderson, and Brian C Lovell. Shadow detection: A survey and comparative evaluation of recent methods. *Pattern recognition*, 45(4):1684–1695, 2012.

[36] Chee Kiat Seow and Soon Yim Tan. Non-Line-of-Sight Localization in Multipath Environments. *Transactions on Mobile Computing*, 7(5):647–660, 2008.

[37] Dongeek Shin, Ahmed Kirmani, Vivek K Goyal, and Jeffrey H Shapiro. Photon-Efficient Computational 3-D and Reflectivity Imaging with Single-Photon Detectors. *Transactions on Computational Imaging*, 1(2):112–125, 2015.

[38] Dongeek Shin, Feihu Xu, Dheera Venkatraman, Rudi Lussana, Federica Villa, Franco Zappa, Vivek K Goyal, Franco NC Wong, and Jeffrey H Shapiro. Photon-efficient imaging with a single-photon camera. *Nature communications*, 7, 2016.

[39] Ming Yang Shuiwang Ji, Wei Xu and Kai Yu. 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

[40] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014.

[41] Rangachar Kasturi Ullas Gargi and Susan H. Strayer. Performance Characterization of Video-Shot-Change Detection Methods. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1):1–13, 2000.

[42] Koen Van De Sande, Theo Gevers, and Cees Snoek. Evaluating color descriptors for object and scene recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

[43] Andreas Velten, Thomas Willwacher, Otkrist Gupta, Ashok Veeraraghavan, Moungi G Bawendi, and Ramesh Raskar. Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging. *Nature communications*, 3:745, 2012.

[44] Alex Wallar, Brandon Araki, Raphael Chang, Javier Alonso-Mora, and Daniela Rus. Foresight: Remote Sensing for Autonomous Vehicles Using a Small Unmanned Aerial Vehicle. In *Proceedings of the Conference on Field and Service Robotics (FSR)*, 2018.

[45] John Wang and Edwin Olson. AprilTag 2: Efficient and Robust Fiducial Detection. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[46] Georges Younes, Daniel C. Asmar, and Elie A. Shammas. A survey on non-filter-based monocular visual SLAM systems. *CoRR*, abs/1607.00470, 2016.

[47] Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.