Stony Brook University

Computer Vision-based Robot Tracking and Navigation for the MINT Testbed

A Thesis Presented

by

ARVINDHAKSHAN MADHAVAN

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Computer Science

Stony Brook University

December 2009

Stony Brook University

The Graduate School

Arvindhakshan Madhavan

We, the thesis committee for the above candidate for the

Master of Science degree, hereby recommend acceptance of this thesis.

Professor Tzi-cker Chiueh - Thesis Advisor

Department of Computer Science

Professor Samir Das - Thesis Committee

Department of Computer Science

Professor Jie Gao - Thesis Committee

Department of Computer Science

This thesis is accepted by the Graduate School

Lawrence Martin

Dean of the Graduate School

Abstract of the Thesis

Computer Vision-based Robot Tracking and Navigation for the MINT Testbed

by

ARVINDHAKSHAN MADHAVAN

Master of Science

in

Computer Science

Stony Brook University

2009

A platform for testing wireless protocols before they can be deployed on a large-scale has always been a challenge. A simple approach is to use software simulation. But accurate modeling of real-world characteristics like propagation and interference of radio channels is extremely difficult in software and the fidelity of this approach is questionable. On the other end of the spectrum, the protocol can be tested in large-scale custom test bed. Although the result of the testing is accurate, the cost of the setup is high and reconfiguration is difficult. Mint-m is a miniaturized wireless test bed proposed at Stony Brook University, which addresses this problem by providing a cost-effective and reconfigurable wireless test bed for testing generic protocols. It combines the advantages of above approaches by providing an accurate yet affordable platform for testing wireless protocols.

Support for mobility is an important characteristic of a wireless test bed. Mint-m supports mobility by mounting test bed nodes on top of robots. The tracking of the movement and control is achieved by using overhead cameras. Mint uses color patches on top of the robots for node tracking and address the problem of collision detection by using the cameras to detect the obstacles.

In this thesis, we propose a refinement of Mint's mobility system, which improves the accuracy of the node positioning and tracking and enables accurate movements during simulation. The importance of movement accuracy is amplified by the very nature of miniaturization. A small change in accuracy of movement in test bed implies proportionally very large change in the system it models. To address this problem, we re-design the optical tracking system by using a SIFT (Scale Invariant Feature Transform) based node recognition system and propose different algorithms for movement and proactively avoid collisions instead of collision detection in the original design. We also add capability for movement during experiments.

To

My Parents & my dear Sister

Table of Contents

List of Tables

List of Figures

Acknowledgements

First of all, I would like to thank Professor Tzi-cker Chiueh for the opportunity to work under him and his constant guidance and support through the course of the thesis. I am always awe-struck at his ability to provide instant solutions to complex problems. I would also like to thank Jui-Hao Chiang for his constant support during the implementation phase.

Chapter 1

Overview of MiNT

This chapter presents a brief overview of the MiNT-m wireless test bed. We explain the MiNT-m architecture briefly with specific emphasis on the MiNT-m's mobility system.

1.1 MINT Wireless Testbed

MiNT [1], a miniaturized mobile multi-hop wireless network testbed, is a generic network platform that can be used to test arbitrary network protocols and was developed at Stony Brook University. MiNT achieves the accuracy of a large scale wireless testbed without the associated hassles involved like the cost of setup, large physical area required and the difficulty in re-use and reconfiguration.

MiNT also introduces a novel concept of hybrid NS-2 simulation. MiNT can run majority of NS-2 scripts without any changes in the read-world setup. MiNT achieves this by using the NS-2 scripts until the network layer and replacing the simulated Data Link, Medium Access control and Physical Layers with the actual layers in the testbed. This hybrid mechanism adds more power and flexibility to the testing wireless protocol implementations.

MiNT testbed consists of multiple testbed nodes each fitted with multiple (typically 4) wireless interfaces. These nodes can communicate with each other over one or multiple hops. Nodes are the heart of the MiNT system and this is where simulations are run.

Typical size of a MiNT testbed fits into a room. The miniaturization is supported by attenuating the signal strength of the wireless cards in senders and the receivers. This feature allows the testbed size to be as small as the size of the room. A typical range for communication between nodes is 4 feet.

MiNT supports topology reconfiguration by allowing the user to control signal characteristics between nodes. The reconfiguration is achieved by placing nodes across the testbed such that the signal characteristics are at required level between each and every pair of nodes. This flexibility in re-configuration in addition to the easier setup and low cost makes it an ideal platform for evaluating wireless protocol.

MiNT supports mobility of nodes by mounting these nodes on top of a Robot. For minimizing the cost of the setup, MiNT uses iRobot vacuum cleaners with programmable interface as robotic platforms. The node movement is tracked by using overhead cameras.

MiNT also supports a visual control interface MOVIE, Mint cOntroller and Visualization InterfacE, which can be used to monitor and control the experiment nodes.

1.2 MINT System Architecture

MiNT system comprises of wireless testbed nodes, which are controlled by a central controller. The position and location of the nodes within the testbed are tracked by a component called Tracking server. The software and hardware components of MiNT are explained in this section.

Testbed Nodes

MiNT testbed nodes are made of low power computing devices namely RouterBoard RB-230. Each of these nodes is fitted with 4 mini IEEE 802.11 a/b/g wireless cards which allow multiple experiments to be run simultaneously. The mobility is achieved by mounting the node on top of an inexpensive

Vacuum cleaner robot from the iRobot. The node controls the movement of the roomba, with appropriate inputs from the tracking server and the control system.

The software component of the node comprises of a noded daemon, which is central control in the testbed nodes. Noded acts as the platform where Ns-2 experiments are supported. The NS-2 component layer in the noded, uses the underlying LLE, Link Layer Emulation component of the noded to send and receive packets over the wireless cards fitted with the nodes. The noded also has a monitor agent, which captures all packets in the network in promiscuous mode and reports it to the controller, where time-stamp based arrangement and duplicate pruning is performed to give a global picture of interference and signal strength characteristics. The noded daemon also interacts with the central controller daemon and translates the movement commands from the controller into the serial commands for robot.

Controller

Controller is a PC fitted with multiple wireless cards for control communication with the wireless nodes. The controller runs a daemon called controld, which acts as the central module for the entire testbed. The controld is responsible for interacting with the noded and getting the testbed simulation output and interfaces with the MOVIE interface to display the trace output. It is also responsible for relaying the control commands from the MOVIE interface back to the noded.

Controller is also responsible for the mobility of the wireless nodes. The controller co-ordinates with the tracking server, which is connected with overhead cameras, and gets the positions of node represented as (x,y) co-ordinates within the testbed and the orientation angle with respect to the x-axis of the testbed. It uses these inputs to control the trajectory planning and collision avoidance of the moving nodes.

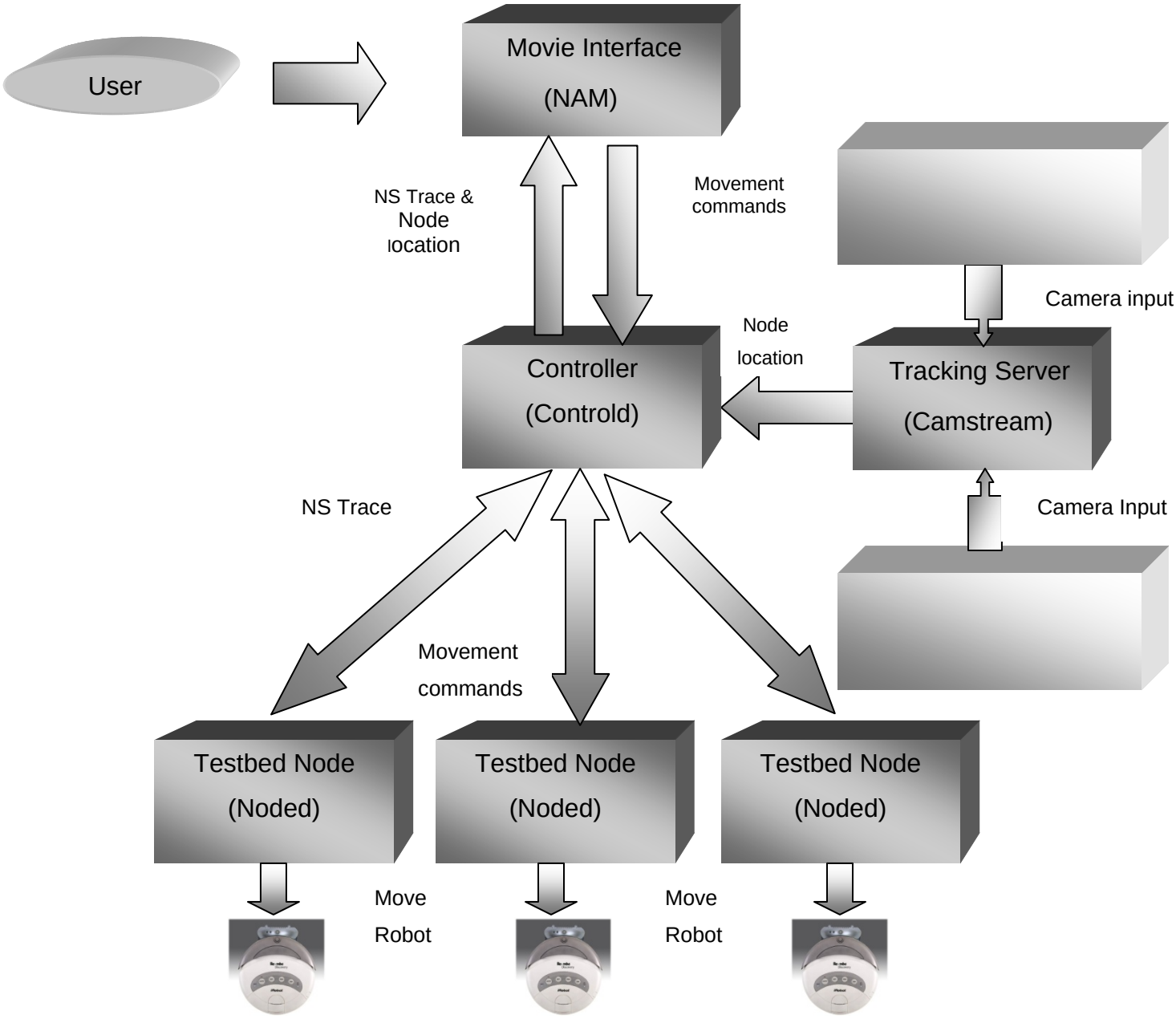

Figure 1 - Mint Architecture

Tracking Server

Tracking server is again a PC fitted with multiple off-the-shelf webcams capable of producing images at resolution 320X240. The tracking server daemon uses open source software camstream to capture images from the cameras. The nodes are fitted with multi-colored patches for identification. The tracking server daemon uses the color patches from the camera images to identify and locate the node. The position and orientation within the camera is calculated in pixels, and then the data is converted into a location in the global testbed. The tracking server continuously tracks images from all the cameras and provides these inputs to the controller for trajectory planning.

MOVIE – Mint Controller and Visualization Interface

This is the GUI (Graphical User Interface) controller interface for the MiNT testbed. Movie is an extension of NAM[2], which is a popular tool for visualization of the NS-2 traces. The NAM based MOVIE interface is modified to display the entire testbed along with the pair-wise signal strength and interference characteristics for nodes, in addition to displaying the trace information. This also acts as a platform through which user can configure different wireless topologies, by moving and placing the nodes appropriately, such that required signal and interference characteristics between nodes is setup.

The block diagram Figure 1 - Mint Architecture summarizes the architecture of MiNT.

Chapter 2

Introduction

Mobility is an important feature of MiNT wireless testbed. The Mobility in MiNT is currently used mainly for placement of nodes within the testbed such that different topological configurations are achieved. The user will use the MOVIE control to interactively move the testbed nodes such that signal and interference characteristics between each pair of nodes are at the required levels.

The Miniaturization aspect of MiNT introduces great challenges in accuracy of node placement. An entire MiNT testbed consisting of 12 nodes would typically fit inside a room of size *14 feet X 12 feet*. This greatly reduces the area available for movement of nodes. Moreover, the signal attenuation of senders and receivers means, the typical distance between communicating nodes is 4 feet. Nodes at a distance larger than 4 feet will be out-of communication range with each other.

Given these small distances, it is imperative that the node mobility is accurate to centimeters. Although MiNT supports mobility, the accuracy of MiNT node placement can be vastly improved. In this thesis we propose we propose a refinement of MiNT mobility system, which overcomes many of the shortcomings of the existing MiNT mobility system. We propose a complete re-design of the MiNT tracking system, using SIFT [2] (Scale Invariant Feature Transform) algorithm. We also improve the trajectory planning system with additional feedback from the robotic nodes and add proactive collision avoidance to Mobility system, which currently supports restrictive and reactive collision prevention.

Chapter 3

Deficiencies of MINT's Tracking and Navigation Subsystem

3.1 Current Design

In this chapter we present a brief overview of the design of the MiNT-m's tracking and navigation system and discuss the shortcomings of the systems.

3.1.1 MiNT Tracking System

MiNT-m currently uses a color based tracking system. The testbed arena consists of 6 cameras mounted on the ceilings. The tracking server is the component responsible node position localization. It uses the input feeds from these cameras to aid in the localization process. The Figure 2 - Testbed Node - Color code shows a node with the color patches mounted at the top.

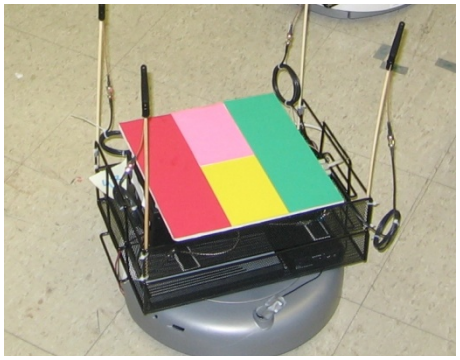

Figure 2 - Testbed Node - Color code

The node has a header patch in color red, tail patch in color green. These colors are common to all nodes and serve as the unique identifier of a node pattern. The colors in the middle (pink and yellow) in this picture serve as the unique identifier for each node. Currently there are 4 colors in use in the

testbed namely blue, yellow, orange and pink. Together these 4 colors can be useful in setting up 16 unique identifiers.

During the experiment setup, the HSV (Hue, Saturation and Value) low and high range for each color in use in the testbed is identified. These values are used during tracking to identify the color patches. First the images from the camera are normalized to reduce the impact of lighting changes. Then these images are scanned pixel by pixel, and are grouped as smaller blobs with same pixel color. These smaller blobs in close proximity are then combined to bigger blobs. A small allowance for combining smaller blobs are required because, some of the pixel values in the middle of bigger blob may be outside the lower or upper boundary of the HSV value and might result in isolation of smaller blobs.

MiNT-m solves the tracking issues in the camera borders by setting up a camera overlap in the borders. This overlap is required to ensure that a node is completely visible in the at least one camera. i.e. a overlap as big as the diagonal of the node is setup. The overlap is illustrated in Figure 3 - Camera Overlap. The overlap eliminates the need other sophisticated mechanisms like stitching algorithms which may be needed otherwise.

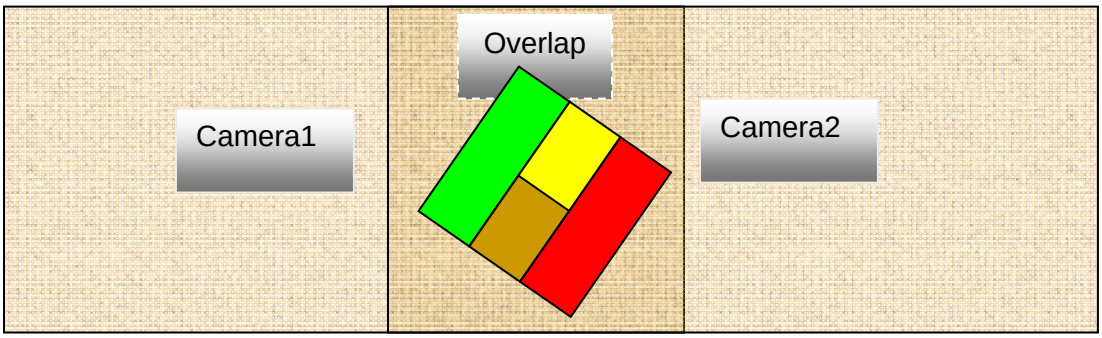

Figure 3 - Camera Overlap

During real-time tracking, images from each camera are scanned for color blobs. Once a blob is located, its center's position within camera in (x, y) pixels and orientation with respect to the camera are computed. As each camera's position in the testbed and pixel to inch ratio (a measure of how many

pixels in the image represent an inch in the testbed) of the camera images, are known, (x, y) coordinates in inches within testbed and orientation with respect to the x-axis of the testbed can be computed.

3.2 Deficiencies

This section outlines the deficiencies in the MiNT tracking and navigation system. We discuss the flaws in the tracking system following by the corresponding defects it creates in the navigation system.

3.2.1 Tracking Deficiencies

The color patch based recognition system, although simple and straight forward, introduces inaccuracy in the tracking of images. It is very difficult to identify and accurately tune the values for each color. Allowing too less an allowance for low and high HSV values would result in blobs not being recognized. Allowing bigger allowances would result in false matching for colors in close range. For example red and pink almost have similar values for hue (under the standard lighting in the experiment setup), but differ very slightly in the saturation and value measures. It is very difficult to accurately differentiate between these colors. Allowing too much allowance would result in pink and red being interchanged or allowing too less allowance can result in blobs not being detected.

Although this problem, to certain extent, can be solved by choosing non-overlapping colors from color space, this greatly reduces the scalability in the number of colors that are available and hence the number of unique identifiers for nodes.

A more challenging aspect of the color based detection is the changes in the HSV values for colors, with little changes in the lighting in the testbed arena. Although the testbed arena is insulated from the natural sun light, the lighting in the room slightly varies through the day in the arena. This combined with other factors like changes in the lighting capacity of the individual ceiling

lights and fading in the color patches, can significantly affect detection of the nodes. Hence it is very hard to calibrate accurate values for the testbed setup, such that the tracking will be accurate at all times.

As the detection is blob based, even small changes in color values can result in skewed calculation of center in pixels within the image and it's the orientation. A typical pixel-to-inch ratio for the MiNT testbed is 6.75. A variation in detection of image center by 30 pixels (which is very common in accurately setup testbed) would result in approximately 5 inches of error in the detection of each node in the testbed under accurate setup. Hence the distance between two pair of nodes is almost always misreported as 2 * 5 inches = 10 inches. As maximum distance in communicating nodes is only 4 feet, we have an error percentage of around 19% in standard conditions. This value will be even higher accounting for the changes in lighting, fading in color patch and even the shadow of wireless antennas falling at an awkward angle inside the node patch. The Figure 4 - Distorted Tracked Images shows the image blobs as classified by the pixel scanner under fine-tuned tracking system. The distortion in the image blobs and the false recognition can be easily seen. The partial detection of the green blob in the pictures can result in much skewed position and orientation values.

Figure 4 - Distorted Tracked Images in testbed

3.2.2 Mobility Deficiencies

Tracking the position of nodes is at the heart of mobility system. As the tracing system in the MiNT has a very large inaccuracy, the navigation system naturally suffers. This section briefly discusses the problems introduced by the incorrect tracking system in the node mobility.

The trajectory planning system, which uses the position as input, allocates sufficient thresholds in the tracking position input so as to avoid collision of nodes. This effectively means, when planning the trajectory, the node destination or source threshold could be well over 8 feet (the average tracking error). This contracts the space available for node movement, in an already miniaturized testbed.

The orientation of distorted images such as those in Figure 4 - Distorted Tracked Images in testbed may vary by around 10 degrees. As these images are taken from a near accurate testbed setup, the average error is approximately 10 degrees. The Roomba robot can only move in forward or backward direction or can make turns up to 360 degree from its current position. Hence, for moving a Roomba from source to destination, it has to be positioned towards the destination and then has to be moved forward. Hence an Incorrect computation of orientation will result in incorrect trajectory planning. The trajectory planning algorithm is self correcting and it takes intermediate tracking inputs and re-computes the path accordingly. But inaccuracies at each error point will result in a roundabout path from source to destination, wasting precious battery power in Roomba. This is explained in Figure 5 – Incorrect trajectory planning in MiNT. The figure shows in dotted lines the straight line path which could be followed and indicates the actual path followed by solid lines. The big diversions in the path at the camera intersections are due to partial tracking errors which are explained in subsequent paragraphs.

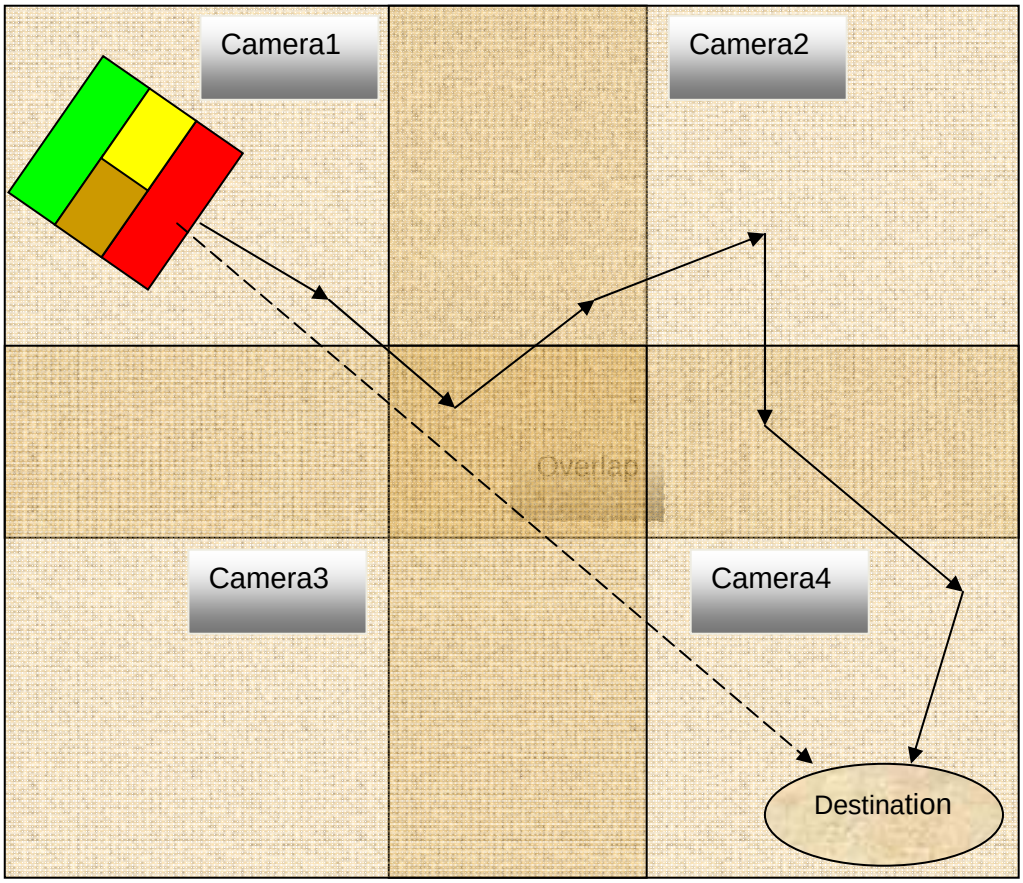

Figure 5 – Incorrect trajectory planning in MiNT

Moreover, Irrespective of how good the image is, if two nodes are close to each other, then the blobs accumulation might result in incorrect recognition of the nodes. For example if two nodes in the Figure 4 - Distorted Tracked Images in testbed are close to each other, then these two nodes could be detected as a single node with blue and pink identity. i.e by combining blue patch from the node in the left and pink patch from the node in the right. This could lead to complete miscalculation of the node position leading to incorrect trajectory prediction for nodes and collisions.

Worse yet, if there is a slight change in values of HSV of a single color used for representation, the nodes with these colors can be completely ignored and may not appear in the tracking input at all.

A single misbehaving camera can distort the values generated by all the near-by cameras, for nodes in the common region. The Overlap in between cameras is setup so that a node is completely visible in minimum one camera at all times. This means, the overlap region in between each and every camera is at least as big as the diagonal of testbed node, which is 80 pixels. In a typical MiNT setup at least 20% of the area is overlapped. The Figure 3 - Camera Overlap shows the overlap regions. If camera 1 is misbehaving (only for one of the patch colors), and camera 2 is accurate, then the values for position and orientation which should be taken as the average is skewed. Worse yet, if there is a node in a junction of four cameras, then one of the camera misbehaving will negate the effect of three other cameras.

The other problem with the camera overlap arises due to the partial node tracked by the camera. The node in Figure 6 - Partial Camera Image is tracked fully by camera2 and partially by camera 1, camera3 and camera4. Because camera 1, camera 3 and camera 4 do not have the full image of this node the center calculation of these cameras will be inaccurate. Averaging these values with the accurate value generated by camera 2 will ultimately result in an inaccurate center position. It has to be noted that this problem exists even when all the cameras have accurate tuning for the HSV color values. Although this problem can be fixed by setting up a minimum size for the blob, arriving at an optimum value is extremely difficult. Because of the distortion of the pixels as shown in Figure 4 - Distorted Tracked Images in testbed reducing blob sizes will result in genuine blobs being ignored, again resulting in a skewed node center position and orientation calculation. This also explains the distorted path in Figure 5 – Incorrect trajectory planning in MiNT. The incorrect positioning at a distinct point in overlap space, results in incorrect trajectory in the intersection of four cameras and this error is corrected only when the node is under a single

camera, in this case camera2. Again there is distortion at the overlap of camera2 and camera4, which is corrected when the node is completely inside camera4.

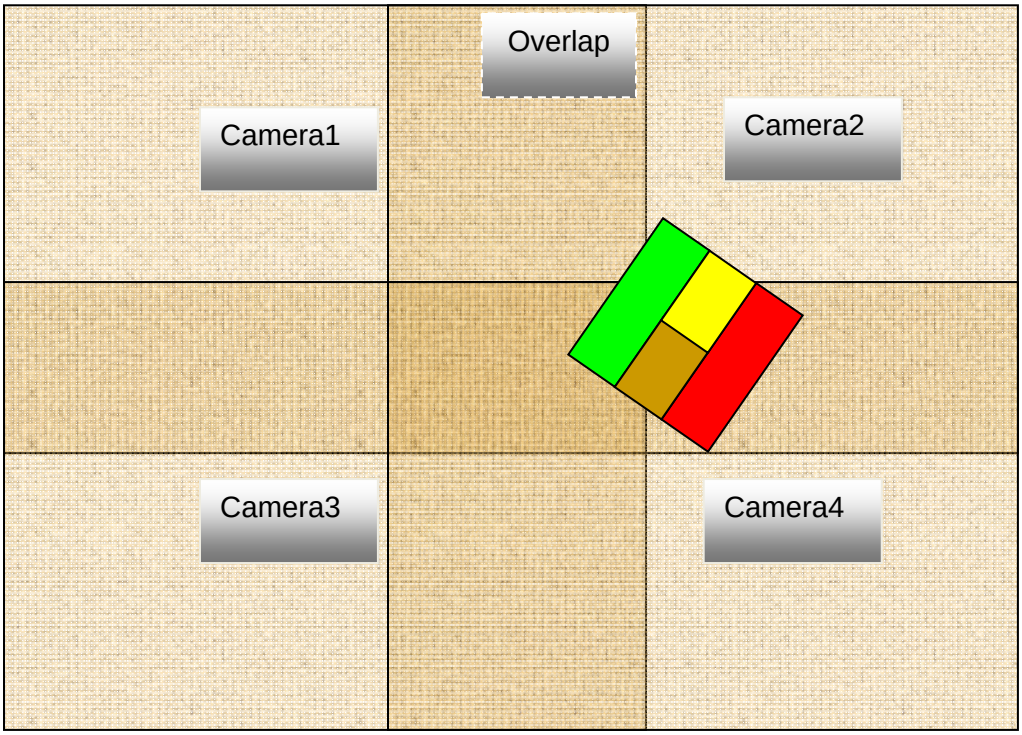

Figure 6 - Partial Camera Image

This problem of partial images in the cameras is amplified when the destination is at the overlap of the cameras. The trajectory of the node is completed distorted at the overlap that, the node circles around the destination point several times before reaching the final destination, again wasting battery resources. Worse, yet the convergence at some awkward positions take very long time to reach the destination.

Lastly, if at some point during the incorrect tracking and resulting incorrect trajectory planning, if the node happens to go out of the camera controlled regions and out of the testbed, manual intervention might be necessary to bring the node back into the testbed arena.

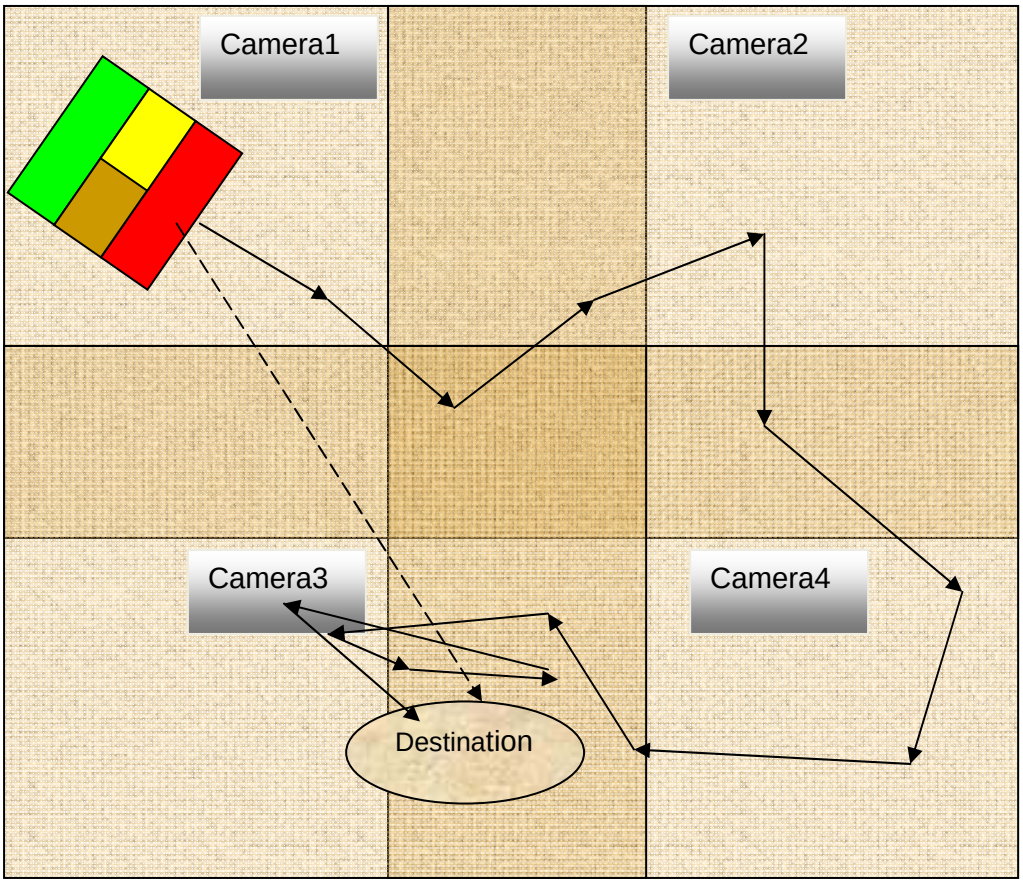

Figure 7 – Incorrect trajectory planning in MiNT at overlaps

Chapter 4

Related Work

In this chapter, we study the ongoing academic research for robot localization and mobility.

Localization and Mobility in contemporary Wireless Testbed

In this section we focus our attention on the contemporary wireless testbeds. We only consider test beds which support automated physical mobility.

One of the early testbed to propose physical mobility was the Caltech Multi Vehicle Wireless testbed [3]. They use custom-made Robots mounted on low friction omni-directional castors and two high powered ducted fans for providing mobility. The direction changes are accomplished using differential velocities of the ducted fans. Unlike other “kinematic” robots, which are able to stop movement instantly, these robots have second order dynamics. Hence these robots unlike others cannot halt instantly (due to inertia). The tracking system consists of four CCD cameras and one vision processing boards for processing images from the cameras at 60 Hz. The high frequency is required to account for the second order dynamics. The nodes are detected with black patches, on a white floor. This system is very expensive as compared to the current MiNT-m implementation also suffers from scalability issues.

Mobile Emulab [4] is a shared access testbed which supports automated node mobility by mounting the nodes on expensive and more sophisticated robots namely Acroname Garcia. This testbed uses overhead cameras for tracking the robots and uses a two colors to identify the nodes. They try to

avoid the problem of camera overlap regions by performing a image stitching and use the stitched image to identify the color patterns. To reduce the performance impact they rely on the odometry and sensor feedback until 1.5m and allow the errors to be accumulated until this distance. Errors are corrected in the next tracking cycle. The sophisticated robots keep the inaccuracy until 1.5m down to 2.2 cms. But the impact on low-cost Robots like Roomba would make this solution too costly. The tracking cycles for Rommba have to happen too frequently than the Acroname Garcia, because the sensory and odometry error accumulation for roomba is larger. Moreover if multiple nodes are moving, the need for stitching increases arising scalability issue.

An improvement to the MiNT-m testbed, namely MiNT-2 [5] is being developed in parallel at SUNY Binghamton. They still use the Roomba as the robotic platform but use a RFID based tracking system. The RFID tags are spread across the testbed at known positions within the testbed. Should the Roomba pass close to RFID tag (within 2.5 cms), the position of the tag is set to the node. In other circumstances, tracking relies on odometry and sensory data from the nodes themselves. As the self correction happens only at fixed intervals, the accuracy between RFID tags is high. Moreover, based on our experiences with Roomba and question over robustness of the Roomba based sensors tracking for larger distances, the lack of central control might lead to more collisions.

Localization in other systems

In this section we go beyond the wireless network testbed prototype and examine a few contemporary vision based node localization systems.

Huang Lee et al [6] present a novel localization technique for image based wireless sensor network. The wireless sensor nodes are fitted with pinhole camera and a beacon node vehicle is passed around the network. The beacon node stops at several locations and broadcasts it position. The authors consider three cases based on whether the beacon node is position aware or

not. If the beacon node is aware of its position, it will broadcast its position to nodes, which can use this data along with the camera image to identify their own position. They also present solutions when the beacon movement plane is parallel or perpendicular to the image plane. When the beacon node is not position aware, it is made to move at a constant velocity. The sensor nodes works in pair to identify the relative position and orientations, by making observations on the moving beacon and exchanging information. Once the pair-wise localization is complete, they can estimate beacon node's position and velocity by triangulation. Although simple and efficient, this method requires more and more moving objects to pin-point the position of nodes. Close to 150 moving objects can provide a positioning accuracy of around 0.2 meter. Moreover as the sensor nodes themselves are not mobile, this localization does not have to happen real-time.

Daniela Fuiorea et al[7] present a localization technique for a video based wireless sensor network. Because video based wireless sensors are deployed in great number, the pictures taken from them can have images that share common field of view and images that are taken from different position and angles .They localize the camera direction and spatial coordination and estimate the video-field overlap of these wireless sensor networks. They use SIFT (Scale Invariant Feature Transform) algorithm for automatic computation of image features and use this as an input for image registration technique, which is indeed used for localization. SIFT produces large number of feature points covering the entire image. These features are distinctive, easy to extract and allow for allow for correct object identification with low probability of mismatch. To optimize the performance, all the SIFT based processing are done in a central server and the results are sent back into the network.

Claudio Mello Jr. et al [8] present a novel idea for Robotic system for inspection of underground cable system. The robot which is fitted with a camera and multiple sensors can detect overheated points, partial electrical discharges and occurrence of cable treeing. Once the robot is inside the cable

system and detects the faults, a localization system for pinpointing the exact faulty location (position and orientation) is used, where the human operator can guide the robot to the faulty location. The robot uses a landmark based localization system, where the images along the path are scanned for natural or artificial landmarks and a virtual map of visited areas is created. These maps when combined with the odometry data from the robots can be used for localization. SIFT algorithm is used for landmark identification and map construction and feature matching is used to compare the current scene in camera with to the virtual map.

S. Panzeiri et al [9] present a low cost vision based localization technique for robots. They use simple landmarks like, ceiling light (which are placed with a regularity inside the buildings) to localize the node. They use off-the shelf webcams and mount them on robots. A map of the known landmarks generated and used as an environment representation for robot. The position and orientation of the landmarks are known. From the landmark image detected by the robot's camera the position and orientation of the robot can be identified. Unfortunately, this technique does not take collision into account and robot is assumed to be in a collision free environment.

Chapter 5

A New Robot Tracking and Navigation System Design

Most of the design decisions for the MiNT-m original Mobility system is based on the design goal for creating a cost-effective testbed. They considered alternatives for camera based tracking and systems like RF/ultrasound systems and other odometry mechanisms. Cost and in the later case inaccuracy, were the main factors for the use of an overhead camera based tracking system. They also explored the possibility for usage of various robot motion planning for complex scenarios, but opted for a simple and computationally efficient mobility options as the mobility patterns in the testbed are restrictive. We are driven by the same goal to minimize the cost and we continue to use off-the shelf low-cost ($90) webcams and use the same Roomba robotic platform which is primarily a vacuum cleaner with limited support for mobility and auto-recharge. Hence we propose a re-design of the mobility system which addresses all the problems in the MiNT mobility system.

5.1 Identifying Robots Using Patterns

From the experiences of the MiNT's color based tracking system, it is very clear that the color based system is the central source of most problems. Although color based system provides reasonable accuracy, the accuracy is not good enough for miniaturized environment like MiNT. Hence we propose a pattern based recognition algorithm which significantly improves the accuracy of tracking. The rest of the chapter is organized as follows. First, we propose a re-design of the tracking system, which is central to most of the issues in the MiNT. Then we propose improvements to the other components of the MiNT mobility system namely trajectory planning and collision avoidance.

5.1.1 Requirements

A tracking system which addresses all problems in MiNT should have the following characteristics

- Firstly, the system should be very accurate in determining the position as well as orientation of nodes in the testbed. As in MiNT, even a pixel inaccuracy of 30 pixels (> 5 inches) makes a big impact in the testbed. Ideally the new system should have an inaccuracy range of less than 2 inches and inaccuracy in orientation in less than 2 degrees.
- The tracking system is invariant to changes in lighting, scale changes and rotation changes in the testbed. The invariance to lighting greatly reduces the need for allowances in HSV in a color based tracking system, where as scale variation insensitivity will be required if the positioning of camera needs to change in the testbed. This aids in easy relocation of the testbed. The rotation invariance is required for accurately predicting the node in all different orientations.
- The tracking system should also be invariant to issues caused by camera overlap. Ideally the system should be able to track an image partially within the camera without biasing the output generated by an adjacent camera which reports accurate position and orientations.
- There has to be minimum calibration in the setup. For example, for the MiNT tracking to work, accurate values for HSV for each color used in the testbed have to be calibrated. As the lighting may slightly vary between the cameras, these calibrations have to be performed for individual cameras. Also, the ideal size of the blobs to be grouped has to be accurately determined. Ideally the new system should have minimum calibration requirements.
- More importantly, one of the original goals of MiNT is the minimization of cost for construction of the testbed. The image tracking algorithm should not increase the cost of the MiNT testbed significantly.

5.1.2 Scale Invarient Feature Transform (SIFT)

One such tracking algorithm, which satisfies most of the above properties listed is the SIFT (Scale Invariant Feature Transform) [2]. The algorithm was developed at the University of British Columbia. This algorithm as the name indicates is a feature point based identification technique. SIFT defines a sequence of complex transformations which scans an image to identify key “feature” points. The feature points produced by the algorithm are distinct enough to identify specific objects among a group of alternatives. These points are invariant to scale, rotation and partially invariant to illumination changes.

A bird-eye view description of the algorithm is presented in this paragraph for completeness. The algorithm uses a staged approach to identify feature points. Each stage in the algorithm is of increasing complexity and filters out candidate points so that more expensive transformations are applied only to filtered matches. The following are sequences of stages performed by the SIFT algorithm to detect the feature points

1. The first stage searches all scales and image locations to find a group of possible candidate points that are invariant to scale and orientation. The computations of these points are performed by using difference-of-Gaussian (DOG) function. The DOG points are compared against the nearby points to identify the maxima and minima.
2. At each point determined in the first step, a detailed model (3D curve fitting) is used to determine location and scale.
3. The low contrast points are then filtered out from the image. The Edge responses are eliminated using Hessian. The key points are selected based on their stability.
4. One or more orientations are assigned to these points based on gradient directions. All future operations are based on these

points which are transformed relative to assigned location, scale and orientation, making them invariant to these attributes.

5. The local image gradients are measured in region around all the key points at a selected scale. These are converted into a representation that allows for significant levels of local shape distortion and change in illumination.

This algorithm can be used to accurately identity an image in a complex real-world scene as shown in

Figure 8 - Feature detection in Sift. For a given training image, the algorithm computes the distinguishing “feature” points. These images points are then compared to feature points detected in the target image. The advantages and disadvantages of the SIFT algorithm are given below

Advantages

- Scale, rotation, illumination invariance.
- Identification of images which are cluttered in a real-world complex scene.
- Required only a minimum of 3 points to identify a matching object.
- Close to 90% repeatability in the identification of feature points.

Disadvantages

- Performance. A single comparison for a 640X480 image with a training image takes on an average 400ms on a machine with Intel Pentium core 2 Duo 2.53 GHZ with a 6MB L2 cache, 3 GB RAM.
- Problems in position and orientation detection of an image with symmetry.
- Requires images with sufficiently large features for accurate matching.

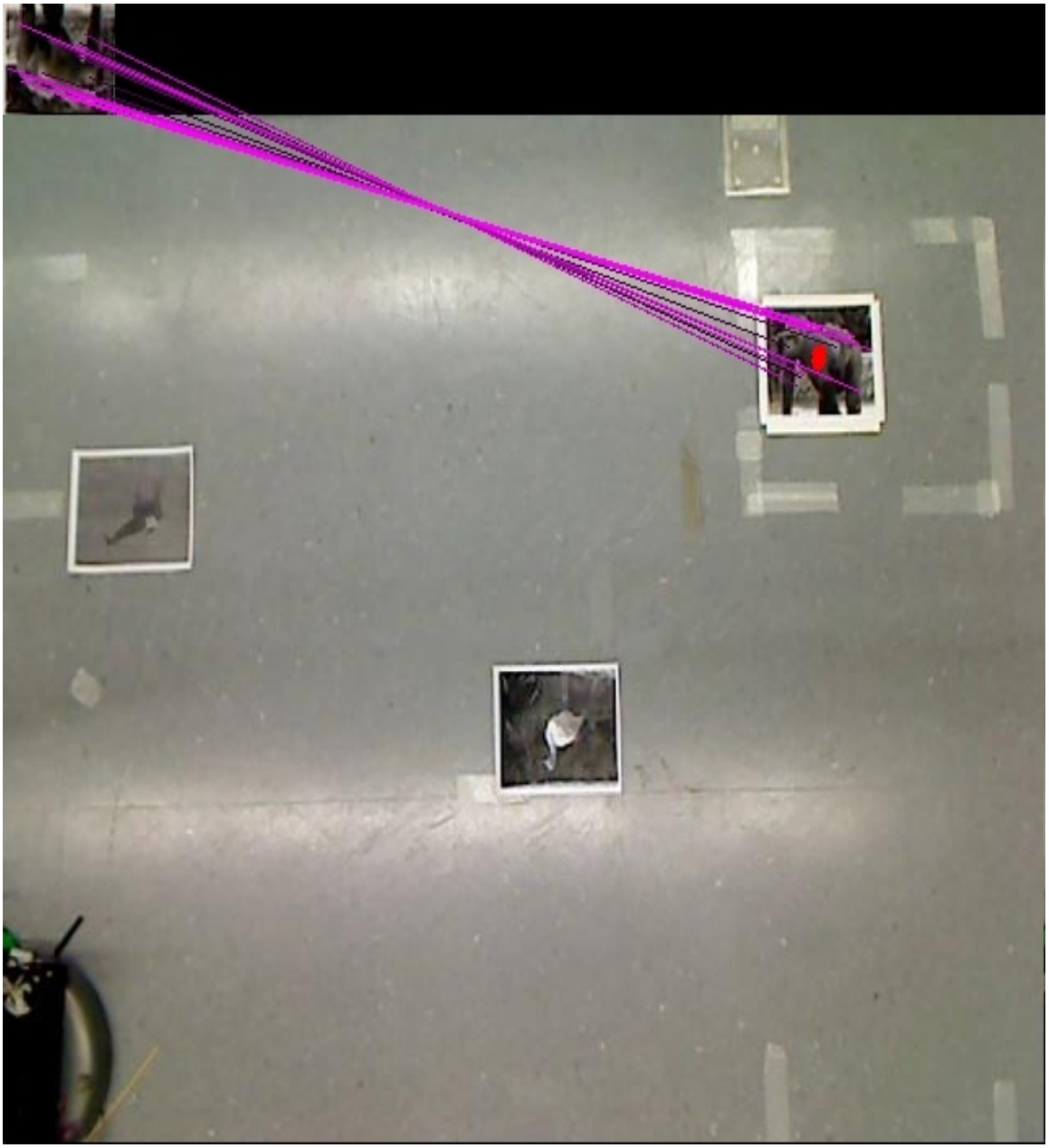

Figure 8 - Feature detection in Sift

This figure illustrates the scale and rotation invariance of SIFT in locating images. The pictures shows matching feature points in pink lines. The picture also clearly indicates that the SIFT does have a small percentage of false positives with some of the surrounding images.

5.1.3 SIFT based Robot Tracking

The SIFT algorithm can be employed in the tracking for MiNT. Images with large number of feature points can be used as training sample as shown in the Figure 9 - Testbed Node with SIFT Training Image. This figure shows a testbed node with sift tracking image on its top.

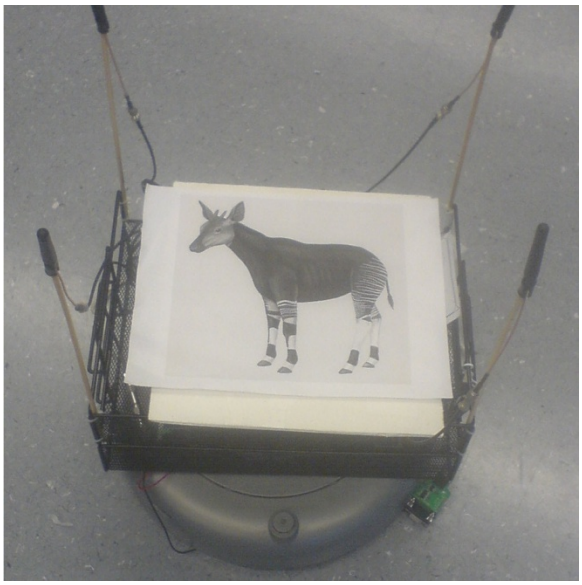

Figure 9 - Testbed Node with SIFT Training Image

All testbed nodes are placed below the cameras and a training image for the SIFT algorithm is generated during the initial testbed setup. These images are used as the input for the SIFT based tracking algorithm. The images from the overhead cameras during the experiment run (i.e. the target images) are captured and are compared against these training images to identify the matching nodes.

SIFT only works at the feature point level and does not work at an image level. It only identifies pairs of matching feature points in the camera target image and the training image and returns position of the point in the target image and the orientation (rotation) of this matching feature point with respect to the training image. This is shown in

Figure 8 - Feature detection in Sift with pink lines. Here the image at the top is the training image. The pink lines connect the feature point in the training image to the corresponding match in the target image. For each point, the (x, y) co-ordinates of this matching feature in the target image is returned by SIFT, in addition to the rotation (orientation) of this point. For a typical training image there are 20-30 such distinct feature points. But as individual points tracked in SIFT may be slightly inaccurate, we use a simple tracking algorithm on top of the feature points returned by SIFT to accurately predict the center and orientation of the node. For computing the center of the node in the target camera image, we compute the center point with respect to individual matching feature points and average these points to find the actual center. The orientation of the node is simply calculated as the average of individual orientations. This eliminates the slight inaccuracies in tracking of individual feature points.

The computation of the centre point with respect to the individual features is explained below and is based on the following assumptions. As the same camera resolution is used to capture the training image and actual camera tracking image, there is a fixed relationship between the points in the node's camera target image and the training image. The only difference being that the camera target could be in a different orientation compared to the training image. Moreover, the center point of the training image is known in advance. For each of the matching training image feature point, we can calculate its distance from the center and the slope of the line connecting these points within the training image plane. We leverage this fact to find the center of the rotated image with respect to individual feature points. Given these, the problem of finding the center is reduced to the problem of finding the rotation of the center point in the target image (which can be solved by shifting center point). The same process is repeated for each pair of matching feature points and the center and the orientation of the node in the target image is computed.

The accuracy of the tracking increases as the training image has more and more feature points. To avoid inaccuracies in tracking we only pick training images with sufficient number of feature points. Experiments have showed that 20 – 30 feature points are good enough to achieve accurate tracking in the order of less than less than one inch or approximately 2.5 cms.

Performance Optimizations

For the trajectory planning and the collision avoidance to be accurate, real-time feedback of testbed is as important as the accuracy of the tracking. A typical roomba movement in the testbed is at an average rate of 1/4 feet per second. Hence a delay of more than a second in the feedback means, an inaccuracy of $1/4^{th}$ of a feet and it defeats the whole purpose of the re-design.

Although SIFT algorithm is very accurate, it is compute intensive as compared to the original color based tracking scheme. In order to reduce the cost of testbed, keeping in mind the original goal of cost-effectiveness of the MiNT testbed, we propose the following optimizations to the reduce need for high-performance machines for running SIFT algorithm.

5.1.4 Predictive Tracking

The first publication of SIFT algorithm, which was in 1999 [10] claims that the tracking time for the SIFT based tracking is less than 2 seconds in a machine with average compute power then. This value will be significantly less, taking into the account, the progress in the CPU speed and memory capabilities of machines. For a typical testbed target image (640 X 480 pixels), the time required to perform a SIFT matching is about 0.4 seconds on an average. In a MiNT testbed with 10 cameras, SIFT algorithm as is not sufficient enough to provide real-time tracking in MiNT for 12 nodes, as each camera image has to be scanned for 12 times, once per each node.

As tracking is resource intensive, the first basic optimization is to move from camera based to tracking to node based tracking. i.e. instead of the

tracking each camera target image for all the testbed nodes, we only look for nodes in the specific areas of the testbed. This predictive node based tracking works as follows. During the initial startup, there is a full scan of all the camera target images and the positions of all the nodes in the testbed are determined. Once this initial position is known, we can switch to node based tracking. As the position of each node is known and we also know the direction of roomba movement if any, we can accurately predict the position of the node in the testbed in the next scan cycle. Instead of scanning the position of the node in all cameras, we can cut a small piece of the target camera image where the node is predicted to be in, and run SIFT in this portion of the image only. The node based tracking reduces the computational complexity by a order of number of cameras which is 1/10th in this case. Also the typical size of the cut image is 1/10th of the original image.

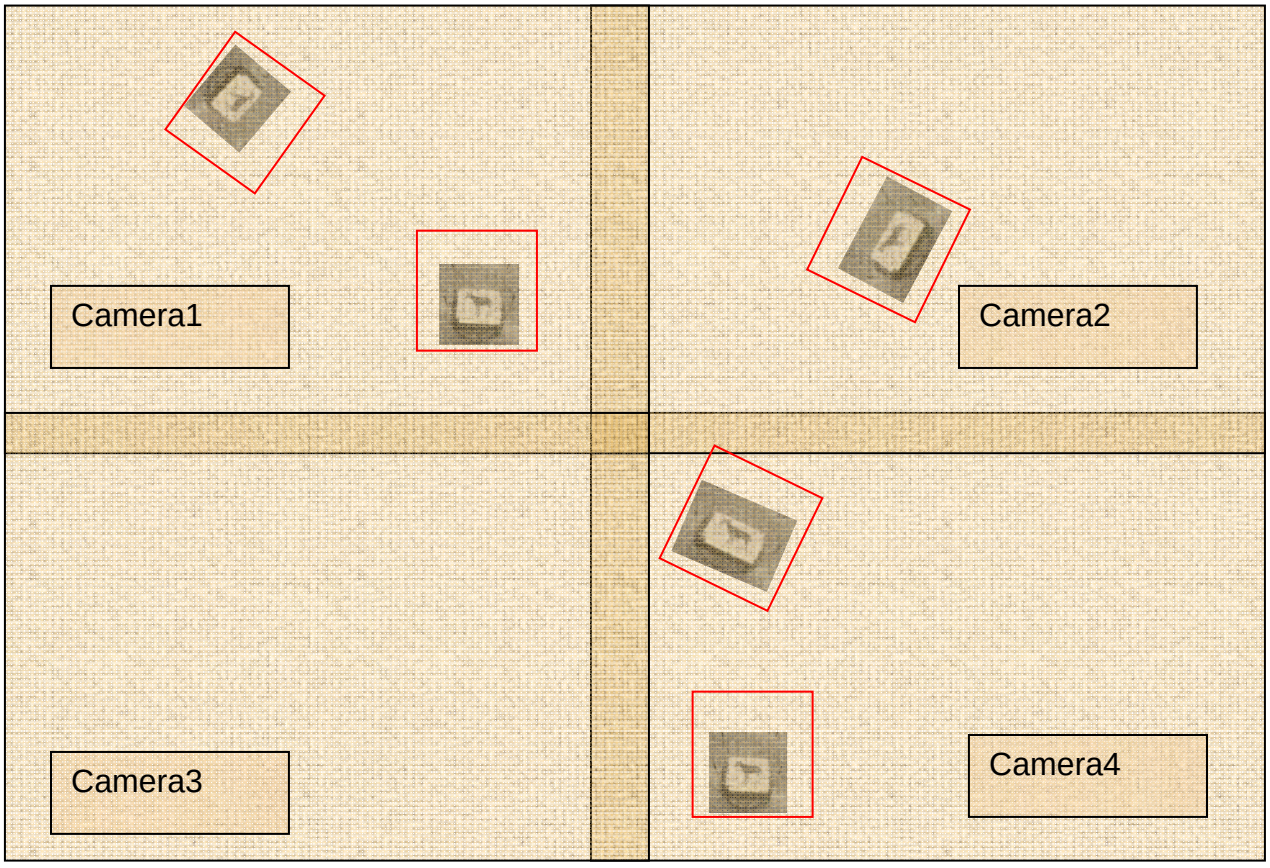

Figure 10- Predictive Tracking in cameras

The predictive scan logic is explained in explained in

Figure 10- Predictive Tracking in cameras . The Camera 1 image is cut into two smaller images (the cut images indicated by the red rectangles) and the only those images are used as input to the SIFT algorithm. Similarly only portions of camera2 and camers4 are scanned. The Camers3 input image is not required as there are no nodes under this camera. The Figure 11- Predictive Tracking Scan Range in the testbed provides a snapshot of a cut- image used in the predictive scan.

Figure 11- Predictive Tracking Scan Range in the testbed

Not all nodes in the testbed will be moving at all times. So we can differentiate these nodes by having different scan frequencies for moving and non-moving nodes. i.e real time feed is not required for non-moving nodes as their position is known priori. Only moving nodes have to be tracked in frequent scan cycles. Assuming an average upper limit of 50% of the nodes in movement in the testbed, we still achieve a 2 times speedup.

Moreover, if nodes in the testbed are distributed under a few cameras, the rest of the camera images need not be processed at all saving the precious compute resources and as well as USB bandwidth of camera traffic.

5.1.5 Improving Accuracy

Figure 8 - Feature detection in Sift shows there could be a few false matching points outside the actual target image. Including these points in the average computation can bias the results resulting in inaccuracy. There are algorithms like RANSAC[3] (Random Sample Consensus) can be applied to improve the outliers in SIFT, we chose to use a simple algorithm for these eliminations because the external disturbances in the testbed arena are minimal and the false positive can easily be identified by a simple algorithm.

As the actual size of the training image is known, the maximum distance a single feature of training image could be away from the center is known. But when SIFT identifies a match, it is hard to predict which of them are inside and which lie outside. Hence for each point, we compute the distance from the other matching points, and try to see if distances are within the threshold as compared to the maximum possible distances in the training images. If the difference is higher, then the point is considered to be an out of an image and is not included in the average.

5.2 Scalable Camera Array-based Robot Tracking

Typical testbed setup consists of 10 cameras. Predictive tracking ensures that the number of cameras connected to a camera server is not limited by the ability to perform SIFT computations. But the bottleneck would be the USB bandwidth and the ability to acquire images from multiple cameras simultaneously. Hence a multi-tiered scalable architecture for tracking is required for proper load balancing.

Moreover, as the SIFT works on feature level, it does not require the entire image to be available to perform tracking, and it can work with partial images. This allows us flexibility as to have little or no overlap in between the cameras. Such kind of architecture however requires a central authority who

can control and integrate the partial feature points from all the cameras and perform the position and orientation detection.

Common solution for both these problems is to have client-server architecture. To this effect, we separate the camera servers, the nodes which are actually connected to the cameras (Camstream in the original architecture) and introduce a new component called the tracking server. The camera servers performs the SIFT calculation based on the image input from the tracking server and sends them all to the tracking server. Tracking server then combines the feature inputs from all the cameras and performs the calculations.

This also means the predictive tracking has to work across the cameras. There can be a maximum of four cameras which can capture a single node image. The captured image may or may not have overlaps. An example of the how the partial image can be setup is illustrated in Figure 12- Predictive Tracking Scan Range across cameras

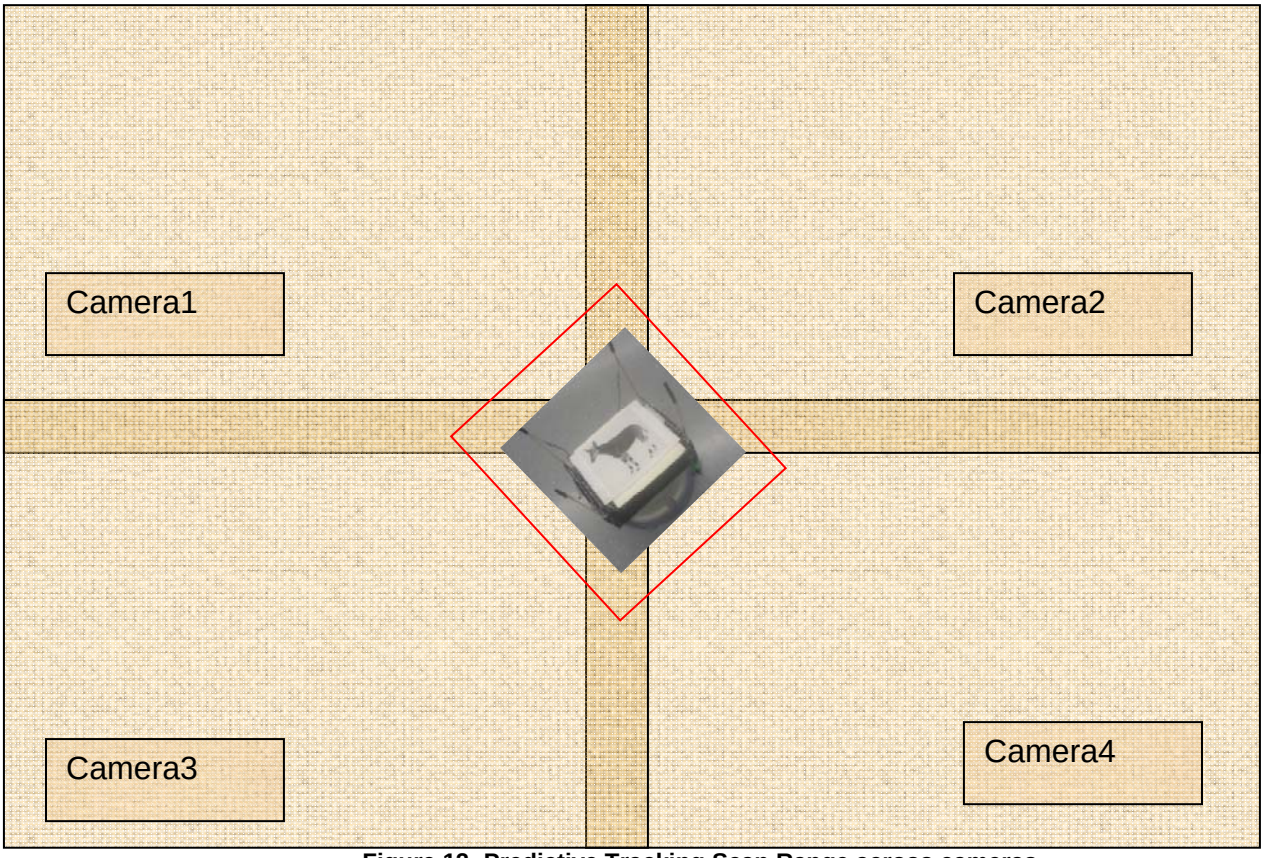

Figure 12- Predictive Tracking Scan Range across cameras

The tracking server needs to find all the cameras where this node could be part off, and sent the respective small regions to be cut-off and tracked to individual cameras. In this case it has to send region within red rectangle to each cameras and perform an integration of the feature as shown in Figure 13- Partial Tracking and Center detection

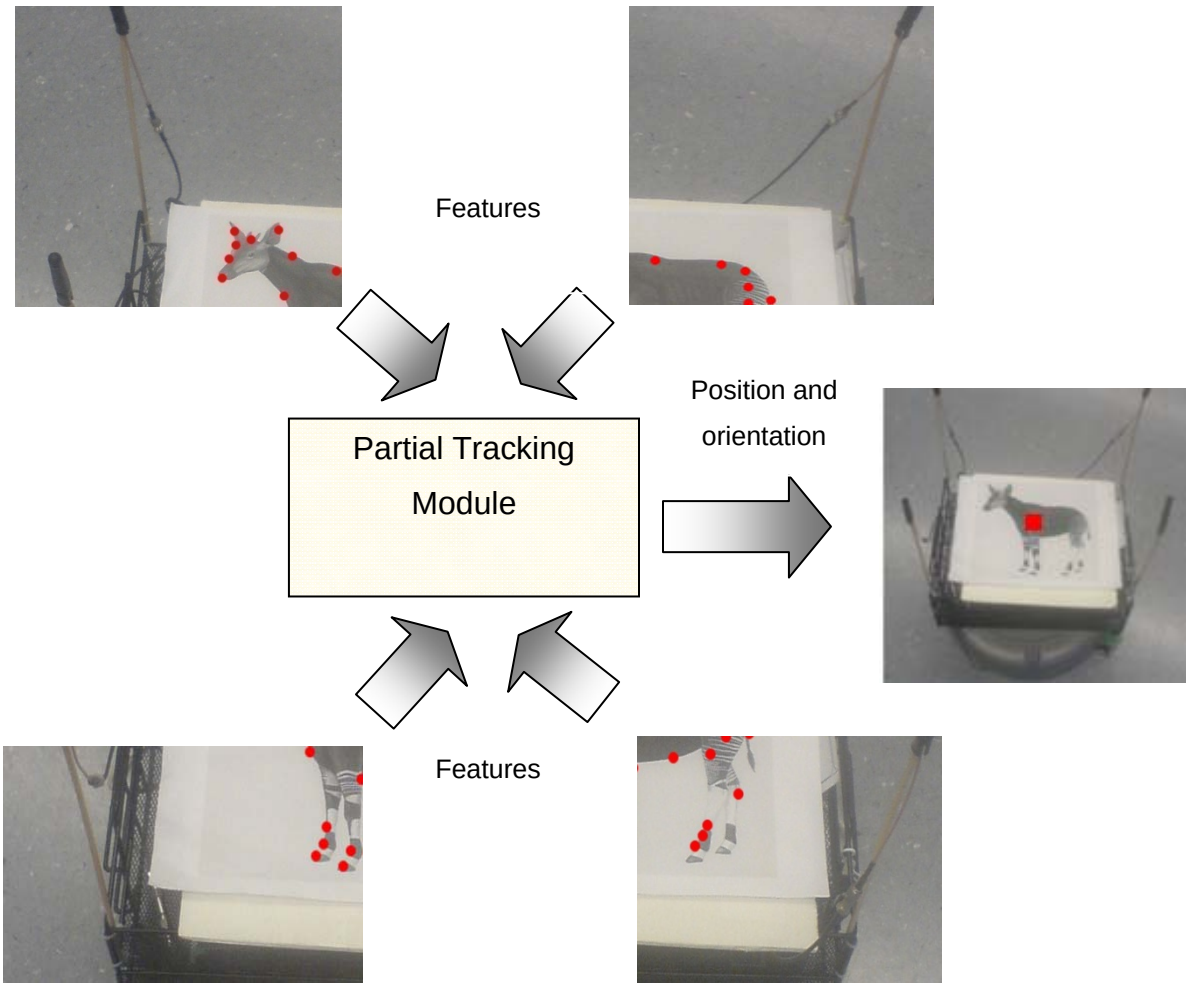

Figure 13- Partial Tracking and Center detection

The tracking module then integrates the feature points from the individual camera servers and computes the actual position and orientation. We eliminate the overlapping points if any, and calculate the center position in (x,y) inches and orientation in degrees. As the testbed setup will be done to ensure that there are no gaps in between cameras, such a model will be able to predict the entire node accurately.

Camera Orientation

One other problem, which has to be addressed in the partial tracking, is with respect to the orientation of the cameras. The Figure 12- Predictive Tracking Scan Range across cameras showed the cameras to be aligned in perfect boundary to illustrate the idea of partial tracking.

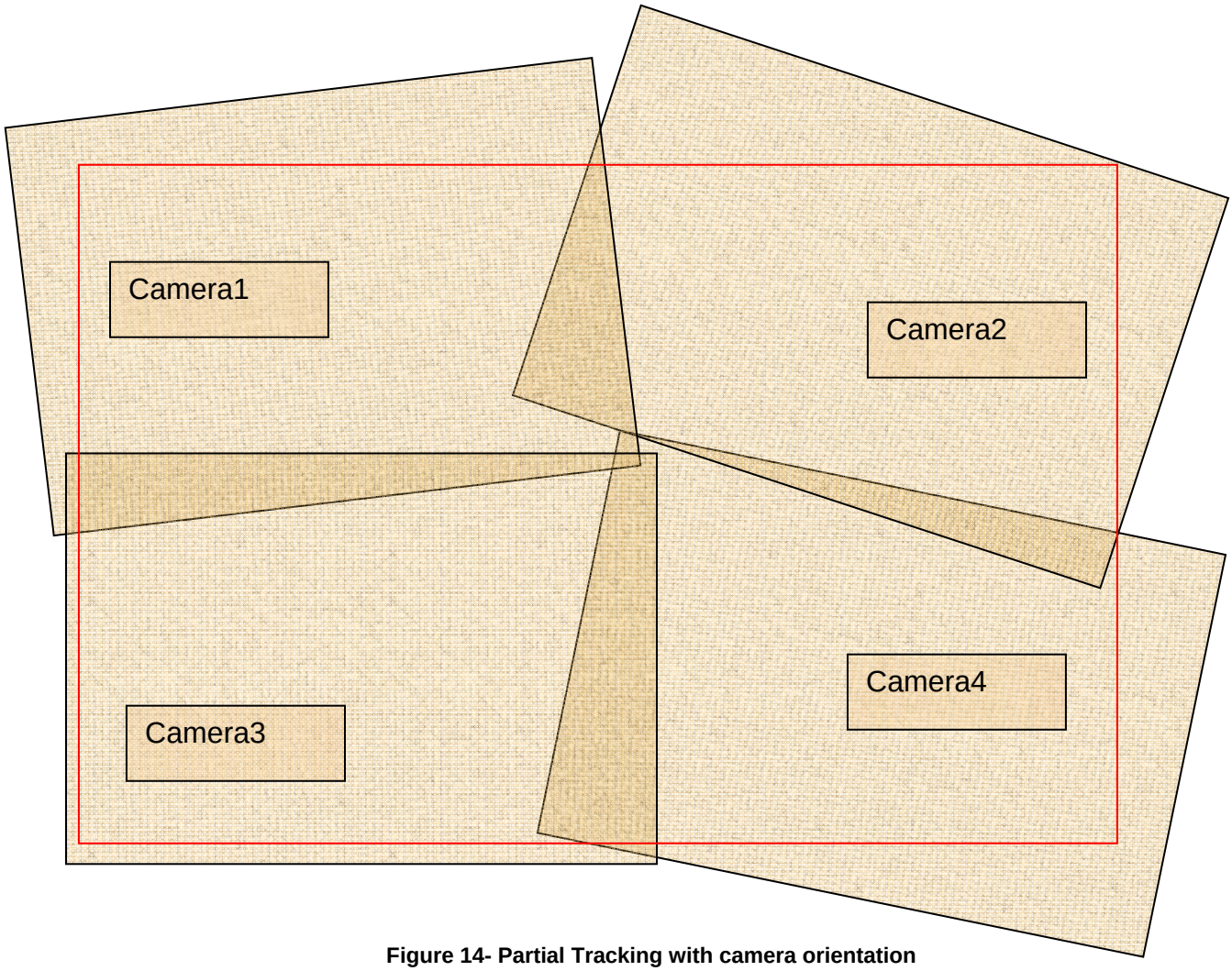

Figure 14- Partial Tracking with camera orientation

But during the setup, the cameras could be aligned in any fashion subject to the restriction that there are no gaps in testbed as shown in Figure 14- Partial Tracking with camera orientation. This arrangement is supported to reduce the stringent constraints on the experiment setup for perfect alignment of cameras.

The red border in the Figure 14- Partial Tracking with camera orientation is actual testbed area. A tool was developed for aiding the setup, which will ensure that the cameras are setup without any gaps in the testbed. The tool will output the final position and orientation of the all the camera in the testbed.

This setup introduces the distinction of position and orientation of nodes within the camera (the local position) and position and orientation of nodes with respect to the testbed (the global position). The tracking server needs to convert the feature points from the local to global position. It also has to calculate the bounded rectangle in terms of the local position of the cameras and use the same as the inputs for the global position calculation.

5.3 Trajectory Planning and Collision Avoidance

The trajectory planning in MiNT-m although heuristic based does not have any major flaws. But due to constraints in the tracking accuracy, the original MiNT-m only had very limited trajectory planning and collision avoidance. The MiNT-m planned trajectories with current testbed data. The path to destination was computed with current placement of obstacles. If there is an obstacle currently in the path, a detour was planned and the trajectory was computed accordingly. At each new tracking image input from the camera, all the nodes which are moving are checked to see if there is any obstacle or possibility of a collision in the very next step. If there are no obstacles, then the node is moved forward. If there is an obstacle or possibility of a collision, a randomly picked node is paused in its place, allowing the other node to move around it.

There is also a delay between the moment commands for movement is given to the node's robot (roomba) and the time when it executes it. But Mint-m relied purely on the position tracking through the cameras and did not receive feedback from the nodes on completion of the command. This introduces the possibility of sending more movement commands, for the same step when the original command is pending execution, causing the node to overshoot its destination and even worse move out of the boundary area of the testbed. To counter this problem, MiNT-m had an upper bound on the time between issuing two movement commands. This introduced additional delays in the each step, causing the node to pause for long time between each scan cycle.

As the node positioning is accurate with SIFT, we can greatly improve upon the trajectory planning. Instead of merely acting upon the current testbed snapshot, we can plan the trajectory accurately ahead of time. As the position and the direction of movement of each node are known priori, we can predict ahead the position of all nodes after a few cycles and plan the path for these nodes accordingly. i.e. we can create a snapshot of where the nodes will be after a few scan cycles and plan the trajectory and collision avoidance accordingly.

Moreover, as Roomba is primarily a vacuum cleaner, with some support for serial API communication, the interface for movement is rather preliminary. It only allows control of the velocity of movement and the radius of rotation. Hence a forward movement is achieved by specifying a longer radius and a specific velocity and, the turning is controlled by specifying appropriate velocity and radii. But due to practical limitations on the speeds that can be specified, a fine grained control of the actual direction in degrees and movement in centimeters is hard to achieve in Roomba. Nonetheless, Roomba provides an API to read the distance and angle of roomba movement. This feedback can be used to correct the errors in specifying the actual degree and distance for movement of roomba. It is important to note that the accuracy of this odometry accumulates

as the distance increases and the data is found to be accurate for short distances.

Taking all this factors into account, we device an algorithm, for trajectory planning as follows. We plan the entire path of the roomba ahead of time for multiple steps. If there are obstacles in the middle, we plan a detour around the obstacle. If two paths collide, we still pause one of the nodes, but predetermine the point of collision, and make one of the nodes stop at the predetermined point, allowing the other node to pass through. We then resume the original node's movement and allow it to continue through to the destination or next collision path. This would make the movement of the Roomba faster thereby reducing the time required for Roomba to move from source to destination.

For this algorithm to work, we need to create a queue of commands in the Roomba controller module (noded). The tracking server can pre-determine a series of steps and queue it at the noded for all the moving nodes. The controller in the mean-time can ensure that the node movements are collision free and take required corrections like halting a node and clearing the noded queue etc. This monitoring is still required to counter the wheel slips if any in the roomba, non-responsive roomba or unexpected node crashes. But the speed of monitoring can be reduced significantly.

Periodic feedback from the Roomba movement can be used to improve the accuracy of movement errors. If Roomba were originally required to move 'n' degrees and due to tuning errors if the node has only moved a little less than 'n' degrees, then the compensation in the next move can happen in the noded. Once a command with sequence of moves is issued to the Roomba, the controller can monitor the node and only communicate to it if there is an unexpected collision due to reasons stated above. In the normal case, the noded can complete the command and send confirmation to the controller on the completion of the command.

In order to control the accuracy of the movement during a turn, we fine tune the velocity of movement at the last few steps, in order to not over shoot the destination. This improves the accuracy of the directional movement and ensures that the node moves according to the pre-planned trajectory. We also make fine grained adjustments to the straight line movement, by proportionally reducing the velocity when the node moves closer to the destination to avoid overshooting the destination and thereby altering the pre-planned path calculated by trajectory planning algorithm.

5.4 Integration with NS-2 to support Mobility Simulation

MiNT-m supported mobility primarily to configure network topology. The user can use the signal and interference characteristics, and configure appropriate values between nodes but it did not support movement during experiments.

The re-design of the mobility system, allows us to support movement during the experiments. Our current implementation relies on specifying the movement in terms of time when the movement should occur, and destination position for nodes in a separate script isolated from the NS-2 script. The integration of the movement into the NS-2 script is a straight forward implementation.

When the movement control commands are executed by the noded, the noded sends a command to the tracking server module. The tracking server module can include that node into the list of moving nodes and include it in the trajectory planning and collision avoidance computations.

Chapter 6

Software Implementation

This section briefly summarizes the Software components of the new mobility system as explained throughout Chapter 5. The tracking server component in the original architecture Figure 1 - Mint Architecture is split up into camera server and a tracking server. Other components in the architecture have significant role changes in the context of mobility. The non-mobility components of the MiNT-m architecture remain the same. The new software architecture is illustrated in Figure 15- Mint Mobility Architecture Redesign.

6.1 Testbed Nodes

The noded daemon which is the central control for movement in noded requires the following changes.

The noded movement control module in the original MiNT was a command interface which merely translates the commands from the controller in for movement specified in (inches and degree) to appropriate velocity and radii and issues the command through serial interface to the roomba.

The noded movement module is refined to have some intelligence with respect to movement commands. The noded can now queue multiple commands and initiate bunch of these commands sequentially. The noded also corrects the inaccuracies in movement between sequential commands, by adding corrections in the subsequent steps. The velocity of movement in the last step is fine tuned to prevent the overshooting of destination.

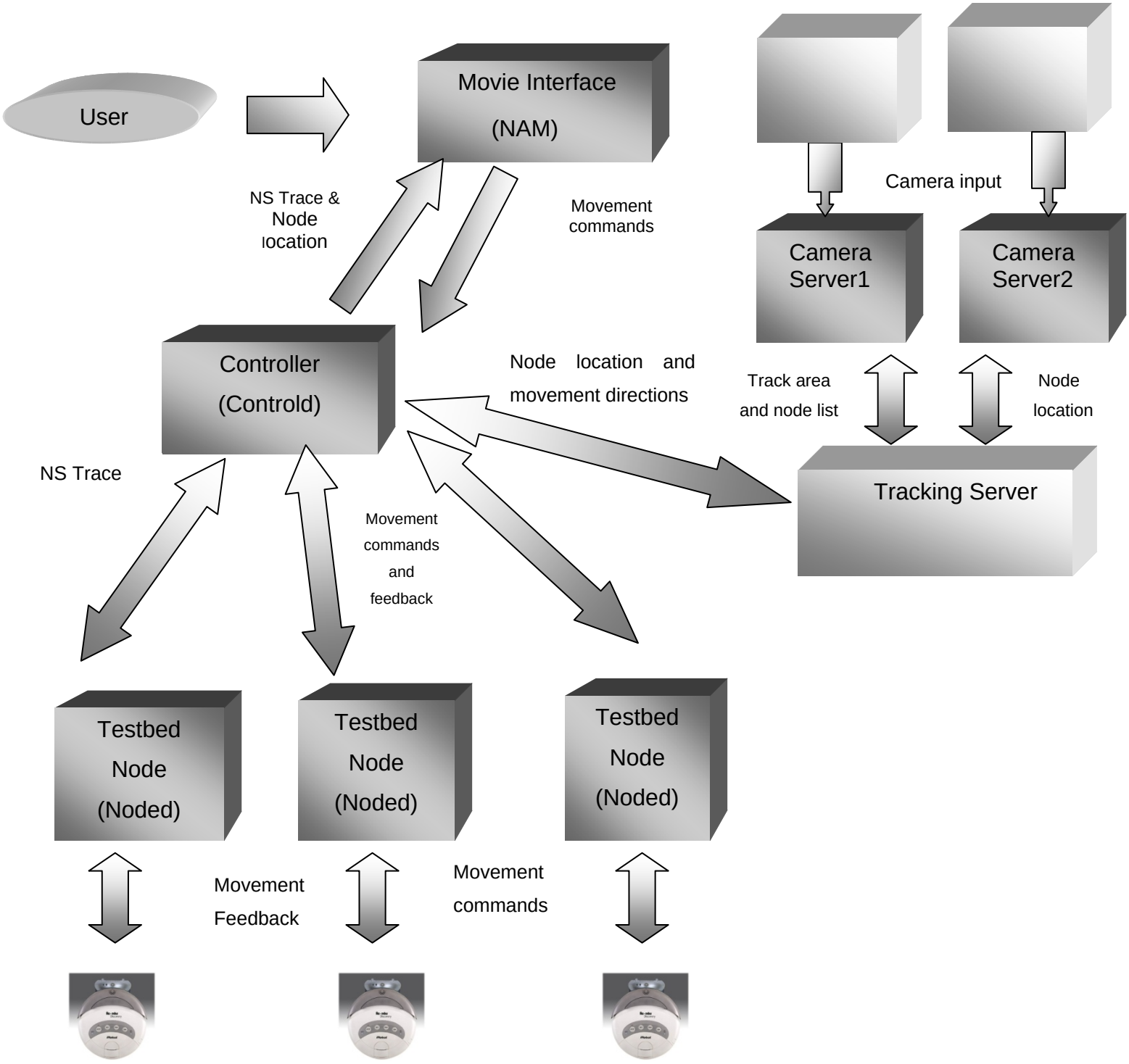

Figure 15- Mint Mobility Architecture Redesign

Noded is also responsible for the movements during experiments. The noded parses the movement scripts and initiates the movement request with the controller module at the time specified in the script and waits for the controller to respond with the movement commands. The received commands are executed in the same manner as explained above.

New Roomba API

The Roomba SCI (serial command interface) [10] is a serial protocol by which commands can be passed through the external serial port of Roomba robot. The SCI supports various commands to control the robot and/or query the internal state of its sensors. The Roomba supports specification of movement by specifying the velocity of the in millimeters per second and radii, The longer radii allow it to move straighter and the shorter radii makes it turn more. The Roomba also provides an API to query the sensory data, to which Roomba responds with distance travelled (from the last such query) in millimeters and the angle of rotation. As Roomba rotation is achieved by applying variable velocity for wheels, the angle returned is the difference in distance travelled between the right and left wheel. This can be easily converted to the angle by using the formula (as specified in the API), where 258 is the distance in millimeters between the Roomba wheels.

Angle in Radians = 2 * angle returned / 258

The serial interface program for communication with the Roomba was completely re-written to correct previous implementation flaws which crashed the roomba and modification to the interface program to query the distance and angle travelled by the roomba in the last step.

6.2 Controller

In the context of MiNT mobility system, Controller is responsible for planning the trajectory and collision avoidance. Controller can receive

movement commands from the MOVIE interface through user query or from the noded command during the scripts.

When a movement command is received in the controller, its plans the trajectory of movement by predicting the snapshot of the testbed in advance (accounting for other moving nodes and static nodes) and sends a sequence of commands, which either takes the roomba to its final destination or an intermediate waypoint (determined by collision). It then translates the trajectory as sequence of rotations and movement steps and communicates the same to the noded daemon. It closely monitors the testbed for adverse conditions like collision and initiates necessary corrective actions.

Controller interacts with the Tracking server for tracking updates and creates a global picture for display in the MOVIE interface. Controller also uses this tracking data for testbed monitoring and movement control.

6.3 Tracking Server

The tracking server is the completely re-designed in the new system. In the older MiNT-m architecture, the tracking server was the machine connected to the cameras. But the newer architecture treats it as an interface to multiple camera servers.

One issue that has to be addressed for the tracking server is the synchronization between the individual cameras. An entire testbed image can be constructed from only after combining the output from all the cameras. Hence the tracking server sends sequences of tracking commands to all the cameras and waits for the responses. Once the tracking data is received from all the cameras the tracking server can combine individual features and send the same controller.

The tracking server is responsible for predictive tracking. Before sending a tracking request, the tracking server interfaces with the controller to get the information regarding the current moving nodes. For each of these nodes,

based on the previous location and the moving direction the tracking server computes the global position in the testbed where this node is expected to be present. It then identifies which camera(s) cover this region and translates this global position into the local position within each of those cameras. It directs individual cameras to track those portions.

Once the replies are received from all the cameras individual node feature points tracked from the cameras are converted back from camera based local positions into global testbed positions. These global positions for features are then used to calculate the node's position and orientation within the testbed.

This component is separated from the controller to achieve better scalability.

6.4 Camera Server

The camera server is the machine connected to the cameras. As the images from the camera have to be queried synchronously with the tracking request from the tracing server, we have changed the open source software streamer implementation to query the data from the camera synchronously with the tracking request.

For each tracking request the camera image is queried from the camera. For each node within the request, the camera server cuts-off the required pieces of camera image and uses the SIFT algorithm to obtain the matching feature points. The individual feature points for each node are sent back as response to the tracking server.

Chapter 7

Evaluation

This section evaluates the performance of the new MiNT Mobility system. Particularly we focus on the movement trajectory, tracking accuracy and the time required for tracking. We use a testbed setup with 4 cameras and 6 nodes, where the nodes are evenly distributed across the cameras and a testbed dimension of 10 X 12 feet. We also tune the velocity that roomba moves approximately at a rate ½ feet per second. The camera resolution is 640 X 480 and the pixel-to-inch ratio is 8.89.

Tracking Accuracy

We start with the tracking accuracy of the entire tracking system. We place nodes under the multiple cameras and run the tracking algorithm with the predictive and partial tracking in real-time. We plot the results for different images used in the testbed tracking. The Table 1 - Tracking Accuracy for different illustrates the average errors for several runs with nodes placed under different portions of the testbed. For estimating the worst case performance of the partial tracking, we used image tools like GIMP to cut-off images at pixel level (purposefully such that a few feature points are taken out) and used these images instead of feed from cameras. Although in real-time calibration, this scenario is not possible as user cannot fine tune the overlap of the cameras to be in the order of pixels, this gives us a maximum bound on the worst-case inaccuracy due to partial tracking.

Image	Number of Cameras	Average Number of feature point in each camera	Error in Position (inches)	Error in Orientation (degrees)
Letter Tag A	1	4	5.12	40
Letter Tag B	1	3	7.29	60

Image of a okapi	1	33	1.12	0.9
Image of a okapi	4	12	1.08	0.897
Image of a okapi	4 pieces of manipulated images	8	1.43	1.1
Image of a Hippo	1	43	1.23	1.2
Image of a Hippo	4	15	1.19	1.023
Image of a Hippo	4 pieces of manipulated images	9	1.68	1.321
Image of a elephant	1	21	1.35	1.28
Image of a elephant	4	8	1.32	1.27
Image of a elephant	4 pieces of manipulated images	3	1.94	1.73
Image of a Chimpanzee	1	55	1.03	0.87
Image of a Chimpanzee	4	20	0.99	0.85
Image of a Chimpanzee	4 pieces of manipulated images	12	1.24	1.02
Average Inaccuracy(exclusive of A and B)			**1.30**	**1.12**

Table 1 - Tracking Accuracy for different images

The accuracy for okapi, Hippo, Elephant and chimpanzee which all have no symmetry in the image and have feature points in the range 21 – 55 have very accurate results. But the usage of letters like A and B which have symmetry in itself leads to larger error in orientations. Also the number of false matches for the letter strokes is high. This statistics clearly indicates that the accurate choice of image for the node is mandatory for the high accuracy of the tracking.

Trajectory Accuracy

We evaluate the accuracy of the trajectory (without obstacle) movement by measuring the distance covered by roomba (as reported by its sensors) measured periodically without accumulating errors. Although this is slightly inaccurate, it is hard to measure the same manually.

For this test, we make the node to repeatedly move across the testbed in diamond shaped paths. The measurements are shown in the table below.

Number of edges covered in the diamond	Straight Line Path distance (in inches)	Additional distance covered by roomba
1	93.722	10.1292
2	187.444	22.2373
3	281.166	31.4742
4	374.888	42.1243

Table 2 – Error in distance moved

The linear increase Table 2 – Error in distance moved in the .error with the proportional increase in distance, instead of a cumulative increase is because of the periodic correction of inaccuracy by tracking inputs.

An illustrative path of the observed tracking path patterns during this experiment are given in Figure 16 – Illustrative Node Path. The red-line shows the straight line paths for the movement. The actual Roomba movement oscillates close to the straight path, primarily due to slight inaccuracies in tracking and the roomba movement.

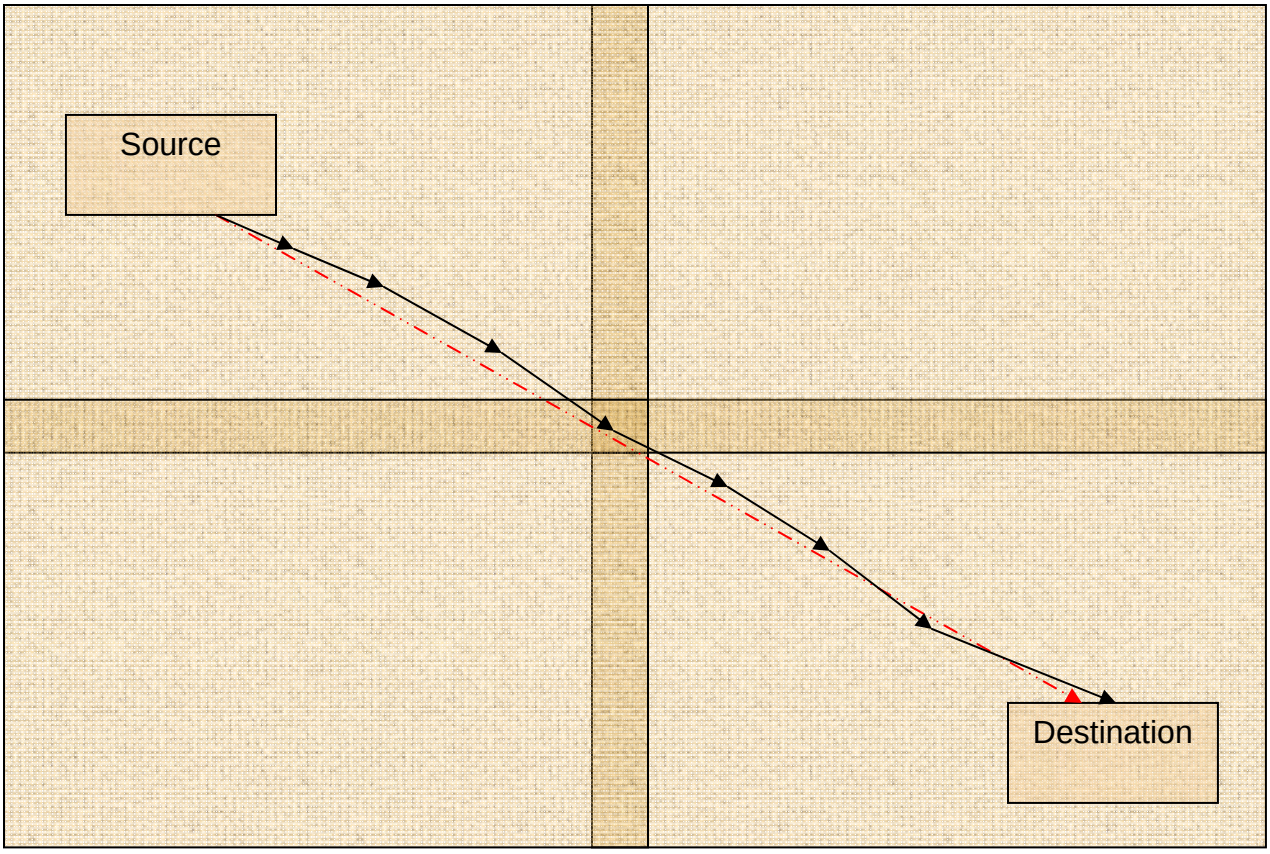

Figure 16 – Illustrative Node Path

Tracking Delay

The time delay for tracking with and without the predictive tracking is given below.

Number of Cameras	Number of Moving nodes	Time in ms for non-predictive tracking	Time in ms for predictive tracking
4	1	1408	46
4	2	1408	97
4	3	1408	150
4	4	1408	194

Table 3 – Time taken for Predictive Tracking

The time for non-predictive tracking is a measure of number of cameras where as that of predictive tracking is a measure of number of moving nodes.

Movement Accuracy

We also measure the movement inaccuracy of the individual node movement as follows. The tracking accuracy provides the minimum value for inaccuracy in the whole system, because this inaccuracy cannot be avoided from the trajectory planning. To compute the maximum bound of inaccuracy, we also need to compute the movement inaccuracy for individual roomba. As the movement is executed as sequence of steps in roomba and periodic feedback from the tracking system corrects these values, at any given point the maximum possible inaccuracy in movement is bounded by the sum of tracking inaccuracy and the roomba movement inaccuracy.

Roomba movement time in (ms)	Velocity in mm per second	Distance to be moved (inches)	Distance actually moved (inches)	Error in movement (inches)
2000	100	8	7.5	0.5
2400	100	9.6	9	0.6
4100	100	16.4	15	1.4
5200	100	20.8	17.5	3.3
2000	200	16	12	4
2400	200	19.2	16	3.2
4100	200	32.8	25	7.8
5200	200	41.6	29	12.6

Table 4 – Straight line Movement accuracy for Roomba

The table clearly shows that the inaccuracy for velocity 100 accumulates as the distance increases. The interesting statistics however is the result for velocity 200. As the velocity of the movement increases, roomba's wheels frequently slip during the initial acceleration accounting for the much reduced

forward movement. Hence the best accuracy (minimum inaccuracy) is achieved by moving roomba in velocity 100 in short steps for less than 2 seconds. Adding it together with the average tracking accuracy of 1.3 in Table 1 - Tracking Accuracy for different images we get a maximum inaccuracy of 1.8 inches at any instance in time in the testbed.

Maximum Movement Speed

In this section, we measure the maximum possible movement speed that can be achieved in the system. Two factors influence the speed of movement of the node. First, is the maximum speed of the Roomba which is 500 mm/s. Second factor, is the speed at which tracking input can be supplied such that the tracking system can monitor accurately, every movement step and avoid collision. If the tracking speed is too slow, the roomba could move at the faster speed than the tracking system predicts it to be, resulting in collisions.

Table 5 – Movement speed vs navigation accuracy

Number of moving nodes	Speed of tracking (number of tracking inputs per sec)	Collision threshold in (inches)	Maximum Speed of movement supportable without compromising accuracy in mm per sec
1	21	6	500
2	10	6	500
3	6	6	500
4	5	6	500
1	21	3	500
2	10	3	500
3	6	3	500
4	5	3	500

The collision threshold defines the minimum possible distance that can be allowed between the roomba before activating the collision avoidance measure (pausing one roomba and allowing others to move). For safe movement, there has to be at least 2 tracking cycles before roomba covers this collision distance.

But as we do collision avoidance instead of collision prevention, as long as there are two cycles before error accumulations reach the collision threshold, there will not be any collisions. As we saw in Table 4 – Straight line Movement accuracy for Roomba, the error increases as the velocity increases due to wheel slips. This problem to certain extent can be countered by moving the first step in 100 and other steps with increased velocity. We assume a theoretical constant error of 0.5 inches for roomba movement as long roomba is moved continuously for less than 1/2 second. The table clearly shows that roomba's At this error, the roomba's maximum movement speed is the bottleneck.

Scalability

The controller and tracking server are the only non-scalable components in the current architecture. More number of cameras and nodes can be added in the system by adding more camserver machines.

In general, as the peak tracking speed for SIFT is 21 calculations per second, and the camera used in the testbed setup currently supports 30 fps frame rate, the camera used is never a limiting factor.

If more and more robots have to be added to the system, the testbed area has to be proportionally increased to support free movement of these nodes. As the SIFT tracking is dependent on the number of nodes to be tracked, as long as nodes are distributed equally within the testbed, the USB bandwidth limits the performance. But as the density of nodes increases within the coverage region of a single camera server, the SIFT tracking becomes a bottleneck.

Chapter 8

Conclusion and Future Work

This chapter summarizes the contributions of thesis. We conclude by presenting future directions.

Summary of Thesis

In this thesis, we proposed a re-design of MiNT-m mobility system using the SIFT based feature identification, which improves the accuracy of the original color based tracking system. With the introduction of complex algorithms like SIFT, always introduces performance concerns. We improved the performance of tracking using SIFT by making various optimizations like predictive tracking so that the image tracking system could be used to feed real-time data.

We also improved the trajectory planning and collision detection by predicting the position of nodes ahead of time instead of reactive measures in the original MiNT-m design. The feedback for node movement from the roomba was used for improving the accuracy of commands issued to the roomba movement. A combination of accurate tracking and velocity control were used to accurately place the nodes in their destination.

Future Work

- Integrating the Node movement control into the NS-2 script. The node movement during experiment is currently specified in a separate script. This can be integrated into the NS-2 script.
- Auto-recharging of the Roomba batteries as well as the node's routerboard batteries are supported in MiNT-m, but this can be

vastly improved by extending the mobility re-design to accommodate the auto-recharge.

- The movement support during experiments could be enhanced by a fine grained control of velocity of movement.
- More fault tolerance and robustness in the noded, which can detect misbehaving Roomba and take corrective actions.

Bibliography

[1] Pradipta De, Rupa Krishnan, Ashish Raniwala, Krishna Tatavarthi, Nadeem Ahmed Syed, Jatan Modi, and Tzi-cker Chiueh, “MiNT-m: An Autonomous Mobile Wireless Experimentation Platform”, In Proceedings of Mobisys, 2006

[2] D. Lowe, “Distinctive image features from scale-invariant keypoints”, International Journal of Computer Vision, 60, 2 (2004), pp. 91-110

[3] Lars Cremean , William B. Dunbar, Dave van Gogh, Jason Hickey, Eric Klavins, Jason Meltzer and Richard M. Murray, "The Caltech Multi-Vehicle Wireless Test bed", Proceedings of the 41st IEEE Conference on Decision and Control, 2002, 86-88 vol.1, Dec. 2002

[4] D. Johnson, T. Stack, R. Fish, D.M.Flickinger, L. Stoller,R. Ricci, J. Lepreau, "Mobile Emulab: A Robotic Wireless and Sensor Network Testbed", INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, 1-12, April 2006..

[5] C. Mitchell, V.P. Munishwar, S. Singh, Xiaoshuang Wang, K. Gopalan, N.B. Abu-Ghazaleh, "Testbed design and localization in MiNT-2: A miniaturized robotic platform for wireless protocol development and emulation", Communication Systems and Networks and Workshops, 2009. COMSNETS 2009, 1-10, Jan. 2009.

[6] Huang Lee, Laura Savidge, Hamid Aghajan, “Subspace Techniques for Vision-Based Node Localization in Wireless Sensor Networks”, In Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Toulouse, France, May 2006

[7] Daniela Fuiorea, Vasile Gui, Dan Pescaru, Petronela Paraschiv, Istin Codruta, Daniel Curiac and Constantin Volosencu, “Video-based Wireless Sensor Networks Localization Technique Based on Image Registration and

SIFT Algorithm”, WSEAS TRANSACTIONS on COMPUTERS, Issue 7, Volume 7, July 2008

[8] Claudio Mello Jr., Eder Mateus Gonc¸alves, Emanuel Estrada, Gabriel Oliveira, Humberto Souto Jr. Renan Almeida, Silvia Botelho, Thiago Santos, Vin´ıcius Oliveira, “TATUBOT – Robotic System for Inspection of Undergrounded Cable System”, IEEE Latin American Robotic Symposium, 2008

[9] S. Panzieri, F. Pascucci, R. Setola, G. Ulivi, “A Low Cost Vision Based Localization System for Mobile Robots,” 9th Mediterranean Conf. on Control and Automation – MEDSYMP 2001, June 27-29, Dubrovnik, Croatia, 2001

[10] David G. Lowe, "Object recognition from local scale-invariant features," International Conference on Computer Vision, Corfu, Greece (September 1999), pp. 1150-1157