Indoor robot gardening: design and implementation

Citation: Correll, Nikolaus et al. "Indoor Robot Gardening: Design and Implementation." Intelligent Service Robotics 3.4 (2010): 219–232. Web.

As Published: http://dx.doi.org/10.1007/s11370-010-0076-1

Publisher: Springer-Verlag

Persistent URL: http://hdl.handle.net/1721.1/71699

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Noname manuscript No.
(will be inserted by the editor)

Indoor Robot Gardening: Design and Implementation

Nikolaus Correll · Nikos Arechiga · Adrienne Bolger · Mario Bollini · Ben Charrow · Adam Clayton · Felipe Dominguez · Kenneth Donahue · Samuel Dyar · Luke Johnson · Huan Liu · Alexander Patrikalakis · Timothy Robertson · Jeremy Smith · Daniel Soltero · Melissa Tanner · Lauren White · Daniela Rus

the date of receipt and acceptance should be inserted later

Abstract This paper describes the architecture and implementation of a distributed autonomous gardening system with applications in urban/indoor precision agriculture. The garden is a mesh network of robots and plants. The gardening robots are mobile manipulators with an eye-in-hand camera. They are capable of locating plants in the garden, watering them, and locating and grasping fruit. The plants are potted cherry tomatoes enhanced with sensors and computation to monitor their well-being (e.g. soil humidity, state of fruits) and with networking to communicate servicing requests to the robots. By embedding sensing, computation and communication into the pots, task allocation in the system is de-centrally coordinated, which makes the system scalable and robust against the failure of a centralized agent. We describe the architecture of this system and present experimental results for navigation, object recognition and manipulation as well as challenges that lie ahead toward autonomous precision agriculture with multi-robot teams.

This work was done at the Department of Electrical Engineering and Computer Science and the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02478, USA. Corresponding author nikolaus.colorado@colorado.edu

An abbreviated version of this paper was presented at the International Conference on Intelligent Robots and Systems (IROS), St. Louis, MO, 2009, (Correll et al 2009).

N. Correll
Department of Computer Science, University of Colorado at Boulder, 1111 Engineering Drive, 80309 Boulder CO

Nikos Arechiga · Adrienne Bolger · Mario Bollini · Ben Charrow · Adam Clayton · Felipe Dominguez · Kenneth Donahue · Samuel Dyar · Luke Johnson · Huan Liu · Alexander Patrikalakis · Timothy Robertson · Jeremy Smith · Daniel Soltero · Melissa Tanner · Lauren White · Daniela Rus
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Vassar Street 32, 02139 Cambridge MA

1 Introduction

Our long-term goal is to develop an autonomous green house consisting of autonomous robots and pots and plants enhanced with computation, sensing, and communication. The network of robots, pots, and plants transforms energy, water and nutrients into produce and fruits. In this type of precision agriculture system water and nutrients will be delivered locally on-demand and fruit will be harvested when they are ripe. Plants will drive the robots' activities in the garden using sensors to monitor their local environment conditions, and maintain a local database storing fruit location and maturity that is obtained in interaction with robots. By relying sensing and coordination of the system to individual plants, the proposed system operates without any centralized control which makes it scalable and robust.

From an economical perspective, cultivation of specialty crops (such as fruits and vegetables) requires a huge amount of manual labor when compared with broadland crops. This need has recently led to multiple initiatives in the United States (e.g. the *Comprehensive Automation for Specialty Crops (CASC)* program) and Europe (e.g. with in the scope of the 7th Framework program which aims at sustainable crop and forestry management, among others).

From a social perspective, robotic precision agriculture might allow for the economical maintenance of gardens in cities and buildings, which might positively contribute to the building's and city's climate and possibly even the autonomy of its inhabitants. For instance

Fig. 1 Tomato plants are arranged on a platform of 3×4 meters. Plants are illuminated by growing lights.

Fig. 2 Roof-garden in Osaka, Japan. Plants might positively contribute to the city climate, and potentially even contribute to the building's inhabitants diet.

Figure 2 shows a roof garden in Osaka, Japan, on an integrated shopping, commercial and residential complex. While the impact of a single installation on city climate is marginal, a robot-supported, city wide deployment might not be. Other possible installations are indoors in shopping malls or inside integrated living, shopping and commercial high-rises.

From an ecological perspective, the ability to precisely manipulate individual plants bears not only the potential to save water, pesticides and fertilizer, but might also allow to move from the dominating monocultures to heterogeneous plant cultures, which we believe will have tremendous advantages for insects and wildlife, and might lead to more efficient soil utilization.

We argue that the precision agriculture as we envision it is not only a manipulation challenge, but as tending is sensor-based and plant-centered, requires novel sensing methods to sense plant status and coordination mechanisms to allocate the robots and its capabilities to plants in an on-demand basis.

We describe the system architecture and algorithmic details that constitute the distributed robot garden that consist of autonomous robots and sensing, computing and communicating pots. We also describe experimental results for watering plants, identifying tomatoes, harvesting tomatoes, and coordinating the interaction between robots and plants in this system that have been collected during a spiral-based design approach and motivated our design decisions.

Our work provides a proof of concept working system but much work remains to be done to achieve our vision of self-sustaining precision agriculture with teams of gardening robots.

1.1 Project Structure

The system has been designed following a spiral-design philosophy that aimed at experimental evaluation of working sub-systems of the system after each term (Summer '08, Fall '08, Summer '09) and a rotating student population working on this project. This approach has led to a series of major design revisions for almost all of the hardware components, sensors and algorithms and has led to the system presented in this paper.

In order to arrive at a working system as fast as possible, we relied as much as possible on off-the-shelf hardware and algorithms that are available open-source. This approach allowed us to focus on the key algorithmic and systems aspects that were needed to achieve a certain function.

1.2 Related work

Commercial agriculture has reached a high level of automation, although mainly for broad-land crops, e.g. using autonomous combiners (Blackmore 2007). Also, fine-grain satellite imagery is commercially available for targeted pesticide and fertilizer application, leading to the novel paradigm of *precision agriculture*, with the primary goal of saving pesticides and water (Al-Kufaishi et al 2005; Srinivasan 2006). At the same time, interest is growing to move towards individually smaller robotic platforms that can apply precise sensing and manipulation in the field (Blackmore et al 2006).

Indeed, advances in mechatronics and computer vision (Jimenez et al 2000) have led to autonomous solutions for harvesting specialty crops such as apples (Tabb et al 2006), cherries (Tanigaki et al 2008), cucumbers (van Henten et al 2002), tomatoes (Kondo et al 1996),

melons (Edan 1995), mushrooms (Reed et al 2001), and strawberries (Kondo et al 2005) among others.

Similarly, automatic weed control is an active area of research. Åstrand and Baerveldt (2002), for example uses gray-level vision to navigate in a structured outdoor environment and color vision to distinguish crops from weeds and control a weeding tool, whereas Lee et al (1995) applies chemicals using a similar system.

Besides their application in the field, we envision precision agricultural robots to also enable gardening in urban environments, indoors — as demonstrated in this paper — or in outer space Jagannatha et al (2001).

Also, embedding sensors in the environment for monitoring crops has been studied (Wang et al 2006). For instance, in (Zhang et al 2004; Kim et al 2006) distributed wireless sensor networks for precision agriculture (e.g. irrigation) are presented. Our work aims to extend this body of research by focusing on precision agriculture tasks where autonomous mobile agents are networked with the environment. Other instances of systems that heavily rely on sensors and actuators embedded in the environment in a home companion context are (Saffiotti et al 2008), (Rusu et al 2008).

1.3 Outline

We start with describing the system architecture that consist of robots and plants networked with each other (Section 2). We will then describe the individual components of the robotic system: Navigation and Path-Planning (Section 3), Object Recognition (Section 4), Inventory (Section 5), Visual Servoing and Grasping (Section 6), and Task Allocation (Section 7). We will then present experimental results for selected components of the system in Section 8.

2 System Architecture

The distributed robot garden architecture (Figure 5) consists of robots and computationally enhanced plants (Figure 3) and includes the following subsystems :

Robots

- Notebook computer running Ubuntu Linux.
- An iRobot Create providing a bumper sensor (left/right), four cliff sensors, a wall sensor and an infra-red sensor on the robot base. The robot has been retrofitted with a rear-end caster wheel to improve odometry, has been powered by an external battery and is outfitted with a laser-cut Delrin structure to attach its peripherals. The robot is controlled from the notebook using an USB-to-Serial connection.
- Servo board (Lynxmotion SSC-32), which controls the arm, provides an analog-digital converter for the force sensor, and PWM control for the water pump using an USB-to-Serial connection.
- Water tank and pump (Hargraves) connected to the SSC-32.
- 4-DOF arm (Crustcrawler robotics) with gripper and resistive force sensor (0-10 Newton) connected to the SSC-32 (Figure 3, *right*).
- Lithium-Polymer battery pack (19.2V, 130Wh) for powering the entire system.
- Logitech Quickcam connected to the notebook using USB.
- Hagisonic Stargazer relying on markers mounted at the ceiling for global localization.
- Atheros PCMCIA radio card.

The ability to program, debug, and visualize the system on the notebook (vs. an embedded PC) drastically sped up the software engineering process. Also, USB (together with a series of TTL serial port to USB converters) proved to be an elegant solution for connecting all of the robot's components. The Crustcrawler arm does not allow for position-based feedback control, but turned out to be sufficient for proof-of-concept visual servo based grasping at a fraction (around 1/20th) of the cost of the smallest available industrial arm that we could find at the time of writing. Using a single battery pack for powering all of the system's components is motivated by the need for autonomously charging the system using a docking station. Battery power is fed directly to the notebook and regulated down using a series of switching regulators to accommodate the voltage requirements of the different platforms. The Logitech Quickcam has been selected for its low price trading off achievable frame rate (factor 3) with a Firewire camera. The indoor localization system makes the system independent of the arrangement of plants in the environment, but requires the deployment of infra-red reflecting markers at the ceiling.

Plants

- Cherry tomato shrubs of approximately 1m height that continuously carry fruits in all stages of maturity (flowers, green and red tomatoes). Every shrub grows in a dedicated pot.
- iRobot docking station, which provides an infrared field that allows a robot to autonomously dock. The dock is retrofitted to provide charging current to the 130Wh battery pack.
- Humidity sensor (Vegetronix) outputting values in the range of 0-1024, with "0" corresponding to absolute dryness.

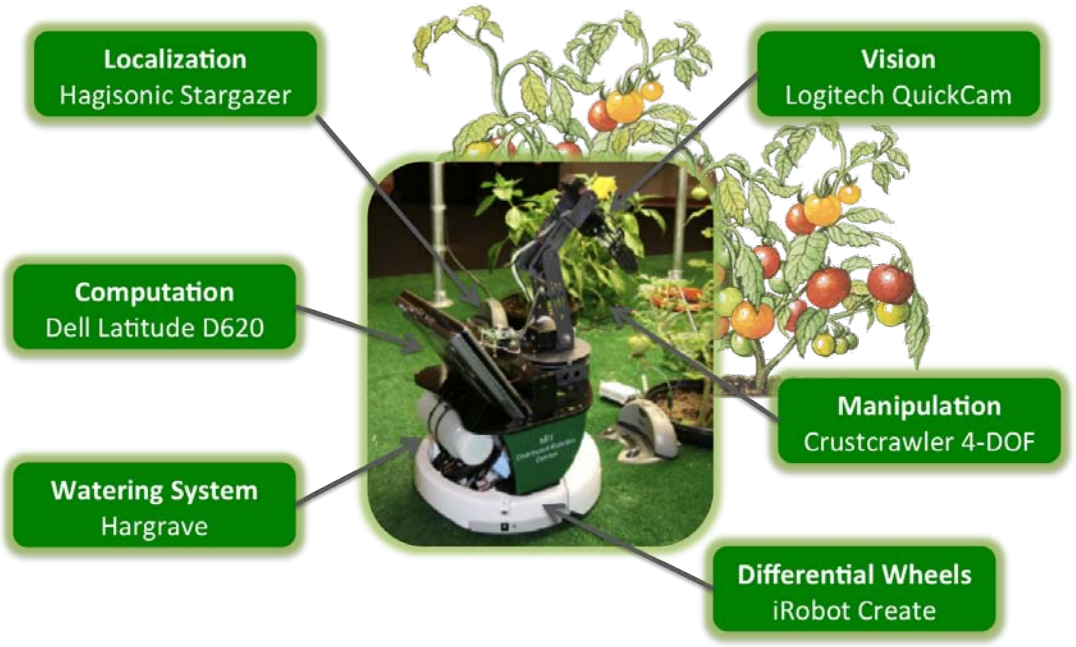

Fig. 3 Robots are equipped with a 4-DOF end-effector, a monocular camera, an indoor, global localization system and watering device that are controlled by a notebook computer. Robots coordinate with each other and intelligent pots using wireless radio.

- Wireless sensor node running OpenWRT Linux (temperaturealert.com) on an Atheros AR2315 radio-on-a-chip.
- Each plant has a unique identity provided by its IP address in the mesh network.

Embedding the plants with sensing, computation and communication abilities turned out to have numerous advantages: performing humidity sensing on the plant guarantees to respond to detect plant needs in the most timely fashion (as opposed to a mobile robot measuring each plant using a probe). Storing the result of fruit inventory performed by a robot on the plant vs. a central server provides a distributed and hence scalable and robust solution and comes at very little computational cost. Finally, coordinating plant-specific tasks by the plant itself allows to forgo a centralized task allocation system, which again contributes to the scalability and robustness of the overall system.

In our experimental setup, four potted plants are aligned in a grid on a 3×4 meters elevated astro-turf field (Figure 1). Robots and plants communicate via an ad-hoc mesh network. The plants provide the following functionality:

- A plant periodically reads its humidity sensor and posts a watering request when the reading falls below a manually-specified threshold every 10 seconds.
- Upon request, a plant reports the location and maturity (red or green) of its fruits in a local coordinate frame.
- A plant accepts the location and maturity status (red or green) of a new fruit from a robot and stores it in its local database.
- Upon request, a plant deletes a fruit from its database.

The robots have the following functionality

- Navigate to a specific plant and dock at its pot.
- Water a plant.
- Provide an inventory of a plant (fruit maturity and location in local coordinates) using its on-board camera, merging it with the current inventory obtained from the plant.
- Pick a specific fruit on the plant.

All of these functions are implemented as web services, leveraging standard tools, and rely on a TCP/IP connection to a specific robot or plant. Here it is important to understand that not the robots query plants about their status, but plants will solicity robots for services. This allows the plants to keep communication and CPU time to a minimum.

2.1 Robot Architecture

The software architecture for the robot is depicted in Figure 5. The core processes are the *Scheduler* and the *Planner*. The Scheduler's job is to coordinate and maintain a queue of high level tasks that the robot needs to accomplish, such as "Water plant #2", these tasks

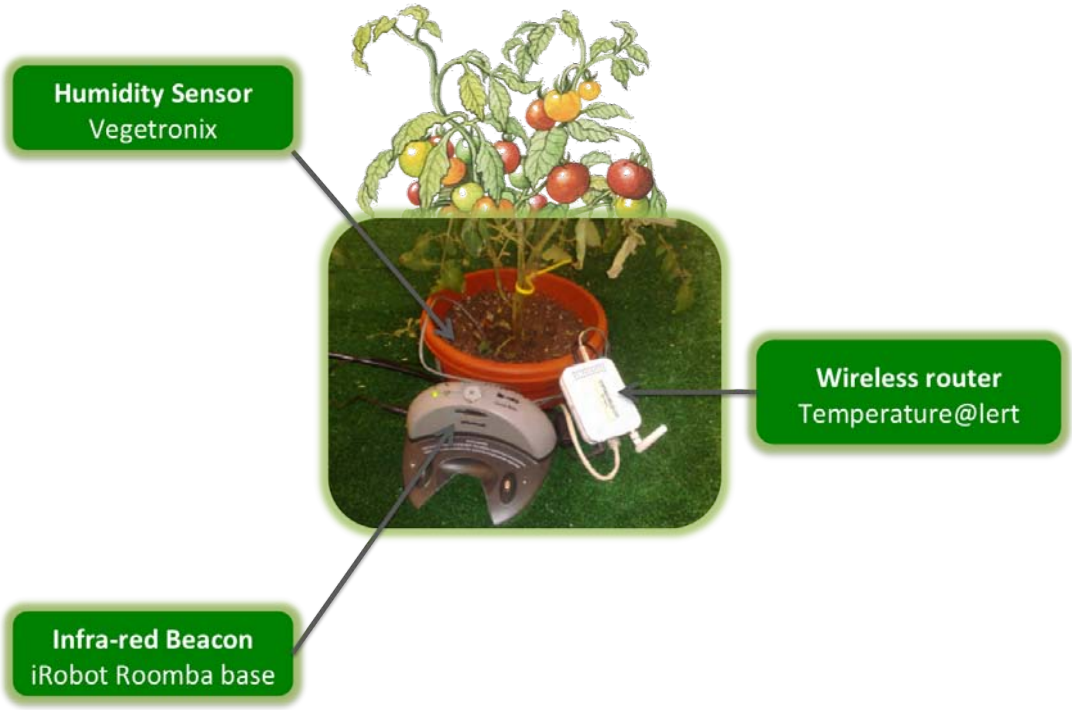

Fig. 4 Plants are enhanced with a wireless embedded Linux device that can monitor its status using a humidity sensor and information collected by the robot as well as issue requests.

can be either generated by the plants or injected by a human supervisor. For this, the scheduler receives messages from the common gateway interface (CGI) of an Apache web server that accepts HTTP requests on the local mesh-network IP.

The Planner is in charge of getting a task from the scheduler, decomposing it into a series of jobs each of which can be handled by an individual module, and then overseeing the execution of those jobs. For this, the planner interfaces with the navigation and visual servoing process as every task in our system can be decomposed into a navigation and a manipulation step.

All software modules are locally communication using the inter-process communication (IPC) framework ROS (Robot Operating System) (Quigley et al 2009), which is available open source and provides bindings for a wide range of programming languages (Python, LISP, Octave, and C) and works by exchanging custom data packets using shared memory or using a local network. ROS differentiates itself from other packages with similar intent such as LCM (Lightweight Communication and Marshalling) (Russell et al 2008) by providing a large number of open source algorithms, and visualization and management tools that can be readily integrated into ROS systems.

In their current implementations, both ROS and LCM are only of limited use for wireless inter-robot communication. We initially implemented the system using LCM as it does not require a centralized name server, which we consider preventive in a distributed system. It turns out, however, that the main bottleneck is that only little control is available where messages get routed (in fact LCM uses UDP flooding) and that UDP is unreliable as it is a connection-less protocol. We therefore implemented robot-robot and plant-robot communication primitives using web services, i.e. information is transmitted by requesting a specific document on the remote machine using the HTTP protocol. Using the common gateway interface (CGI) on a robot's or plant's web server, this approach allows for execution of arbitrary code on the target platform and to report its output via the HTTP protocol. Likewise, robots can interact with each plants using a lightweight web server on the plant (lighttpd) and the *libcurl* toolchain.

2.2 Plant Architecture

The plant's (Figure 4) functionality was implemented using the Linux *cron* daemon for periodically checking humidity status, the *wget* tool for issuing HTTP requests to the robot and the *lighttpd* web server together with a PHP interpreter to serve and update information about location and maturity of fruits. This information was stored and transmitted in JSON format, which is

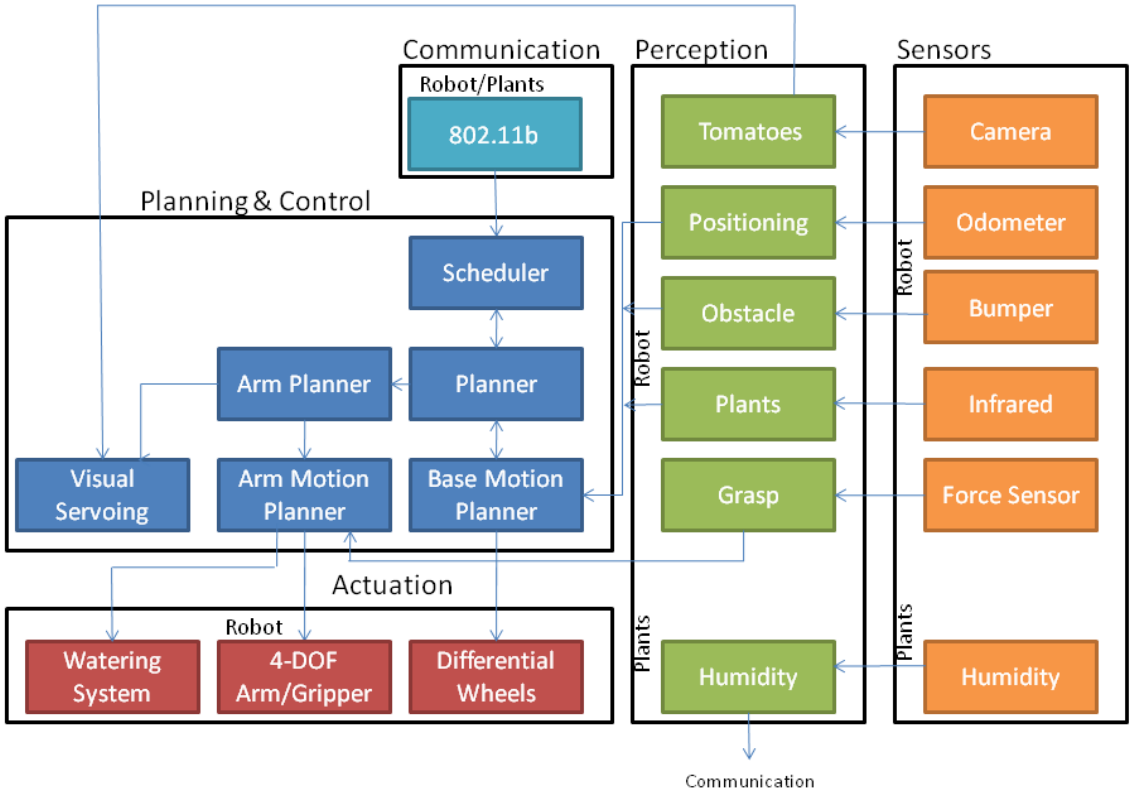

Fig. 5 Overall system architecture including robots and plants and grouped into low-level sensing, perception algorithms, high-level control, actuation and communication.

similar in intent than a XML description but provides a more compact representation for simple table data.

2.3 Network Architecture

The robots and plants establish an ad-hoc mesh network using the *Optimized Link State Routing* (OLSR) algorithm (which is available in binary form for both Ubuntu and OpenWRT). OLSR is a link-state routing algorithm. OLSR implements a series of optimizations geared toward large-scale wireless networks and has proved its performance in city-scale networks involving multiple thousand nodes. Each node floods information about its links to its neighbors until every node has discovered the other nodes and links. Each node then runs the same local shortest path computation algorithm from itself to any other node. We implement broadcasting by issuing HTTP requests sequentially to all IP addresses that are shown in the kernel routing table. Since the routing table also maintains a hop count (roughly corresponding to the spatial distribution of the nodes), this approach can also be used to address only nodes that are within 1-hop communication.

3 Navigation and Path-Planning

Robots use the basic navigation structure provided by the iRobot Create platform to travel to specific plants. It turned out that solely relying on odometry for localization is not sufficient for navigating passages that are only a few centimeters wider than the robot. In particular, using only odometry does not allow the robot for recovering from errors, as it does not provide a global reference frame. For global localization, we selected the Hagisonic Stargazer that provides relatively accurate localization and orientation ($\pm 2cm$) by detecting infrared markers mounted at the ceiling. Each marker consists of at least four reflecting stickers that are arranged in a 4x4 matrix and encode a unique ID. Knowing the position of each marker in a global coordinate frame then allows each robot to compute is position using a transformation and rotation.

The Hagisonic sensor provides position information at roughly 2 Hz whereas odometry sends updates at 50Hz. Therefore, and because we expect odometry to outperform the global localization system on short distances, position information from both sensors is fused asynchronously. Whenever new sensor data arrives, the robot either integrates odometry with full confidence or updates the position estimate using the global positioning information using an exponential average filter with 10% weight for new incoming data. We choose this policy for filtering occasional jumps in position that correspond to noise from the Stargazer sensor, which happens rarely, however. This is particularly noteworthy as the bright lights emitted by the growing system emit considerable large amounts of light in the IR spectrum.

Each robot is equipped with a configuration-space map of the environment that is provided by ROS *mapserver* component. The pots are arranged in a regular pattern, which let the system scale also for large numbers of pots. For path-planning, the navigable space of the map is discretized into a grid of $1cm^2$ cells and paths are planned using the Wavefront algorithm implemented by the ROS package with the same name. Although the Wavefront algorithm is inferior to state-of-the-art planning algorithms such as D* **?** and does not address multi-robot navigation and coordination explicitly such as Otte and Correll (2010), we have chosen it in order — in line wiht our spiral-based design philosophy — to arrive at a working system as fast as possible.

The Wavefront package also accepts updates to the local map by a laser range scanner, which is a standard message type in ROS. We are using this interface for injecting the position of other robots on the field into the map. For this, our localization node periodically (1 Hz) reads positions from other robots visible on the mesh via the HTTP protocol and emulates a laser range scanner mounted at the origin of the coordinate system.

To navigate to a plant, the robot plans a path to a point in front of the pot's docking station and then docks at the plant using the Create's built-in docking algorithm. The actual location that is required for successful docking is known by the plant and transmitted to the robot at each service request.

In case of a collision (usually due to poor localization), the robot launches a low-level avoidance behavior, which eventually leads to recover a safe position, from which the planner will re-plan.

The dock provides three infrared rays (red buoy, green buoy and an omni-directional force-field) that encode three different bytes and can thus be distinguished by the robot and provide a sense whether the dock is to the left or to the right in the direction of driving. Although the Create can detect whether it is charging at the dock, we retrofitted the dock for providing the charging source for the 130Wh battery to the robot. We therefore rely on a combination of red buoy, green buoy and force field detection for establishing whether the robot is close enough to the dock.

4 Object Recognition

Object recognition is considered one of the hardest problems in robotics and computer science. Though there are solutions to more confined environments such as "block worlds", a world where everything exists in high contrast and is polygonal shaped, the real world provides many more complications where these algorithms fail. In the garden environment, there is no guarantee that an object has an exact size or shape. Also, objects might be partially obscured or show spectral highlights due to reflection of light.

The goal of object recognition is to identify the centroid of red and green tomatoes, associate the centroid with coordinates in the plant-fixed coordinate system, communicate this location to the plant and use it for visually servo to a fruit for grasping. The location is used by the plant to maintain the fruit inventory and to give harvesting robots guidance on where the tomatoes to be harvested are.

Recognizing plants and tomatoes is a significant challenge because the foliage has complex geometry and affects lighting in unpredictable ways. The tomatoes are generally round, but their exact size and shape has many variations. The tomatoes may be partially obscured by other tomatoes or leaves. Their shiny skin gives spectral highlights due to the reflection of light.

For doing an inventory of the plant that can be stored on its wireless router, we are interested in finding both ripe (red) and unripe (green) tomatoes in an image. Because we are searching for both red and green tomatoes, color has been ruled out as discriminative factor. This does not apply for visual servoing, however, where we are solely interested in picking ripe fruits.

We investigated two distinct approaches for object recognition: feature-based, resource intensive classifiers as well as filter-based classifiers that rely on a combination of Hough circle transform (for detecting circles), the Sobel filter (for detecting smooth areas), and color filters for detecting red and green areas as well as spectral high-lights. The output of each detector was then weighted and combined to a single estimator.

Image processing routines were implemented in Matlab (for feature-based approaches) and SwisTrack (Lochmatter et al 2008), which provides a graphical user interface

to OpenCV and allows for rapid prototyping of image processing algorithms. SwisTrack was then interfaced to ROS allowing the visual servoing component and the inventory component to receive fruit locations.

4.1 Feature-based Object Recognition

Object recognition is formulated as the convolution of an image with a target pattern. If the target pattern can be found, the convolution will yield high values in this region. An extension of this approach is to use many small discriminative features, i.e. small patches that are relevant to the target pattern (Torralba et al 2004), and store the relative location of each feature with respect to the object's center. The set of features serves as a joint classifier that collectively votes for a specific location. In order to identify which features are most discriminative, Torralba et al (2004) applies a machine learning technique known as boosting. In practice, this involves taking a set of pictures with known object locations and choosing random features of the images to vote on the center of the object by convolving the feature with the target image in an offline learning step. These features are tested on other images from the training set, which allows selecting those that are most promising to detect objects in a wide range of images. If a feature turns out to be useful for a number of images in the training set, its weight is increased. If not, its weight is decreased. Figure 6 illustrates this process.

We generated a training set consisting of a large number of red and green tomatoes. We labeled the data using the online tool LabelMe [1] (Russell et al 2008). The 40 most dominant features were than extracted using the boosting algorithm (available online[2]).

After the best features and weights pairs are extracted from the training set, a convolution of a feature and the target image highlights possible locations of the class described associate with this feature. Figure 7 shows two examples with green tomatoes. As tomatoes vary drastically in size depending on the distance to the camera, we convolved the features not only with the target image, but also with a number of down-scaled versions.

Processing time of this algorithm is approx. 10-30 seconds per image, and depends on the number of features and number of scale variances used (see (Torralba et al 2004) for details on the computational complexity of the algorithm). Since the main feature of tomatoes is its relative lack of features, the algorithm achieves only around 34% success rate in identifying tomatoes. We next compare this approach to a filter-based algorithm for recognizing tomatoes.

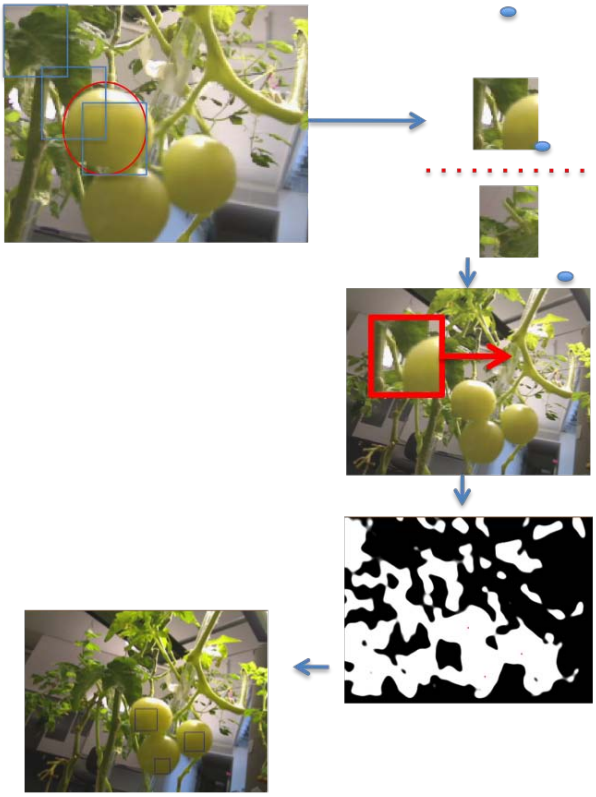

Fig. 6 Feature-based object recognition: Image with 3 quadratic features (squares in the top left image) and target object (circle). Each feature is associated with the relative position of the centroid of the target object (ellipsoids). A candidate image is convoluted with every feature. Bright areas in the gray image correspond to zones with high likelihood for the object location. Peaks in the gray image are possible object locations (squares in the bottom left image).

Fig. 7 *Left:* Result of convolution of input image with 40 feature/location pairs. Red dots correspond to local maxima. *Right column:* Detected fruits are highlighted with a rectangle. Notice the false-positive and misses in the bottom row.

4.2 Filter-based Object Recognition

This algorithm relies on shape, color, size, and spectral highlights that result from the shiny skin of the tomatoes. Possible tomato locations are given by a collective

[1] http://labelme.csail.mit.edu

[2] http://people.csail.mit.edu/torralba/shortCourseRLOC/boosting/boosting.html

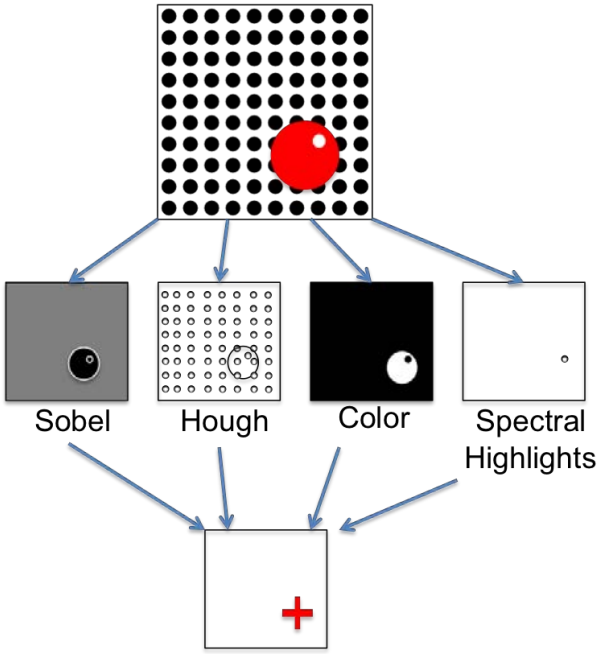

Fig. 8 Schematic representation of the filter-based object recognition algorithm. The source image (top) is filtered with Sobel, Shape, Color and Spectral Highlight filters that each yield possible tomato locations, which are then combined into a collective estimate.

estimate (weighted sum) of these filters. The gains for each filter were hand-tuned using quantitative feedback from a series of test images. The Hough transform reliably detects the tomato's outer contours, but leads to a large number of false-positives triggered by the foliage. Relying on spectral highlights (corresponding to white image patches on each fruit (but not on the leaves) is not robust with respect to changing lighting conditions. Smoothness (corresponding to dark regions after Sobel-filtering) is the most dominant feature for both red and green tomatoes. By combining this information with a constraint on the minimum and maximum area of each blob we were able to develop a tomato recognizer with high precision performance. Color is used to differentiate between red and green tomatoes. The filter-based tomato detection algorithm is shown in Algorithm 1 and illustrated in Figure 8.

Sample detection results for tomatoes of different colors are shown in Figure 9. The processing time of this algorithm was in the order of 1/3 second per image.

5 Inventory

The inventory task consists of a systematic scan of the plant to identify and store the location and color of each fruit on the plant's router. For this, the robot docks at the pot and servos its camera to six distinct locations

Algorithm 1: Tomato detector (Pseudo Code)

Data: Image frame from color video
Result: Position and color of all detected tomatoes in the frame

```
foreach frame do
    smooth = Sobelframe
    red = MinMaxColor(RedPlane(frame))
    green = MinMaxColor(GreenPlane(green))
    redTomatoes = MinMaxArea(red ∪ smooth)
    greenTomatoes = MinMaxArea(green ∪ smooth)
    ROSPublish (redTomatoes, greenTomatoes)
```

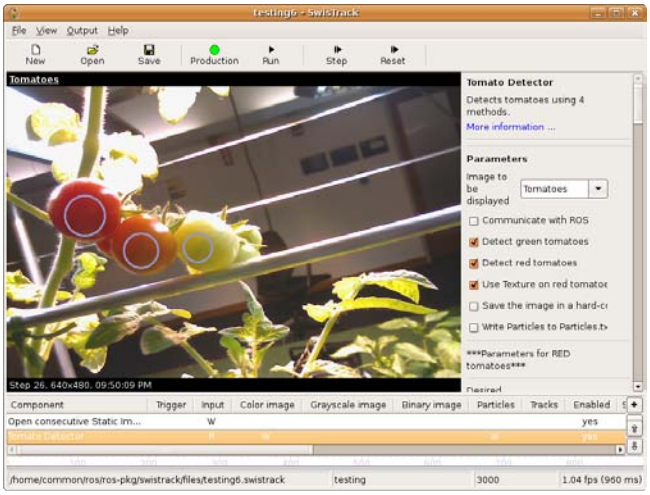

Fig. 9 Detection of red and green tomatoes using a filter-based approach in the tracking software SwisTrack.

in front of the plant (Figure 10) and runs the object recognition algorithm on each image.

If there is previous inventory, the new inventory data is used to compare and update the plant router. A tomato's description consists of: 1) the robots relative position to the plant at the time of detection, 2) the tomato's x and y image coordinates (in the scanning plane), 3) radius, 4) color, and 5) confidence level of the measurement. The confidence level is updated every time using the detection accuracy (75%, see below) such that confidence increases if a tomato's description is consistent throughout several inventories, and decreases otherwise. The inventory algorithm is given in pseudo-code as Algorithm 2.

6 Visual Servoing and Grasping for Harvesting

Harvesting tomatoes requires several steps: identification, reach, grasping, and removal. We use the output of the object recognition and inventory module as input to a visual servoing algorithm for reaching, grasping, and removing the fruit. The visual servoing algorithm

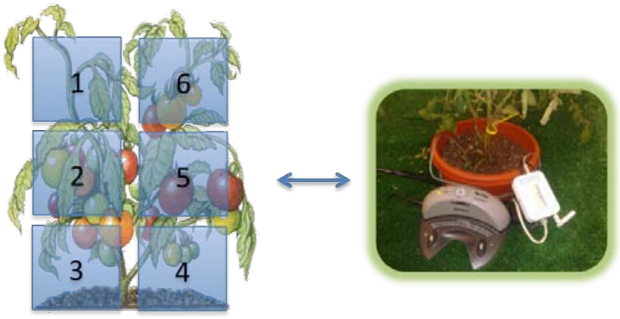

Fig. 10 Fruits and their locations are counted by having the camera arm servo to 6 distinct locations in front of the plant when the robot is docked. Data is stored on the intelligent pot as a probabilistic map that reflects sensor uncertainty.

Algorithm 2: INVENTORY (PSEUDO CODE)

```
Data: Sequence of end-effector positions for systematic
      scan
Result: Position and color of all detected tomatoes on
        the plant, distance threshold for merging
        tomatoes in previous inventories
1 foreach end-effector position do
2     moveToPosition(end-effector position)
3     tomatoes = ROSGetTomatoes()
4     if inventory available then
5         MergeInventory(inventory,tomatoes)
6     else
7         WriteInventory(tomatoes)
```

uses the color-based object recognition and approximate knowledge of the location of a tomato to drive the robot manipulator to grasp the tomato. As the locations of the fruits can only be approximately known from the inventory step using the feature-based classifier, we use a closed-loop control algorithm to align the gripper with the fruit and eventually execute the grasp. We implemented an image-based visual servoing algorithm (Chaumette and Hutchinson 2006). The motion of a robot end-effector is calculated in Cartesian space using the perceived error in the camera image.

Let $\mathbf{u} = (u, v, r)$ be the position and its radius in pixels of a tomato in an image taken by a camera mounted on the end-effector and $\dot{\mathbf{u}}$ the change induced by the camera speed $\dot{\mathbf{x}}$. The relationship between $\dot{\mathbf{u}}$ and $\dot{\mathbf{x}}$ is given by $\dot{\mathbf{u}} = L\dot{\mathbf{x}}$ in which $L \in \mathbb{R}^{3\times3}$ is the interaction matrix related to $\mathbf{u}$ and which entries can be calculated using (Chaumette and Hutchinson 2006) and experimentally. Thus, L relates a change in camera coordinates in meters ($\dot{\mathbf{x}}$) to the change ($\dot{\mathbf{u}}$) a specific pixel will undergo in pixels.

Knowing the desired location of the tomato in the image (within the gripper) $\mathbf{u}^*$ and assuming constant size of the tomatoes, we can calculate the error in image

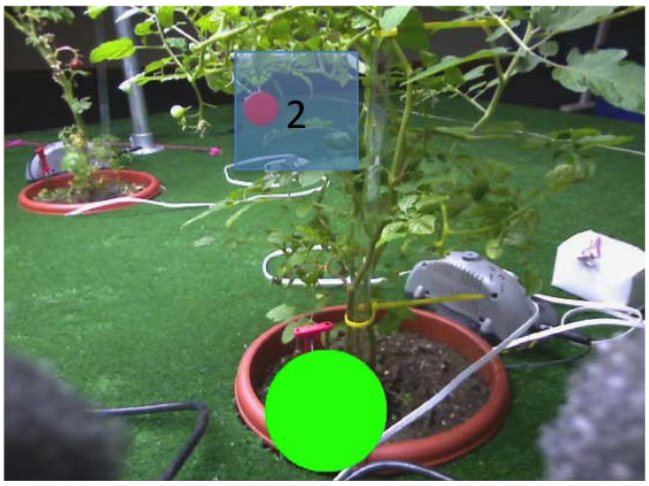

Fig. 11 Image-based visual servoing: actual fruit location within inventory location 2 (red dot) and desired fruit location within the gripper (green disc).

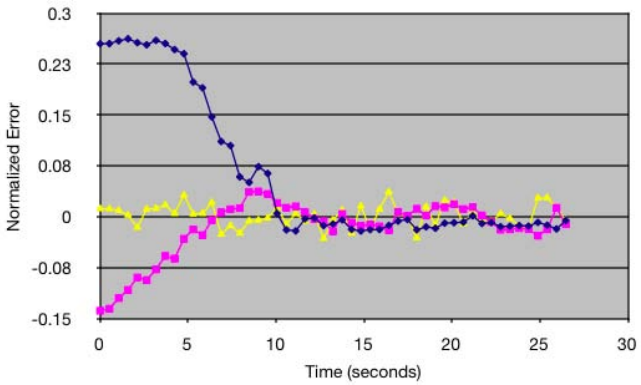

Fig. 12 Sample trajectory describing the Cartesian error (X, Y, and Z coordinates) of the robot's end-effector from its initial position to a successful grasp.

space $\mathbf{e} = \mathbf{u} - \mathbf{u}^*$ (see also Figure 11 that shows the actual (red) and desired position (green) of the tomato).

Using a feedback controller with proportional gain, we can now calculate the desired position of the end effector by the inverse of the interaction matrix. Calculating an estimate of the interaction matrix for a specific camera/end-effector configuration requires knowledge about the geometry of the camera relative to the end-effector, its focal length, and the ratio of pixel width and height, which we established using simple experiments that consisted of moving a target of known dimensions in front of the camera. Notice that the interaction matrix requires an assumption about the distance of the object. We solve this by assuming all the tomatoes to have a constant radius.

The visual servoing algorithm is summarized in Algorithm 3.

Algorithm 3: HARVESTING (PSEUDO CODE)

```
Data: Sequence of image frames, inventory
Result: Grasped fruit
selectTomato(inventory)
while !grasped do
    tomato = ROSGetTomato()
    error = goal-tomato
    if error==0 then
        graspTomato()
    else
        nextpos = ImageJacobian(error)
        moveTo(nextpos)
```

7 Task Allocation

The high-level coordination of all the activities required to maintain the tomato garden is done via a task allocation algorithm. The task allocation algorithm keeps track of the active robot tasks and requests (e.g. watering, inventory, harvesting) and allocates each task to exactly one robot. The algorithm is decentralized and aims for load-balancing among robots. Each task has a unique ID. Tasks are generated either by a user requesting a tomato, or by a plant requesting to be visited (for watering, inventory, or harvesting). For each task, a request is broadcast to each robot over the mesh-network. Robots reply with their cost of task completion. The cost is a function of the distance to the plant, the length of the task queue and the abilities of the robot (infinite cost for infeasible tasks—e.g. a robot that does not have the watering system installed will not be able to complete the watering task).

Task assignments are periodically re-allocated to ensure success even in the presence of robot delays and failures (see also (Amstutz et al 2009) for a similar scheme and its analysis). An example task allocation process for two robots and two plants is depicted in Figure 13. Time increases downward in the graph.

8 Results

The algorithms described have been implemented in C, C++, PHP and Python and are interconnected by ROS messages and web interfaces. We evaluated the reliability of each robot operation, the reliability of the plant sensory-computational system, the ability of the system to coordinate multiple robots, and the effectiveness of the task allocation algorithm with up to 2 robots. Although we developed 6 robots within the course of the project (one for each student team), we chose to conduct experiments with only 2 robots as our current experimental setup is limited to 4 plants due to space

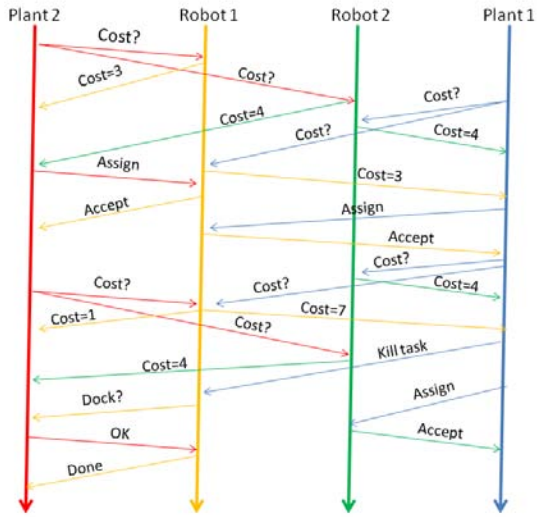

Fig. 13 Task allocation scheme for a two robots, two plant setup. Plants select robots that can offer a task at the lowest cost (time). The cost at which a robot can do a task depends on the length of its task queue and its position. Plants periodically re-evaluate offers and re-assign tasks. For example, robot 1 updates the cost of a task that was previously assigned from plant 1 from cost "3" to"7". As plant 1 knows about robot 2, which performs the task for cost "4", a "kill task" message is sent to robot 1 and the task is reassigned to robot 2.

reasons, and we would like to demonstrate division of labor and coordination, rather than a one-robot-per-plant scenario.

8.1 Networking

We evaluated the data rates in the network of robots and plants and quantified the effects distance and high network load had on the messages that were sent.

In this experiment, a message was judged "successful" if it was sent and received properly. To see what effect distance had, we had one computer send messages to the routers at a distance of 1m, 13m, and 27m. To measure the success rate of messages sent under high load, we had 4 laptops next to each other, all sending messages, at a distance of 1m from the routers.

Both plants and robots use the same wireless network architecture. Thus, the metric we measure affects both plant-robot coordination as well as robot-robot coordination. An example of such a coordination and the typical number of packets that are exchanged during this process are shown in Figure 13.

Overall, our experiments suggest that as long as the robots are within the experimental platform, it is reasonably to expect almost all of our messages ($> 95\%$) to be sent and received within 0.05 seconds even under

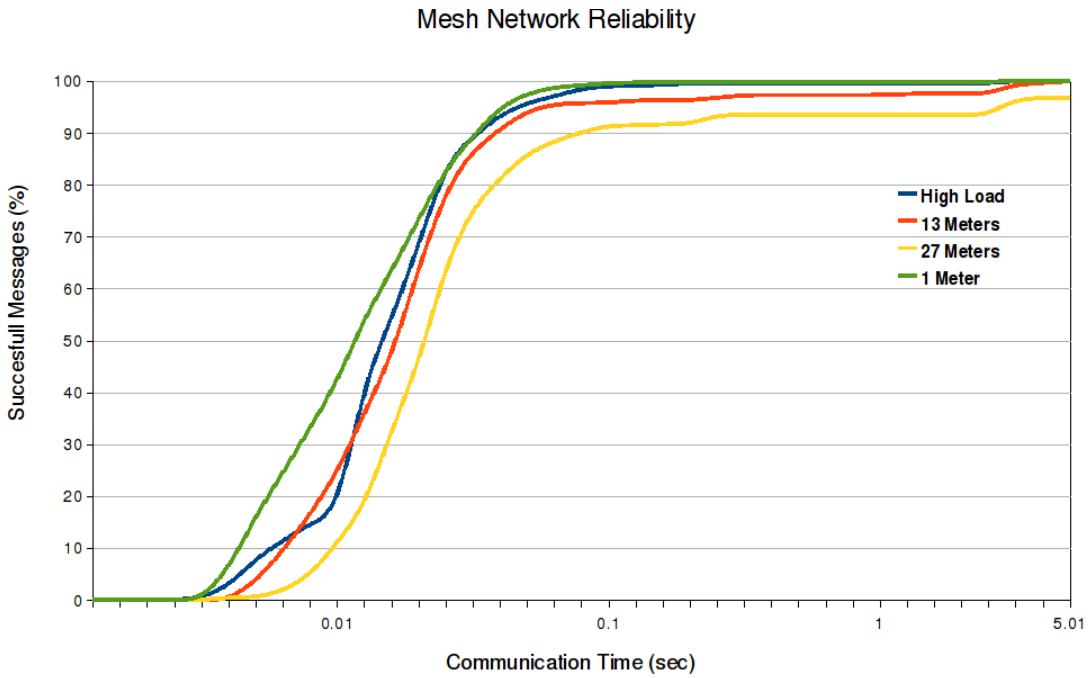

Fig. 14 Delay distribution for packet transmission over the ad-hoc network as a function of the distance and network load.

high network load (Figure 14), which enables robots and plants to coordinate in less than a second.

We also demonstrated a sequence of experiments in which the task allocation algorithm controls two robots in parallel. In this experiment, each robot was tasked asynchronously to go to a different plant. Once docked at the plant, the task of the robot (e.g. watering or harvesting) was done according to the results described below.

8.2 Navigation and Watering

In order to evaluate the navigation performance of our system, we required a robot to loop in a square of 85cm side length. The robot's position was consistently off by around 10% over 10 iterations. Odometry performed worse than expected from the specifications of the iRobot Create base due to the additional weight of arm, notebook and watering system. Even though the astro-turf ground provides sufficient traction to the robot's wheels, it provides significant friction to the front and rear casters, which particularly impacts the accuracy of turns. By adding the external localization system, the performance of the navigation algorithm incurred no accumulated errors over 20+ loops along the test square. Whereas the reactive algorithm to reach the dock compensated for errors in waypoint navigation, odometry was not reliable enough for repeatedly navigating through narrow passages in the arena. In the absence of global positioning information, getting stuck between the plants or supporting poles eventually leads to a total loss of orientation, which has been remedied by the Stargazer sensor.

In rare cases, however, navigation error lead to an orientation in front of the plants in which the robot was attracted by a neighboring pot when docking. This is mitigated by the planner which supervises that the robot stays within a rectangle of 1 by 1.5m around the dock and navigates to the original launching point otherwise.

We evaluated navigation for watering by starting a single robot in the top-left corner of the arena. We tasked the robot with watering two different plants, one closer to the origin, 6 and respectively 4 times and solely relying on odometry. Watering takes approximately 20s and involves moving the arm toward the soil using open-loop control. The average navigation time to reach the closer dock was $79.5s \pm 11.3$ and 94.25 ± 8.9 for the dock farther away. We observed 100% success rate of watering over 100+ trials.

8.3 Object recognition and Inventory

We trained classifiers using the boosting algorithm (Torralba et al 2004) for red and green tomatoes using a training set of 17 images that were labeled using (Russell et al 2008). We then tested the classifiers on live images from the robot. Over 150 images, each containing at least one tomato, the classifiers detected and

correctly classified around 34% of the tomatoes and 38% false-positives. We also counted the total number of tomatoes that were present in the images and calculated the percentage of tomatoes that the algorithm did not recognized at all to 44%. Of these 44% misses, 87% were expected as the tomatoes were occluded or cropped, for which we did not train appropriate classifiers. Note that these results are including both red and green tomatoes as the extracted features do not encode color.

As the detector convoluted 15 scaled instances of each image with the 40 most important features, processing one image requires approximately 16s per image on the Centrino Duo notebook. This makes the feature-based algorithm unsuitable for real-time tracking, which is required for visual servoing and grasping.

Using the Sobel-filter based tracker in Algorithm 1, we improved the rate of correctly classified tomatoes to 75%, although focussing exclusively on red tomatoes.

Inventory took an average of around $45s$ over 10 trials.

8.4 Visual Servoing and Grasping

We conducted the following experiment ten times in a row using both the feature-based and filter-based detection approaches. The robot starts in a docked configuration with a plant. The robot is given the coordinates of a tomato and moved its arm from the docking configuration to a position close to the tomato (given its location stored on the plant). The robot then uses the visual servo feedback controller to grasp the tomato.

Using the feature-based approach, grasping was successful in 50% of the time over 10 trials. The average duration for the successful trials was $30.6s \pm 16.1$ and 30.8 ± 31.6 for the unsuccessful trials.

The filter-based approach improved the performance of the visual servo to 75% successful grasps over 20 trials. The average time was $28.3 \pm 10s$ for the successful trials. The success rate of this algorithm is closely related to the object recognition performance. In case the robot started in a position that did not lead to any of the expected detections, the visual servo immediately terminates.

Reasons for failure were mostly due to singularities on the trajectory from initial position to desired grasping location and false-positives on object recognition that lead to premature grasping. The force sensor helps detecting failed grasping attempts for false-positive detections. It does not help, however, when the gripper grasps another part of the plant, which we also observed.

We also observed several instances in which the gripper was protruded by branches and leafs. As the plant is flexible, this often led to movements also of the fruit itself. While such movement could be partially mitigated by the visual servo, this often leads to a "deadlock" where the fruit becomes unreachable. In this case additional sensing would be needed to detect such a situation as well as to better understand the geometry of the branch structure and its location in space.

9 Discussion

Networking. OLSR will scale for thousands of nodes. However it might make more sense (for this particular application) to forgo routing altogether and simply flood the requests into the network with a limited hop count. Although IEEE 802.11b allowed us to leverage off-the-shelf software and protocols, a less resource intensive, short range communication system such as infrared might be more appropriate. As a side-effect the infrared signals could also be leveraged for navigation.

Plants. The system performance can be enhanced by using a model of plant growth. This will allow plants to predict their status in between updates received from robots and it is part of our current work.

Navigation and Path Planning. If global positioning is available, the gardening system can recover from navigation errors as described in this paper. If global positioning is not available, robots with better odometry and navigation sensors are needed. Although we employed a global positioning system that also makes the system independent from the structure of the environment, a simpler solution in an environment with matrix arrangment of plants — such as on a field — would be to encode the plant id on the docking signal to re-locate the robot.

Object Recognition. Object recognition using joint boosting is resource intensive and is difficult in the gardening domain. A key problem is that texture-less tomatoes provide only very limited features by themselves. Thus, dominant features are mostly located at the boundary of tomato and background, which leads to poor performance when the background varies strongly as in our experiment (astroturf, leaves, other robots, laboratory). An advantage of a feature-based method is that its success rate is independent of color, i.e. red and green tomatoes are treated equally.

Detecting green tomatoes using a filter-based approach is much harder, however. Although the Hough transform (detecting circles) and the extrema detection for spectral highlights provide strong clues for the detection of green tomatoes, tomatoes are still hard to

differentiate from leaves that exhibit similar features. For this reason, only red tomatoes — using color as the final discriminant — can be detected reliably using a filter-based approach.

Visual Servoing. Visual servoing fails when the robot is unable to detect the tomatoes at the expected position. This situation can be improved by implementing an additional planning step using position-based visual servoing, which systematically explores the region of interest from different angles, potentially also involving movement of the base. The performance of the visual servo would also been drastically improved when using stereo vision, which allows for depth estimation.

Grasping. As the orientation of the stem growing out of the tomato fruit is not always vertical it turns out that we could only grasp a subset of tomatoes without an additional degree of freedom (wrist rotation) and appropriate image detection algorithms. Also, the limited workspace of the arm imposes constraints on reachable tomatoes, and an arm that provides the full six degrees of freedom would be necessary to reach tomatoes in all possible orientations. Finally, performing collision-free grasps might require a complete 3D reconstruction of the branch structure as pushing branches aside will lead to undesired motion of the entire plant, which makes fruit collection infeasible in some cases.

Scalability The distributed architecture of the system should theoretically ensure the scalability of the systems for large numbers of robots and plants. In particular, all information is stored on the plants and each plant stores only its own information and robots interact only with plants that solicit them and those that are within 1-hop communication radius for collision avoidance. In practice, however, a large-scale deployment must ensure that robots are evenly distributed in the environment and that plant requests can always reach at least one robot via multi-hop communication (see the discussion on networking above). At the same time, additional coordination between robots would be required to assure equal coverage of the environment even if individual robots fail, e.g. using the approach described in Schwager et al (2009).

10 Conclusion

This paper describes our experience with designing and building a distributed robot garden. We have developed a network of mobile manipulators and plant sensor networks. We demonstrated that our system can coordinate plant requests and robot activities for precision plant watering, fruit inventory, and fruit harvesting in a fully distributed way. Open problems are the robustness of the operations provided by the hardware and the limited workspace of the chosen arm. Particular challenges are global localization for recovering from navigation errors and the limited robustness of the state-of-the art vision algorithms to changing lighting conditions of object recognition during visual servoing.

Our current focus and challenge in this project is achieving persistent autonomous operation of the distributed gardening robots for periods on the order of several weeks, which requires overcoming challenges in power autonomy, self-charging and improved manipulation capabilities. In the long run, we are interested in developing algorithms and mechanisms that can physically change the location of a plant in order to autonomously distribute plants on the landscape that maximizes growth and minimizes resources. Also, we would like to study the relation between plants and various pests — in particular those which are known to have a repellent affect — and leverage this plant-to-plant interaction by controlled deployment and mobility to minimize pesticides.

Acknowledgements This work was supported in part by the Swiss NSF under contract number PBEL2118737, MURI SWARMS project W911NF-05-1-0219, NSF IIS-0426838, EFRI 0735953 Intel, and by the MIT UROP and MSRP programs. We are grateful for this support. This paper is partly the result of a challenge class" taught at MIT (6.084/85) by the authors.

We would like to thank A. Torralba for his help on feature-based object recognition, J. French, J. Myers and A. Zolj who have been working on the Distributed Robotics Garden as part of the MIT Summer UROP program in 2008, Kevin Quigley and Marsette Vona for providing their visual servoing implementation, and Michael Otte for helping out on navigation.

References

Al-Kufaishi S, Blackmore B, Sourell H (2005) The potential contribution of precision irrigation to water conservation. In: Proceedings of the 5th European conference on Precision Agriculture, pp 943–950

Amstutz P, Correll N, Martinoli A (2009) Distributed boundary coverage with a team of networked miniature robots using a robust market-based algorithm. Annals of Mathematics and Artifcial Intelligence Special Issue on Coverage, Exploration, and Search 52(2–4):307–333

Åstrand B, Baerveldt AJ (2002) An agricultural mobile robot with vision-based perception for mechanical weed control. Autonomous Robots 13(1):21–35

Blackmore B (2007) A systems view of agricultural robots. In: Proceedings of the 6th European Conference on Precision Agriculture, pp 23–31

Blackmore B, Fountas S, Gemtos T, Vougioukas S (2006) Ecobots: Improved energy utilisation through smaller smarter machines. In: Proceedings of the 3rd IFAC International Workshop on Bio-Robotics

Chaumette F, Hutchinson S (2006) Visual servo control part i: Basic approaches. Robotics & Automation Magazine 13(4):82–90

Correll N, Arechiga N, Bolger A, Bollini M, Charrow B, Clayton A, Dominguez F, Donahue K, Dyar S, Johnson L, Liu H, Patrikalakis A, Robertson T, Smith J, Soltero D, Tanner M, White L, Rus D (2009) Building a distributed robot garden. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), St. Louis, MO, pp 1509–1516

Edan Y (1995) Design of an autonomous agricultural robot. Applied Intelligence 5(1):41–50

van Henten E, Hemming J, van Tuijl B, Kornet J, Meuleman J, Bontsema J, van Os E (2002) An autonomous robot for harvesting cucumbers in greenhouses. Autonomous Robots 13(3):241–258

Jagannatha S, Levesque A, Singh Y (2001) Approximation-based control and avoidance of a mobile base with an onboard arm for mars greenhouse operation. In: Proceedings of the 2001 IEEE International Symposium on Intelligent Control, pp 103–108

Jimenez A, Ceres R, Pons J (2000) A survey of computer vision methods for locating fruit on trees. Transactions of the ASAE 43(6):1911–1920

Kim Y, Evans R, Iversen W, Pierce F (2006) Instrumentation and control for wireless sensor network for automated irrigation. ASAE Paper No 061105 St Joseph, Michigan

Kondo N, Nishitsuji Y, Ling P, Ting K (1996) Visual feedback guided robotic cherry tomato harvesting. Transactions of the American Society of Agricultural Engineers (ASAE) 39(6):2331–2338

Kondo N, Ninomiya K, Hayashi S (2005) A new challenge of robot for harvesting strawberry grown on table top culture. In: Proceedings of ASAE Annual International Meeting, Tampa, Florida, p Paper number: 053138

Lee W, Slaughter D, Giles D (1995) Robotic weed control system for tomatoes. Precision Agriculture 1(1):95–113

Lochmatter T, Roduit P, Cianci C, Correll N, Jacot J, Martinoli A (2008) Swistrack - a flexible open source tracking software for multi-agent systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nice, France, pp 4004–4010

Otte M, Correll N (2010) The any-com approach to multi-robot coordination. In: IEEE International Conference on Robotics and Automation (ICRA), Workshop on Network Science and Systems Issues in Multi-Robot Autonomy (NETSS), Anchorage, AK, USA

Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng A (2009) Ros: an open-source robot operating system. In: International Conference on Robotics and Automation, Workshop on Open-Source Robotics, Open-Source Software workshop

Reed J, Miles S, Butler J, Baldwin M, Noble R (2001) Automatic mushroom harvester development. Journal of Agricultural Engineering Research 78(1):15–23

Russell B, Torralba A, Murphy K, Freeman W (2008) Labelme: A database and web-based tool for image annotation. International Journal of Computer Vision 77(1–3)

Rusu R, Gerkey B, Beetz M (2008) Robots in the kitchen: Exploiting ubiquitous sensing and actuation. Robotics and Autonomous Systems, special issue on Network Robot Systems 56:844–856

Saffiotti A, Broxvall M, Gritti M, LeBlanc K, Lundh R, Rashid J, Seo B, Cho Y (2008) The peis-ecology project: Vision and results. In: Proc. of the IEEE/RSJ Int Conf on Intelligent Robots and Systems (IROS), Nice, France, pp 2329–2335

Schwager M, Rus D, Slotine JJ (2009) Decentralized, adaptive coverage control for networked robots. International Journal of Robotics Research 28(3):357–375

Srinivasan A (ed) (2006) Handbook of Precision Agriculture: Principles And Applications. CRC Press

Tabb A, Peterson D, Park J (2006) Segmentation of apple fruit from video via background modeling. In: Proceedings of the American Society of Agricultural and Biological Engineers ASABE Annual International Meeting, Portland, OR, 63060, p 11 pages

Tanigaki K, Fujiura T, Akase A, Imagawa J (2008) Cherry-harvesting robot. Computers and Electronics in Agriculture 63(1):65 – 72, special issue on bio-robotics

Torralba A, Murphy K, Freeman W (2004) Sharing features: efficient boosting procedures for multiclass object detection. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp 762–769

Wang N, Zhang N, Wang M (2006) Wireless sensors in agriculture and food industry – recent development and future perspective. Computers and Electronics in Agriculture 50(1):1–14

Zhang W, Kantor G, Singh S (2004) Integrated wireless sensor/actuator networks in agricultural applications. In: Proc. of ACM SenSys, p 317