

Distributed System for Domestic Robot Operation Using Computer Vision

Marcel-Titus Marginean

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Science in Information Technology

Department of Computer and Information Sciences

The Jess & Mildred Fisher College of Science & Mathematics

DISSERTATION APPROVAL PAGE

This is to certify that the dissertation prepared by Marcel Titus Marginean, entitled "Distributed System for Domestic Robot Operation Using Computer Vision", has been approved by this committee as satisfactorily completing the dissertation requirements for the degree of **Doctor of Science in Information Technology**.



Dr. Chao Lu

Chair, Dissertation Committee

4/26/2016

Date

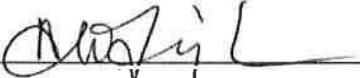


Dr. Marius Zimand

Member, Dissertation Committee

4/26/2016

Date

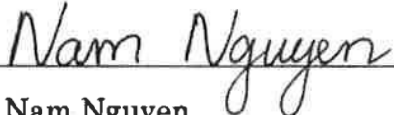


Dr. Alex Wijesinha

Member, Dissertation Committee

4/26/16

Date



Dr. Nam Nguyen

Member, Dissertation Committee

4/26/16

Date



Dr. Chao Lu

Chair, Department of COSC

4/26/16

Date



Dr. Ramesh Karne

Doctoral Program Director

4-26-2016

Date



Dr. Janet DeLany

Dean of Graduate Studies

4-27-2016

Date

Acknowledgements

I want to express my gratitude first and foremost to Dr. Chao Lu for his precious guidance and ideas offered during the doctoral program.

I want to also thank Dr. Marius Zimand for his guidance in navigation of the processes and procedures.

A special thank you goes to my wife Angela and my son Mihai who tolerated me through a lot of mood changes while working in the doctoral program and picking up the slack for household tasks. Your help made this thesis possible.

A big thank you goes to my mother, poet Maria Marginean from whom I inherited the dedication and grit to cope with difficult moments while keeping sustained efforts on polishing my work towards perfection.

And in the end, I am dedicating this thesis to the memory of my father Titu Ioan Marginean who instilled in me from an early age the ardent passion for science, technology and engineering.

ABSTRACT

Anticipated for a long time in science-fiction literature, domestic robotics are timidly starting to appear. While the first applications, like vacuum-cleaners or toys, do not share much of the versatility of the robotic servants envisioned in literature and movies, it is just a matter of time before more useful robots appear, mainly driven by the demand for help for the aging population in the industrialized world.

This dissertation research aims to study, propose, and start to develop an integrated home automation (domotic) system in the form of an Intelligent House infrastructure. We envision the model of the fully integrated Smart House of the future as being a self-sufficient intelligent system able to take care of the inhabitants, with the robots becoming just the autonomous mobile components of the assisted living environment. So far Computer Vision (CV) seems to be the most promising technology to allow robot navigation in domestic environments, therefore a large part of the works is focusing on the aspects of using CV for domestic robot operations and addressing challenges that come with it.

The work began by proposing a distributed processing architecture for controlling a robot operating in a domestic environment and navigating it using computer vision. The system is composed of a set of fixed cameras mounted on the walls near the ceiling overlooking the various rooms, a set of networked computers located in the house, home automation devices communicating with domotic computers and one or more mobile units (robots) having on board their own camera and processing equipment. We are taking advantage of the already existing Wi-Fi and wired networks in any modern house

to provide the communication between equipment and this allows us to keep the cost of the system within an affordable range.

As a basic implementation of the proposed architecture we experimented with a set of software components called Camera Module processing data from fixed cameras, a Situation Awareness Module integrating data from all the rest of the modules and building a map of the environment and the robot control software. The robot control software is in itself distributed between the computer board located on the robot and a part running on the base station. The components of the robot control software are in constant communication between themselves over Wi-Fi.

The first task we handled on the Computer Vision side of work was to implement an object tracking algorithm by fusing together multiple well known CV operations into a multi-paradigm tracker. The MP-Tracker algorithm developed runs inside a Camera Module receiving images from a fixed camera, detecting moving objects, and sending information about them to other components in the system.

Situation Awareness Module (SAM) uses homographic projection to create a map of the environment. Once the model is built, the same equations are used to translate the coordinates of the moving objects received from the tracker into absolute coordinates in the room. The Robot Module is part of the robot controlling software that runs on the Base Station and uses the data model built by SAM to perform path planning and sends navigation commands to the Autonomous Robot Module.

Autonomous Robot Module (ARM) is the part of robot control software that runs on the robot itself. Besides translating the high level navigation commands from Robot

Module in hardware control signals, it also processes video from the on-board camera. Images from the mobile camera are used to both calculate optical-flow for low level navigation and for maintaining a trajectory, as well as providing upon request to other modules for epipolar geometry calculations.

To tie all the modules together, we developed a new communication protocol sDOMO designed from the beginning as a protocol for domestic home automation and robotic systems. A major area of concern for using robots and generally any automation device in domestic environments is the security and privacy of the inhabitants. To address this concern we designed sDOMO to implement a self-sufficient home-centered automation network where the devices are able to perform their duties over the house network and any access to the outside world would be strictly controlled. The protocol has multiple layers of security and privacy protection and is being offered as an open source project for general purpose home automation and building of robotic systems.

A new framework for processing multiple messages in parallel is being proposed as a new design pattern. The Multi-Threaded Message Dispatcher framework is a generalization of the mono-threaded Reactor design pattern to work on a heavy multi-threaded environment while encapsulating all the logic required to provide guaranteed deadlock avoidance. The framework will take care of all low levels details about locking and unlocking the access to critical resources allowing the programmer to focus on the problem he needs to solve instead of being distracted with critical section management.

To be able to test the system we built from scratch a small domestic robot powered by a Raspberry PI 2 embedded computer board and an Arduino-Nano micro-controller to access the hardware in real-time. The robot is based on a differential drive

platform with two DC motors commanded by Arduino via H-Bridges circuits. The robot Camera is mounted on a Pan-Tilt mechanism powered by two servo-motors. The embedded computer board is running the ARM software communicating with the rest of the system via sDOMO on Wi-Fi network. Current dissertation provides most of the information required for our robot to be replicated by other researchers.

Table of Contents

| | |
|------------------------------------------------------|-----|
| List of Figures | xi |
| List of Tables | xiv |
| 1 INTRODUCTION | 1 |
| 1.1 Why | 1 |
| 1.2 Contributions | 4 |
| 1.3 Literature Review | 8 |
| 2 CONCEPTS ABOUT DISTRIBUTED SYSTEMS..... | 15 |
| 2.1 Distributed Systems Architectures | 15 |
| 2.1.1 Client Server Architecture | 17 |
| 2.1.2 Service Oriented Architecture (SOA)..... | 18 |
| 2.1.3 Peer-To-Peer | 19 |
| 2.1.4 Clusters and Grids | 20 |
| 2.2 Data Exchange Models..... | 21 |
| 2.2.1 Message Passing | 21 |
| 2.2.2 Remote Procedure Call | 22 |
| 2.2.3 Distributed Shared Memory..... | 23 |
| 2.3 Distributed Computing Considerations | 24 |
| 2.3.1 To Distribute or not..... | 24 |
| 2.3.2 Data-link Considerations | 26 |
| 3 CONCEPTS ABOUT COMPUTER VISION..... | 30 |
| 3.1 MOG Background Subtraction and Segmentation..... | 30 |
| 3.2 Shi-Tomasi Corner Detection..... | 31 |
| 3.3 Lucas Kanade Sparse Optical Flow | 32 |
| 3.4 Homography..... | 33 |
| 3.5 EPIPOLAR Matching | 34 |
| 4 CONCEPTS ABOUT ROBOTICS | 36 |
| 4.1 Differential Drive Kinematic | 36 |
| 4.2 Controlling DC Motors | 41 |
| 4.3 DC-DC Buck Converter | 46 |
| 4.4 Controlling Servo-Motors | 47 |

| | | |
|-------|-----------------------------------------------------------|----|
| 4.5 | PID Controller | 49 |
| 5 | SYSTEM ARCHITECTURE | 51 |
| 5.1 | Overview | 51 |
| 5.2 | Software Architectures | 53 |
| 5.3 | Major Messages into the System..... | 56 |
| 5.4 | Environment Model..... | 57 |
| 6 | sDOMO PROTOCOL | 59 |
| 6.1 | Overview | 59 |
| 6.2 | Data Communications | 63 |
| 6.2.1 | Overview | 63 |
| 6.2.2 | Discovery and Configuration..... | 65 |
| 6.2.3 | Packets and Messages | 67 |
| 6.3 | Security and Privacy Considerations..... | 68 |
| 6.3.1 | Packet Level Security Considerations | 69 |
| 6.3.2 | Message Level Security Considerations | 72 |
| 6.4 | Additional Services | 74 |
| 6.4.1 | XML Files | 74 |
| 6.4.2 | House Intelligence Unit and Internet Gateway | 76 |
| 6.4.3 | Data Integrators and Adapters | 78 |
| 7 | MULTI-THREADED MESSAGE DISPATCHER | 81 |
| 7.1 | Problem | 81 |
| 7.2 | Proposed Solution | 83 |
| 7.3 | Mission Critical Application Support | 84 |
| 7.3.1 | Dealing with Race Conditions and Deadlock Prevention..... | 84 |
| 7.3.2 | Support for Separation of Concerns..... | 85 |
| 7.3.3 | Support for Unit Testing | 87 |
| 7.4 | Design Details | 87 |
| 7.5 | Typical Usage..... | 92 |
| 8 | MULTI-PARADIGM OBJECT TRACKER | 94 |
| 8.1 | MP-Tracker Introduction..... | 94 |
| 8.2 | Method | 95 |
| 8.3 | Target Modeling..... | 99 |

| | | |
|-------|----------------------------------------------------------------------------|-----|
| 8.4 | Target Life Management | 102 |
| 8.5 | Fuzzy Histogram | 103 |
| 8.6 | Lucas-Kanade Tracking | 105 |
| 8.7 | Hypothesis Management | 107 |
| 8.8 | Second Chance Algorithm | 109 |
| 9 | VISION BASED NAVIGATION | 112 |
| 9.1 | Multiple Landmark Localization of a Stationary Robot | 112 |
| 9.2 | Localization of a Mobile Robot Using a Single Landmarks (Running Fix) | 121 |
| 9.3 | Visual Odometry by Optical Flow | 124 |
| 9.3.1 | Method | 125 |
| 9.3.2 | Optical Flow Vector Classifier Algorithm..... | 128 |
| 9.3.3 | Homography in Speed Measurements | 132 |
| 10 | THE ROBOT – ROBI-1 | 135 |
| 10.1 | ROBI-1 Hardware | 137 |
| 10.2 | ROBI-1 Software Infrastructure | 143 |
| 10.3 | ROBI-1 – Autonomous Robot Module | 145 |
| 11 | EXPERIMENTS AND RESULTS..... | 147 |
| 11.1 | System Implementation..... | 147 |
| 11.2 | sDOMO Protocol..... | 150 |
| 11.3 | Message Dispatcher..... | 152 |
| 11.4 | MP-Tracker | 156 |
| 11.5 | Localization Experiment Using Pre-existing Landmarks | 159 |
| 11.6 | Localization Experiment with Active Artificial Landmarks..... | 162 |
| 11.7 | Static Landmark Localization Experiment..... | 166 |
| 11.8 | Running Fix Localization Experiment | 176 |
| 11.9 | Localization Experiment with Fixed Camera..... | 182 |
| 11.10 | Optical Flow Odometry Experiment | 186 |
| 11.11 | Navigation Experiment with Fixed Camera Alone | 192 |
| 11.12 | Conclusion and Future Work..... | 194 |
| | References..... | 197 |
| | Curriculum Vitae | 206 |

List of Figures

| | |
|------------------------------------------------------------------------------|----|
| Figure 1 Client-Server Architecture..... | 17 |
| Figure 2 Peer-To-Peer Network | 19 |
| Figure 3 Cluster Computing System..... | 20 |
| Figure 4 Network Segment with Switches..... | 28 |
| Figure 5 Euclidean Relationship between Two Views, Epilolar Geometry..... | 34 |
| Figure 6 Classical Differential Drive | 38 |
| Figure 7 Differential Drive Kinematic..... | 39 |
| Figure 8 Simple DC Motor | 41 |
| Figure 9 Speed-Torque Characteristic Curve for DC Motor | 42 |
| Figure 10 Controlling a DC Motor with a Bi-Polar Transistor..... | 43 |
| Figure 11 PWM Modulation | 44 |
| Figure 12 : Typical implementation of an H-Bridge..... | 45 |
| Figure 13 Typical Zener Circuit..... | 46 |
| Figure 14 Typical Buck Converter..... | 46 |
| Figure 15 A Servo-Motor Disassembled..... | 48 |
| Figure 16 Servo Angle Control via Fill Factor | 48 |
| Figure 17 Feedback Control System..... | 49 |
| Figure 18 Overview of the System | 51 |
| Figure 19 Main Software Modules | 54 |
| Figure 20 sDOMO - Hybrid Communication Architecture | 61 |
| Figure 21 Device State Diagram while Connecting to the Hub..... | 66 |
| Figure 22 Intelligence Tag for Washing Machine Example..... | 77 |
| Figure 23 Multiple Message Sources , Shared Data ,One Message Processor..... | 82 |
| Figure 24 Dispatcher Main Components | 84 |
| Figure 25 Class Diagram related to Safe Data Access..... | 88 |
| Figure 26 Class Diagram for Message Management..... | 89 |
| Figure 27 Class Diagram for Message Dispatching | 91 |

| | |
|--------------------------------------------------------------------------------|-----|
| Figure 28 MPTracker Algorithm..... | 97 |
| Figure 29 Fuzzy Histogram Membership Function | 104 |
| Figure 30 LK Cropped Window and Mask..... | 106 |
| Figure 31 LK Tracking Procedure | 107 |
| Figure 32 Geometric Locus of a Robot , Two Landmarks | 112 |
| Figure 33 Complete Locus as a Reunion of Two Circles | 115 |
| Figure 34 Disambiguation Using External Blob Tracking Information | 116 |
| Figure 35 Exact Localization with Three Landmarks..... | 118 |
| Figure 36 Moving Robot and One Landmark in Robot Coordinates..... | 121 |
| Figure 37 Moving Robot and One Landmark in Room Coordinates..... | 123 |
| Figure 38 Calculating Visual Odometry | 126 |
| Figure 39 Optical Flow from a Forward Moving Camera Looking at the Floor | 129 |
| Figure 40 Pseudocode for Vector Classifier Algorithm | 130 |
| Figure 41 Optical Flow from a Rotating Camera | 131 |
| Figure 42 Robot Camera Calibration..... | 133 |
| Figure 43 ROBI-1 | 135 |
| Figure 44 Raspberry PI2 - the Main Computer Board of the Robot..... | 136 |
| Figure 45 MAX-97 Base..... | 137 |
| Figure 46 Pan-Tilt Mechanism for Camera | 137 |
| Figure 47 ROBI Hardware Block Schema..... | 138 |
| Figure 48 Arduino Nano Microcontroller Board | 139 |
| Figure 49 L298N Dual H-Bridge Module | 139 |
| Figure 50 DC-DC Step down Converter..... | 140 |
| Figure 51 12 V Sealed Lead-Acid Battery | 142 |
| Figure 52 DCS-932 L WiFi Camera overlooking the room..... | 147 |
| Figure 53 MTM-Dispatcher Performance Graph | 154 |
| Figure 54 MP-Tracker Performance Measurement | 158 |
| Figure 55 Lucas Kanade Landmark identification..... | 160 |
| Figure 56 Active Landmark | 162 |

| | |
|----------------------------------------------------------------------------------------|-----|
| Figure 57 Original Image..... | 165 |
| Figure 58 Red Channel | 165 |
| Figure 59 Diff1 | 165 |
| Figure 60 Result..... | 165 |
| Figure 61 Four Active Landmarks Detected..... | 166 |
| Figure 62 First Camera Calibration | 168 |
| Figure 63 GeoGebra Simulation of Localization Experiments..... | 171 |
| Figure 64 3D Stability Plot of the Static Landmark Based Equations..... | 174 |
| Figure 65 Graph of the partial derivatives of the 3D stability plot at the center..... | 175 |
| Figure 66 Simulation of first Running Fix test..... | 176 |
| Figure 67 Landmarks Detected in Side View Experiment..... | 177 |
| Figure 68 Partial Derivative Stability Plot for Running Fix equations..... | 178 |
| Figure 69 $Abs(Lu(l_0+x, l_1+y)-Lu(l_0, l_1))$ | 180 |
| Figure 70 $Abs(Lv(l_0+x, l_1+y)-Lv(l_0, l_1))$ | 180 |
| Figure 71 Experiment Design for Localization with External Camera | 182 |
| Figure 72 Test1..... | 185 |
| Figure 73 Test 2 with corrected calibration | 185 |
| Figure 74 Test 1 Homographic projection | 185 |
| Figure 75 Test 2 Homographic projection | 185 |
| Figure 76 Center of the robot versus center of the perceived image | 186 |
| Figure 77 Direction from Odometry | 188 |
| Figure 78 Speed Odometry Calculations | 188 |
| Figure 79 Pan-Tilt mechanism puts all the weight on one servo's axis..... | 189 |
| Figure 80 Example of discontinuity for $\arctan(1/x)$ | 190 |
| Figure 81 Experiment Design for Navigation with External Tracking..... | 193 |

List of Tables

| | |
|--------------------------------------------------------------------|-----|
| Table 1 Speed of Light on a Different Media | 27 |
| Table 2 Wiring Table for Arduino Microcontroller | 141 |
| Table 3 Performance of sDOMO File Transfer vs. FTP Software | 151 |
| Table 4 MP-Tracker Performance Comparison Summary | 158 |
| Table 5 Results of First 3 Landmark Localization Experiment | 167 |
| Table 6 Results of Second 3 Landmark Localization Experiment | 169 |
| Table 7 Results of Third 3 Landmark Localization Experiment | 170 |
| Table 8 Results of Fourth 3 Landmark Localization Experiment | 172 |
| Table 9 Data Input for Calculation of Homography Matrix | 183 |

1 INTRODUCTION

1.1 Why

Anticipated for a long time in science-fiction literature, domestic robots are timidly starting to appear. While the first applications, like vacuum-cleaners or toys, do not share much with the versatile robotic servants envisioned in literature and movies, it is just a matter of time before more useful robots appear, mainly driven by the demand for help for the aging population in the industrialized world.

The UN report [22] presented by the Population Division on the 2002 World Assembly on Aging and their follow-up documents highlight an unprecedented level of aging showing that the population aged 60 and over is expected to grow from about 600 million at the beginning of the year 2000 to 1.5 billion in 2025. This demographic trend is also expected to coincide with a slower growth in young population driving up the percentage of elderly in the industrialized world from around 20% in 2000 to 30% in 2025 and about 35% in 2050 with certain regions experiencing much higher percentage. For example, in Europe in the year 2025 it is projected that there will be 2.6 elderly over 65 for every child below 15 years old. In the United States over the 2012–2022 period, driven mainly by elderly care, the national health spending is projected to grow at an average annual rate of 5.8 percent. By 2022 health spending financed from public dollars alone is projected to account for 49 percent of national health spending and reach a total of \$2.4 trillion as reported in [23]. These trends have the potential to put to the test the industrialized nation's ability to provide assistance to their aging population.

In this context, the technology is expected to play a major role in elderly care and

assisted living. An active area of research is already taking place for technologies to achieve what is called “Independent Assisted Living” where the senior citizens are enabled by technology to live alone in their own houses, as opposed to being moved into an assisted living facility, while being able to receive help as needed [14]. Domestic robots are expected to play a prime role in this field in the not so distant future [14, 12] as part of an integrated Domotic environment.

The task of providing Independent Assisted Living is eased also by the recent “silent revolution” in embedded systems and mobile devices helping the concept of Intelligent Houses to become a reality. Combining a ubiquitous sensor network in the house, with wearable devices and robots as the mobile component, for the first time in history, the concept of a house smart enough to take care of the residents is finally within reach. Just in time to help address the challenges we are facing due to a worldwide aging population. However, a few hurdles still remain that need to be overcome by engineering and research work.

One of the main obstacles faced by the development of mobile robots is the ability to properly operate in domestic environments where they must be able to safely navigate and avoid objects, people or pets [13]. While non-visual methods have been attempted, computer vision emerges as the most promising technology [3, 4, 5, 6, 15] but this brings with it the major challenge of processing in real-time the humongous amount of information captured by cameras on an energy efficient embedded computer.

A second major difficulty stems from the lack of a unified communication method between various home automation devices produced by different vendors. Nowadays, most manufacturers of home automation (domotic) devices rely on their own proprietary

communication protocols between devices. This approach has two major drawbacks: First, it limits the functionality that can be implemented into a domotic system to the capabilities provided by a particular vendor; a system cannot be extended with devices from another vendor if a particular function is not available from the original company because the device is not able to communicate with the rest of the system. The fact that the consumer is limited to the devices offered from the vendor of the system already installed in the house, does not provide the real competition required to bring the prices down.

As a direct result, today's market of domotic systems is composed of overly expensive disjoint subsystems that do not provide advanced functionality as needed for the vision of Independent Assisted Living to truly take place. New approaches to the problem are needed to provide an interoperable infrastructure allowing various subsystems to inter-operate seamlessly.

Another major problem with the home automation system is the problem of security and privacy. Domotic systems brings a whole new dimension to the security and privacy problem, very rarely encountered in computer security before. Since a home automation system can directly affect the living environment, a security breach from a person with malicious intent can do property damage and bodily harm to residents. With the advent of domestic robots, the problem is exacerbated by the fact that a compromised robot can even kill the unsuspecting inhabitants while asleep or eating food that has been tampered with. The big push from the industry toward the over-hyped “Cloud Based Internet of Things (IoT)” can be a recipe for disaster. The “Cloud Based IoT” has been shown to have multiple security and privacy issues needed to be fixed [24, 25, 26, 27, 28,

29, 30] and they can represent serious threats for smart homes of the future. Solutions that alleviates some of these concerns has been offered in many of the paper cited above, but so far most of the problems persist.

1.2 Contributions

In the current dissertation research, we studied, proposed and started to develop solutions to address many of the concerns above in order to help further development of domestic robotics and home automation systems. Our work resulted in four published research papers and some of the software has already been made available to the interested public under a liberal open source license.

We began by proposing a distributed computing architecture for domestic robot navigation. In this work we envisioned a system model where we proposed using various equipment already existing in a modern house (camera, wired and wireless network and house based computers) in order to aid robot navigation using both internal (on the robot) and external (fixed on the walls) cameras. The rationale for this architecture has been to take advantage of already existing equipment to keep the cost down, to outsource as much as possible of CPU intensive Computer Vision processing into the Base Station allowing us to drive the robot with cheap energy efficient embedded boards and provide a better situation awareness by integrating images from multiple cameras overlooking the same scene. The proposed architecture has been presented in our first paper from 2013: “A Distributed Processing Architecture for Vision Based Domestic Robot Navigation”.

During implementation of one of the components presented in this design, Camera Module (CM), we performed our major work in computer vision by creating the

MP-Tracker algorithm. MP-Tracker is an algorithm for detection and tracking of multiple object moving into the field of view of a fixed camera. The results of the tracking algorithm are to serve as the input for the robot situation awareness, being used by the robot both to detect its location and direction and to be aware about the other people, pets or robots moving inside the same room. Our second paper “A Multi-Paradigm Object Tracker for Robot Navigation Assisted by External Computer Vision” describes the algorithm that has been presented at ACM RACS conference in 2014.

Having multiple modules located across various computers in the network which need to cooperate with each other our project was in need of a communication protocol specially tailored for the task of home automation and robotic systems. After experimenting for quite some time with the well-marketed AllJoyn protocol specifically developed for the Internet of Things we found it severely inappropriate for the task, and decided to implement our own protocol specially designed for this kind of applications. The work resulted in what we consider our most important contribution so far within the scope of this dissertation: the sDOMO communication protocol. As a communication protocol specially developed for home automation and building of the robotic systems and released under an open source license, sDOMO is highly suited to provide the backbone for a highly integrated system for the intelligent homes of the future. Being optimized for small devices, sDOMO allows some 8 bit microcontrollers to be a full featured, independent node in the domotic network, yet the protocol is powerful enough to provide soft-real-time communication for our computer-vision distributed system for domestic robots.

sDOMO is not only a communication protocol but defines a data model for

highly integrated smart homes, where special software modules like House Intelligence Unit can be choreographed into coordinated actions between multiple devices and external information to achieve holistic behavior of the living environment. Using protocol adapters, sDOMO allows devices speaking through third party protocols to be integrated into the system appearing as sDOMO devices.

Being designed with security and privacy concerns in mind, sDOMO has unique features to protect the residents, some of them are not matched even by heavy-weighted competing protocols. For example, by protocol design sDOMO guarantees that any attempt to hijack a sDOMO device from a Trojan Horse or Virus infiltrating behind firewall in the domotic network will trigger automatic retaliation from the system against the attacker. From a privacy point of view, the protocol has provisions that makes it difficult for the manufacturers to attempt to spy on their customers, standardized XML files for automatically shaming on public forums rough companies are being considered as part of the protocol specification.

The work with sDOMO resulted in a companion website where the code and demos are being released. It also resulted in two papers “sDOMO – A Simple Communication Protocol for Home Automation and Robotic Systems” presented at IEEE International Conference on Technologies for Practical Robot Applications and “sDOMO Protocol in the context of the Internet of Things” accepted at CSTA-2016. I would like to mention that the work on sDOMO is planned to continue beyond the scope of this dissertation and will be released as an open source protocol for general purpose home automation and the building of robotics systems.

The usage of well-tried software design patterns and application frameworks is

often encountered in Mission Critical and Safety Critical Applications development due to the high stakes involved in case of failures. To increase reliability, some frameworks attempt to separate the implementation of business logic and low level implementation details and move the latter inside of framework-implementation in order to allow the developers to focus as much as possible on the problem to be solved, while still providing the necessary infrastructure for easy to use API's.

In a paper pending for submission, we proposed a brand new design pattern for Message Dispatching and Processing in a heavy multi-threaded application. The Multi-Threaded Message Dispatcher algorithm (MTMDDispatcher) is a reusable set of classes that implements an enhancement of well-known Reactor [56] design pattern so that it operates in a multi-threaded environment while using the Partial-Ordering Deadlock Avoidance algorithm to guarantee that no two threads dead-locks during process. The framework design uses modern C++11 techniques to encapsulate all the required mutex locking/unlocking in the resources access API managed by the Dispatcher itself preventing the users from making mistakes that can lead to a dead-lock. This way, junior programmers can take the task of implementing mission critical applications without risking the introduction of hard to debug anomalous interlocking behavior.

We fully derived the analytic geometry equations for landmark based localization with the camera located on the robot both for multiple landmarks and a stationary robot and for a single landmark and a mobile robot. The result of this work has been presented in this dissertation.

Another algorithm we created from scratch and is being presented in this dissertation is the Vector Classifier Algorithm capable of processing the noisy input results from the Optical Flow calculation from the moving robot camera, eliminating outliers and providing a clean set of motion vectors that can be used for visual odometry.

1.3 Literature Review

Robot navigation is a complicated problem and various solutions have been attempted to try to solve it, as we reviewed in our first paper. The computer vision however, eventually augmented with other equipment appears to be the leading technology for the task.

A combination of ultrasonic sensors, laser range-finders, and RFID tags were used [13] for indoor robot navigation without using computer vision. Stereo vision has been used for mapping [16] the data being structured as a 2.5D occupancy, elevation and slope grid. Davidson [17] presented a method to do real-time localization and mapping of the environment using a monocular camera, while [18] artificial landmarks are placed on the ceiling and a vertical looking camera is used to detect their position and orientation and infer the pose of the robot.

Visual sonar [21] is a relatively new technique which attempts to recover a depth of information from monocular images by deriving cues based on real life constraints used to eliminate the ambiguity inherent in monocular vision. Due to the large amount of processing required for computer vision, researchers have always tried to employ a whole plethora of methods to improve the localization by using various pre-defined or innate sets of knowledge about the environment, or by aiding the visual localization system with external information.

In [1] outdoor navigation and mapping, the computer vision is aided by a differential GPS and location information is processed by a distributed Extended Kalman Filter. Dead-reckoning is used by Cobos et al. [2] beside the Visual Odometer to help with robot localization. In Cluj-Napoca [3] they used a laser beam to detect dynamic obstacles, while a laser scanner has been employed by Biber, Fleck, and Duckett [4] to collect data for model building. High level prior-knowledge of the environment has been employed [15], where the indoor space has been modeled as horizontal and vertical planes having different orientations while the obstacles (objects) have not been modeled, only noted in the grid.

In line with our research, Pizarro et al. [5] used a rig of calibrated and synchronized cameras to achieve robot and obstacle localization with a collaborative system employing external cameras, also presenting a mobile robot in [6].

Tracking multiple persons / robots / pets and moving objects is an essential task for situation awareness in robot navigation and operation. It is also a relatively complicated problem of computer vision and multiple solutions have been proposed in literature. We approached the problem in our second paper [?] too. In the following paragraphs a short review of related work is presented.

The most common tracking methods are based on a variation of Multiple Hypothesis Tracking (MHT) [31] developed originally for RADAR by Donald Reid in his seminal paper from 1979 and adapted by later researchers for tracking using computer vision [32]. While deeply rooted in the theory of probabilities, MHT also needs to accumulate a relatively lengthy history before it can decide with a good enough confidence that a particular detected signal can be associated with a previously detected

one. As a result, MHT based trackers may exhibit a delay that is less than desirable in real-time tracking in close call situations.

The background subtraction and segmentation method has been presented in an introductory material in [33] and has been used by [34] for tracking vehicles on a highway, while an algorithm based on direct searching and failure recovery using histogram matching in HSV color space has been presented in [35]. Grouping features on hierarchical levels and using simulated annealing to find optimal configurations at object level has been presented in [36]. Lucas-Kanade method for tracking has been used in [37] taking advantage of a dedicated hardware to perform computationally intensive task while in [38] it has been combined with Histogram of Oriented Gradient based detection.

Today's market for domotic systems is dominated by proprietary communication protocols incompatible with each other. This not only prevents engineers from building highly integrated home automation and domestic robotic systems, but keeps prices artificially high due to lack of competition in the market since the customers have to restrict their choices to components compatible with the system already deployed. By developing sDOMO [62,63,64], we propose a communication protocol for fully integrated domotic systems running on top of house network, protocol which is simple enough to accommodate as peer's micro-controller based sensors but scalable enough to handle a house wide distributed computer vision system for domestic robot assisted living. Based on a house centered philosophy, the protocol is putting emphasis on privacy and protection of the residents.

A relatively similar architecture with ours has closely related security goals, has been presented in [39], however their reliance on public key infrastructure will not permit

smaller devices to connect directly with the rest of the network, therefore the system must employ Room Bridge acting as protocol adapters. While bridging architecture it is also supported by sDOMO for devices that are unable to speak to the protocol, our work reduced the protocol requirements to the basics, allowing much smaller devices to be part of the network directly.

To solve the interoperability problem between otherwise incompatible platforms in [40] the idea of a SOAP based middle-ware was proposed. A similar SOAP based middle-ware was presented in [44]. The high overhead and verbosity associated with SOAP however would require that any small sensor or device to depend on a more powerful adapter.

In [45] a microcontroller based module was presented but it relied on external serial-to-lan converter and no overall domotic network architecture has been attempted. An IP based home automation system was presented in [46] but the reliance on ssh excluded microcontroller based devices from working as peers.

A similar idea with our Home Intelligence Unit (HIU) was presented in [41] by Kao and Yuan where they developed a more elaborated and complex set of meta-rules than our HIU. We may be looking more in depth at their ideas for our future development of HIU.

The idea of a framework for developing domotic systems by following the model driven approach was introduced in [42] while [43] presented a state of the art report on home automation technologies.

The concept of the Internet of Things is a very broad concept covering various aspects

from embedded RFID tags to clothing till fully integrated city wide utility information systems [54]. Therefore, the scope of communication protocols used in IoT is equally wide. A summary overview of some of the architectures used in IoT has been presented in [55] while in [27] we see few critical considerations about the design of smart homes.

A survey [26] conducted at the university of Essex highlighted the major concerns of the people in regard with intelligent homes, among which are important to notice: The feeling of being in control, Privacy and Cognitive Workload. As we are headed toward pervasive intelligent environments, the human factor must be considered and I believe more studies of this kind are necessary for us, engineers, to understand the response of the users of our smart environments. Another interesting point about management of trust in “Cloud based IoT” has been made in [30] where they highlight that because some IoT company uses third party services for authentication (like for example Facebook or Google) if the user of the IoT service trusts the third party used for authentication it tends to project the same trust onto the Cloud service provider, despite the fact that there is no relationship between the two. This is yet another way in which Cloud based service providers can mislead users, with or without intent.

Extensible Messaging and Presence Protocol (XMPP) [47] developed originally for Instant Messenger - “Jabber” provides near-real-time data exchange via XML based messages. The protocol is implemented as an open standard, using a client-server architecture. Systems speaking, the protocol can be isolated from the internet by implementing them behind a firewall in a private XMPP server. However, the verbosity of XML and the lack of native binary data transfer (which must be encoded base64 inside XML) makes it difficult for it to be used for small microcontroller based devices.

Nest Service Data Model [48] is tied to direct Internet access. All Nest devices, connect directly to Nest Service which is hosted online by company servers which it uses to access the devices via web-site or applications. The system relies on JSON data packing using REST interface over HTTPS excluding small microcontroller devices from being able to speak to the Nest API directly. Hosting the data services on company websites also raises questions of privacy and security of personal information.

The Constrained Application Protocol (CoAP) [49] is a very interesting recent development in IoT. It appears to be even more compact than sDOMO fitting inside 8-bit microcontrollers. The minimally binary packed header allows very small CPU and bandwidth overhead but it lacks security and privacy features. The Message size in CoAP is also limited to the datagram size (about 1.1KB over UDP) while sDOMO uses the Message Carrier Packs to allow messages with a theoretical limit of 4GB. CoAP was designed to have its message be easily translated to HTTP by an adapter but does not in itself define a network model beyond that of device to device communication. However, some authors [50, 51] are working on expanding CoAP so this protocol's evolution may be of interest.

In [52] there was a framework presented, for connecting devices to “the cloud” via REST Web Services. In the proposed architecture sensors speaking their own native protocol are connected to the Internet via an adapter connected to an open source “Internet of Things” website that allows publishing sensor data using HTTP.

Originally, accepted with enthusiasm without too much attention to security and privacy, in recent years IoT has finally started receiving the needed scrutiny. In [53, 24, 25] the authors analyze the current state of cloud-based IoT and discuss a number of

considerations about it.

Taking advantage of the lesson learned from others engineers, experience is the main driver for using well known design patterns instead of “reinventing the wheel from scratch” and running the risk of wasting time solving the same problems and making the same mistakes. A whole set of design patterns are well known in literature from which here we are reviewing a few related with our work.

Reactor Pattern [56] handles concurrent requests delivered to an application, by synchronous demultiplexing them within the context of a single thread and delivering them to appropriate service handlers. The Reactor it is a very influential pattern and our MTMDispatcher can be viewed as a multithreading extension of it.

To handle concurrency Monitor Object Pattern [58] synchronizes execution to ensure only one method runs within an object at any given moment in time. Active Object Pattern [57] provides each object its own thread of control and decouples method invocation from method executions. A review of other very useful concurrency design patterns can be found in [59].

2 CONCEPTS ABOUT DISTRIBUTED SYSTEMS

A distributed computing system is a system in which the software required to solve a given problem runs on separate machines connected to each other over a network. In order to allow a problem to be solved on a distributed system, the problem is divided into a set of tasks and each task is allocated to one or more computers in the systems.

2.1 Distributed Systems Architectures

Availability of network connectivity enable software designers to create distributed systems where data processing afferent to a single computing goal is no longer constrained to take place on a single machine. There are various rationales for choosing the distributed computing model, among them we note:

- Management of access to resources that need to be shared between multiple computing nodes.
- Proximity to data resources. A node can engage in processing of a large amount of data locally then send it to the data integration node in a smaller amount as essential data derived from the larger input.
- Insufficient computing resources available in a given system. A node sends data to be processed to either a node that has higher computational resources or to a node that has a smaller workload.
- Real-Time requirements. The node must be able to receive and pre-process incoming data in real-time and then unload the workload to other machines for further processing to free resources to be able to receive new data.

- Redundancy for safety critical applications. The same set of data is sent to a (usually odd) number of machines implementing code intended to achieve exactly the same functionality but written by separate teams without any algorithm-design/implementation “cross pollination” between them. The results of the computation are compared by a voter machine accepting the data output generated by the majority of systems.
- Classified systems (Red-Black separation). Some algorithms or some special information (for example encryption keys) used to process the incoming data can be having a higher level of secrecy. The main computer after doing its own task will send data to be processed to a special classified computer (with self-destruct circuitry if tampered with) which is running the secret algorithms.

By looking at what operations are being performed on a distributed system we can classify them as two types:

- Specialized systems: Are systems where on each node only a certain subset of operations is taking place, for example an SQL server performing only database management or a classified system performing only data encryption.
- General system with pre-planned functionality: Are systems on which each node can perform all the functions required and what actually gets processed is based on the incoming request. For example, an enterprise portal in which each server has access to the same database and executes the request received from the user.
- General purpose systems with user programming: Are systems able to process any type of functionality by the virtue of its ability to receive scripted command or

programs to be executed along with the request.

The architecture of a distributed system describes the topology of the connected components and how they interact. There are a relative largely number of potential distributed architectures, here we are mentioning the most important of them.

2.1.1 Client Server Architecture

In Client-Server model, one or more nodes called Clients are configured to request computing services from a well-known machine called a Server. Due to its simplicity and robustness it is one of the most used models for distributed computing and it also serves as a variation to many other architectures.

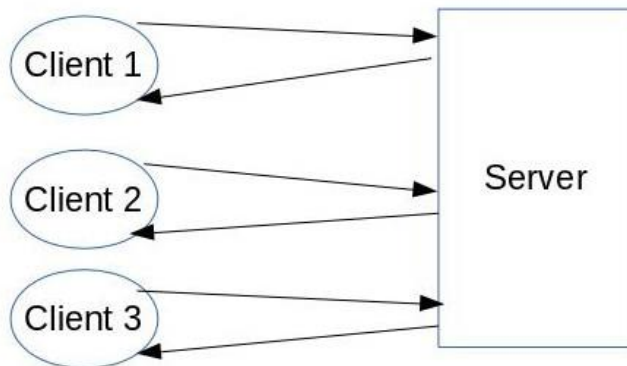


Figure 1 Client-Server Architecture

The server typically implements a set of well-known services and upon receiving a request it starts a procedure to perform the processing and return the results back to the client. The data to be processed can be received from the client, can exist onto a local data store like a file or a database or the server can also request-it from another server.

One of the disadvantages of the client-server model consists in the fact that the service becomes unavailable if the server is not functioning properly or if there are too

many client requesting simultaneously services from the server. To handle this challenge often the schema of Server-Replication is employed. In this schema multiple servers, sometime known as the Server Farm, are able to provide the same service and each requests are addressed to one of the servers based on it health and work-load. To implement this, various methods are available like a Facade server, Dynamic DNS queries or Load Aware routers.

Based on how the processing of the data is distributed between the client and server two major types of architectures are often recognized.

- Thin Client (Fat Server) model uses the server to perform most processing, the client is usually just requesting what type of operation is necessary.
- Fat Client (Thin Server) uses the server mainly as a provider/arbiter to the data resources performing most of the processing on the client itself.

2.1.2 Service Oriented Architecture (SOA)

It is a generalization of the Client Server model in which multiple servers can server data to multiple clients over the network. A Service is often described as a stand-alone function that can be invoked by the clients.

To facilitate discovery of what services are available instead of having the client know before-hand the full list of services, a server is providing discovery services. The client will query the discovery service with full or partial information about of what services it requires and receives in response the list satisfying the query.

In order to allow inter-operability between services offered by various vendors, standard protocols and meta-data defining the interface of services has been developed.

A Cloud Computing model is in essence just SOA distributed over the Internet offering: data-storage, specialized or raw processing or even full software packages as a service often on subscription basis.

2.1.3 Peer-To-Peer

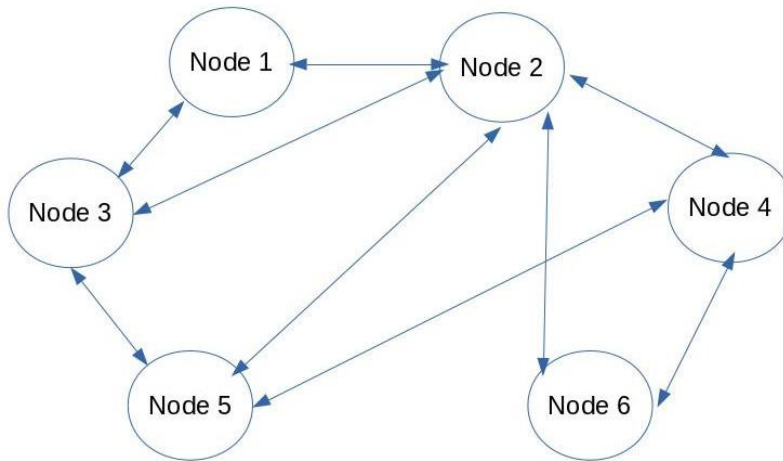


Figure 2 Peer-To-Peer Network

It is also a generalization of the client-server architecture, in which each computing node is running both as client and server on the same machine. The types of services offered by each node can be identical or different types of services can be offered by different nodes.

In addition to offering services to others, each peer can also implement operations to discover and keep the list of services offered by other nodes, presence and health monitoring and sometimes even providing dynamic load balancing of requests.

It is not necessary for each node to requests services from all the others however, that is definitely an option. A totally connected peer to peer networks of N nodes will have each node maintaining $N-1$ connection with its peers resulting in a total number of

$$\frac{N \cdot (N-1)}{2}$$

bi-directional connections.

2.1.4 Clusters and Grids

A computer cluster is a distributed system composed of multiple computers connected over a network and configured in such a way as to perform as a single large computer system. The most common application of clusters is high-performance computing (super-computers) with applications in scientific research and military applications.

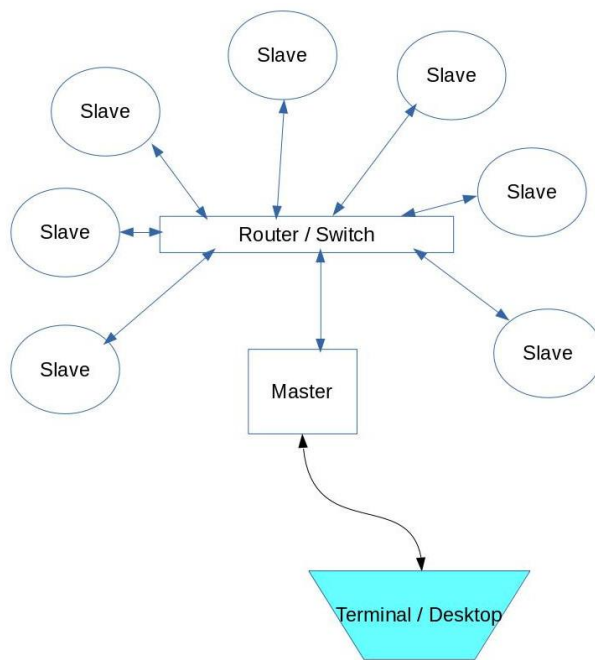


Figure 3 Cluster Computing System

Other applications include high availability (fault redundant) systems and load balancing server farms. Usually, each computing node in a cluster performs the same type of operation and they are located in relative proximity of each other connected by a high speed LAN and belong to the same administrative domain. The amount of inter-node

communication can be significant in clusters. Computing nodes in a cluster (Slaves) are often managed by a server called Master and the outside communications happen exclusively with the Master.

By contrast, Grid computing is a method to bring together computers from various administrative domains for the purpose of executing a particular task. In grids usually each node executes a different type of task and the communication between nodes is usually much reduced.

2.2 Data Exchange Models

A distributed system can also be regarded as a concurrent collection of processes interacting with each other over a communication medium. Therefore, a very important aspect of communication is the data exchange models implemented for Inter Process Communication (IPC). In a distributed system IPC it is used for both data exchange and for timing coordination between different processes, referred to as synchronization.

While at the lowest levels, the communication mechanism relies on packet exchange over underlying network technology, at the highest level there are three major IPC paradigms: Message Passing, Remote Procedure/Method Call and Distributed Shared Memory

2.2.1 Message Passing

The message is defined as a standalone block of information passed between processes. A message can be sent directly from a process to another in what is called a Point-to-Point Message Passing or it can be broadcast into the system in what is often referred to as a Publish-Subscribe model.

In the Point to Point (P2P) exchange, the message carries some information to

identify the intended target and is then delivered by the network stack to the target process. A popular variation of P2P method is the Request-Reply exchange in which messages are categorized as either a request for action or a reply to it and for each request received by a process it emits a reply either confirming the action or reporting an error.

The Publish-Subscribe method, the Subscribed process expresses its intention to receive a particular type of message while the Publisher broadcasts the message tagged with the mentioned type. The network stack looks up the list of subscribers for each message and delivers the message to them. The implementation of Publisher-Subscriber mechanism can either rely on a middle-ware (bus, message broker) or they can use the direct underlying broadcast or multicast capability of the network layer to distribute messages without the need for a middle-ware. While the second method can provide faster communication it requires the client to have the capability to implement more complex algorithms and be able to allocate enough resources to manage the subscriptions, especially if it is a protocol that requires a certain guarantee for Quality of Service (QoS). By contrast if a middle-ware is employed the clients can have very small capabilities since the middle-ware server (Hub) handles all the QoS requirements. This is the model implemented by our sDOMO protocol which allows very small devices to be full peers on the network.

2.2.2 Remote Procedure Call

The Remote Procedure Call (RPC) and its Object Oriented Equivalent Remote Method Call have been designed as a way to abstract the call of a procedure (function, method) across the network between two processes. In order to preserve the function, call

semantic, a client stub is implemented on the client side which serializes the parameters and initiates the call to the server. The server stub receives the data deserialization and it calls the actual procedure, the reply being sent back in the same way.

An important problem in implementing a RPC consists in the necessity to guarantee semantics of the function call that it either succeeds or returns an error code despite the unreliability of the underlying network. For example, a request for an operation reached the server but the response is lost on the network; in which case the client believes the operation never gets executed while the servers have the update already done. To solve the problem, most systems employs: Unique Request ID, Retransmission on Timeout and Acknowledgements.

Each request from the client to a server is allocated to a unique ID guaranteed to not roll-over into the amount of time a packet can still survive on the network. Each request it is ACK-ed by server upon reception and resources to hold the results are allocated. Then the server proceeds to execute the request. Once finished, the results are sent back to client which have to ACK the reception of the results.

Only when the server receives the ACK from the client the allocated resources used to store the results of the operation are finally released. If the client does not receive the ACK for the request it re-sends with the same id. The server recognizes a previously received request and will not execute the operation again but just re-send the cached results.

2.2.3 Distributed Shared Memory

It is a generalization over the network of Shared Memory paradigm implemented by

many operating systems as a form of Inter Process Communication. In most implementations the physical memory is not entirely replicated on every node but the system only implements a common logically shared memory space. The requests for a block of memory that is not located on the current system will be forwarded over the network to the node that actually manages it and only the requested block will be cached on both nodes.

The biggest advantage of Distributed Shared Memory is the fact that the details of communication are totally hidden from the programmer which does not have any more of the job to handle the data communication. However, this is also the biggest drawback because hiding the cost of communication can lead to inefficient algorithms if the programmer writes the code under the assumption that the cost of accessing any block of memory is virtually free.

2.3 Distributed Computing Considerations

2.3.1 To Distribute or not

An important question in distributed computing is to find out if it makes sense to use a distributed processing model as opposed to local processing. For example, let's consider a server that is twice as fast as the client, sending an operation that takes 3 milliseconds on the client to the server does not produce any benefit if the time required to request the operation over the network takes 2ms since the answer will be received after 3.5 ms, later than if executed locally. However, for an operation that takes 1 second to be executed on the client it makes a lot of sense to request-it remotely because the response will be

received back in 0.502 seconds faster than if it was run locally.

Let's consider a system consisting from a client C able to process N_C instructions/second and a server S capable of N_S instructions per second, with $N_S > N_C$. If the average amount of time to transmit the data from client to server is T_T and to get the results back is T_R . If the problems to be solved need P operations to be performed, the times of solving it locally as opposed to solve it by server request will be:

$$T_C = \frac{P}{N_C}$$
$$T_S = T_T + T_R + \frac{P}{N_S}$$

where T_C and T_S are the times of solving on the client and server respectively.

It is obvious that it makes sense to send the problem to be solved on the server only if $T_S < T_C$ therefore it makes sense to request remote execution if:

$$P > \frac{(T_T + T_R)N_CN_S}{N_S - N_C}$$

When solving a problem composed from two independent sub-problems P_1 and P_2 the most important factor in deciding whether to distribute or not is the ability of the two problems to be solved independently. More precisely, if P_2 needs as input the results generated by P_1 then for each problem we have to apply the rationale from the equation above.

However, in case that the two problems are independent of each other it often makes sense to send the longest problem P_2 on the more powerful computer. The only time when this is not a good idea is when the transmission time over the network is very

high:

$$T_T + T_R > \frac{N_S P_1 + N_S P_2 - N_C P_2}{N_S N_C}$$

2.3.2 Data-link Considerations

The transmission times T_T and T_R are dependent on the characteristics of the datalink.

Any segment of the network is characterized by two main parameters:

- Speed of the data-link – how fast a bit sent at the endpoint A reaches the other end B
- Bandwidth – how many bits per second can be sent over the data-link (data rate of the link).

For a simple data-link consisting by a simple transmission medium between points A and B the speed of the link is dictated by the wave speed (speed of light, c) on that particular medium.

The propagation delay δ is the time that it takes for the signal to reach from point A to point B on a data-link having the length D .

The transmission time τ is the time it takes to transmit a given amount of data S_z over a link with a given bandwidth B_w .

Table 1 Speed of Light on a Different Media

| Medium | Speed of light (c) |
|---------------|--------------------|
| Vacuum | 299792 km/s |
| Air | 299700 km/s |
| Optical fiber | 200000 km/s |
| Cooper wire | 230000 km/s |

Therefore, we have the basic network performance relations:

$$\begin{aligned}\delta &= \frac{D}{c} \\ \tau &= \frac{Sz}{Bw} \\ \Lambda &= \delta + \tau\end{aligned}$$

which defines the Latency Λ as the time it takes for receiver B to get the data packet of size Sz transmitted by the sender A over the distance D . The values for T_T and T_R from the equations from the sub-chapter above are in fact just Λ for the given size of the transmitted respectively received data.

Another important metric in networking is the link capacity or Delay x Bandwidth product. It is often referred to as the amount of data that can “fill the line” or the number of “bits in flight” on the line between sender and receiver. Technically it represents how many bits of data the sender can put on the line before the first bit is received by the receiver. When designing a communication protocol that requires acknowledgements, it is important to have an understanding of the link capacity because defining the size of the

data packet that needs acknowledgement lower than the link capacity will just result in wasted bandwidth while waiting for the ACK for a packet that arrived properly. On the opposite approach, sending packets larger than the link capacity will just result in wasted bandwidth in case that the packet encounters transmission errors and a negative acknowledgement (NACK) is required. Therefore, it would be optimal to send data packets at exactly the link capacity.

If the network contains one or more switches or routers the queuing and retransmission time must be added to the total value of Λ . Let's consider the following segment of the network where between transmitter and receiver there are N repeating switches each of them having a data queuing time Q_i between receiving a packet until it starts retransmitting it.

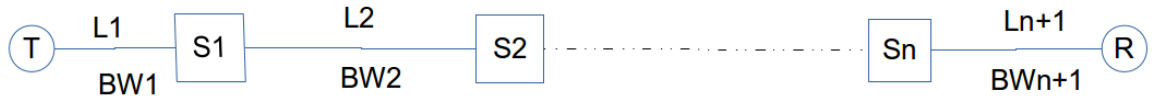


Figure 4 Network Segment with Switches

If the transmitter starts sending a packet of P bits at the moment 0, it finishes sending at P/BW_1 and the packet is received by the switch S1 at $P/BW_1 + L_1/c$. S1 will therefore start sending at $P/BW_1 + L_1/c + Q_1$ and so on. The receiver R will therefore receive the packet addressed to it after:

$$T_T = \frac{P}{BW_{n+1}} + \frac{L_{n+1}}{c} + \sum_{i=1}^n \left(\frac{P}{BW_i} + \frac{L_i}{c} + Q_i \right)$$

therefore, the end to end link throughput will be calculated as the rate at which the P bits arrived from T to R therefore it is:

$$BW = \frac{P}{T_r} = \frac{P}{\frac{P}{BW_{n+1}} + \frac{L_{n+1}}{c} + \sum_{i=1}^n (\frac{P}{BW_i} + \frac{L_i}{c} + Q_i)}$$

3 CONCEPTS ABOUT COMPUTER VISION

Computer Vision is the method of processing images acquired by a camera in order to extract information that can be used directly by computers. Computer Vision is emerging as a leading technology in robotics. The main advantage is the fact that a single sensor, camera, can be used for multiple purposes including but not limited to: collision avoidance, Odometry, localization and object recognition.

In this chapter we will provide an overview of major Computer Vision (CV) techniques used in this work.

3.1 MOG Background Subtraction and Segmentation

Background subtraction attempts to identify moving objects by integrating a model of the background over time and then subtracting the current frame from it, which will highlight the places in the image where change happened. The variation of the aspect of pixels in time and changing illumination conditions however makes the method difficult to apply.

To solve these issues various ideas has been attempted from which one of the most successful [65] proved to be modeling the background as a Mixture of Gaussians (or often referred in literature a Gaussian Mixture Model). In this method, each pixel is represented as a mixture of K Gaussians; at any moment in time t the probability of observing the current pixel value it is:

$$P(X_t) = \sum_{i=1}^k \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

where $\{X_1...X_t\}$ is the recent history of the pixel, $\omega_{i,t}$ is the weight associated with the i^{th}

Gaussian, $\mu_{i,t}$ being the mean value of the i^{th} Gaussian, and $\Sigma_{i,t}$ is the covariance matrix of the i^{th} Gaussian. The Gaussian probability density function being given by the formula:

$$\eta(X_y, \mu, \Sigma) = \frac{1}{\sqrt{|\Sigma|(2\pi)^n}} e^{-\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1} (X_t - \mu_t)}$$

The value of K – the number of Gaussians in the mixture is determined by the available CPU power and desired compromise between performances versus accuracy. In practice $3 \leq K \leq 5$ is often used.

The update procedure for a new pixel value is using a running average formula to recalculate the weights:

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha M_{k,t}$$

where α is the learning rate and M is 1 if the pixel matched the Gaussian in mixture and 0 otherwise. In Open CV C++ library the algorithm is implemented by the class *cv:BackgroundSubtractorMOG*.

3.2 Shi-Tomasi Corner Detection

Corners that are stable in respect to variations in image are an important feature for calculation of Optical Flow, Tracking and object matching. One of the earliest work in good features to track came from H. Moravec [66] and later improved [67] by Harris and Stephen.

A corner is defined as a feature with a large variation of the weighted sum of square differences (SSD) when a patch of an image (u,v) is shifted by the vector (x,y) .

$$SSD(x, y) = \sum_u \sum_v w(u, v) \cdot (I(u + x, v + y) - I(u, v))^2$$

which can be approximated by Taylor expansion as:

$$SSD(x, y) \approx (x, y) A \begin{pmatrix} x \\ y \end{pmatrix}$$

where A being the structure tensor

$$A = \sum_u \sum_v w(u, v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

with I_x I_y being the partial derivatives of the image I. The matrix A should have two large eigenvalues λ_1 , λ_2 for any point of interest (a good corner). Shi-Thomasi algorithm computes directly the value of $\min(\lambda_1, \lambda_2)$ and gets corners that are offers relative good stability for tracking.

3.3 Lucas Kanade Sparse Optical Flow

Lucas-Kanade method [68] uses spatial intensity gradient information to direct the search for the match of the features. The algorithm it is often used with Shi-Tomasi corners but other types of corners can be used too.

In the simplest form, LK method attempts to minimize the error:

$$E = \sum_x [F(x + h) - G(x)]^2$$

where G(x) is supposed to be the F(x+h), the image F shifted by the vector h.

Approximating from truncated Taylor decomposition

$$F(x + h) \approx F(x) + h \frac{\partial}{\partial x} F(x)$$

we get the solution for calculating the displacement vector as

$$h = \left[\sum_x \left(\frac{\partial F}{\partial x} \right)^T [G(x) - F(x)] \right] \left[\sum_x \left(\frac{\partial F}{\partial x} \right)^T \left(\frac{\partial F}{\partial x} \right) \right]^{-1}$$

The solution can be generalized for any linear transformation $G(x)=F(xA+h)$ with A being a matrix expressing the transformation between F and G . A fast implementation using pyramids [69] of the solution for generalized problem has been discovered in 2001 by J. Bouguet.

3.4 Homography

In Computer Vision homography refers to geometric transformation that relates two views of the same planar scene. More precisely, in homogenous coordinates a homography is a transformation given by the equation:

$$\begin{bmatrix} cx_b \\ cy_b \\ c \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x_a \\ y_a \\ 1 \end{bmatrix}$$

One of the simplest methods to calculate the homography matrix (often named H) is Direct Linear Transform (DLT) algorithm. With elementary linear algebra operation, the equation above can be rewritten as: $A_i \mathbf{h}=0$ where vector \mathbf{h} is

$$\mathbf{h}=(h_{11},h_{12},h_{13},h_{21},h_{22},h_{23},h_{31},h_{32},h_{33})^T$$

and matrix A_i will be defined by:

$$A_i = \begin{bmatrix} -x_a & -y_a & -1 & 0 & 0 & 0 & x_b x_a & x_b y_a & x_b \\ 0 & 0 & 0 & -x_a & -y_a & -1 & y_b x_a & y_b y_a & y_b \end{bmatrix}$$

Since the scale cannot be determined from any projection, in the equation above we practically have only 8 degrees of freedom, therefore we can calculate the Homography matrix having 4 matching points.

3.5 EPIPOLAR Matching

In the epipolar geometry technique two or more cameras oversee the same scene from two different points, they can have different orientations. It can be viewed as a generalization of homographic transformation by removing the requirement of the viewed points to all reside on the same plane. The name came from the **epipole** which is the point of intersection between the segment joining the camera centers and the plane of the image for each camera.

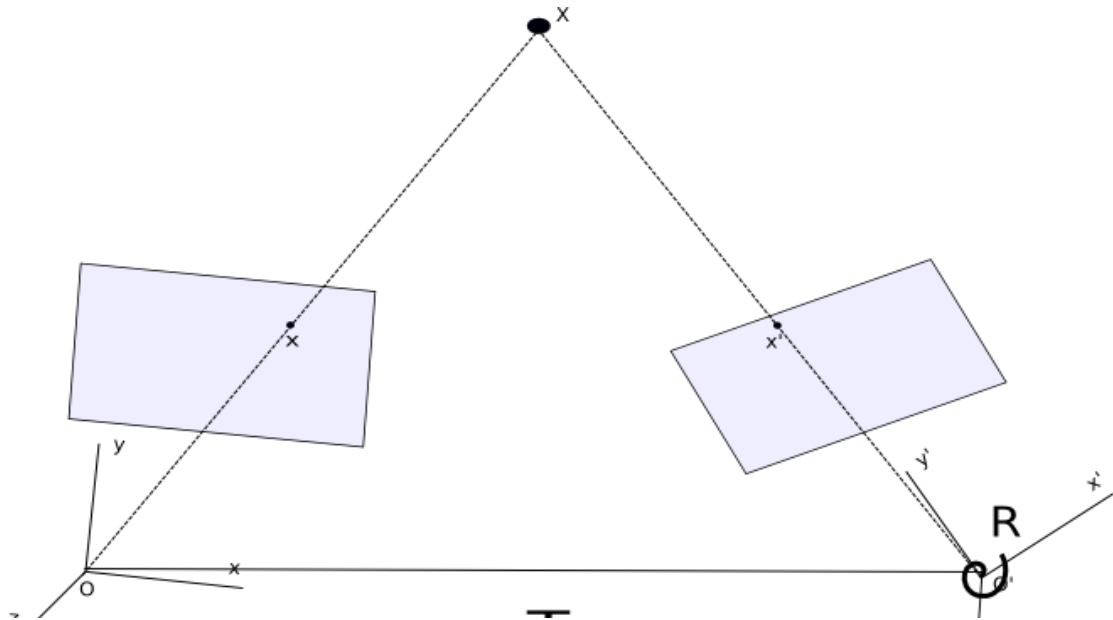


Figure 5 Euclidean Relationship between Two Views, Epilolar Geometry

The point X is projected on the first image plane as x and on the second image as x' respectively. The two points are being related by the equation

$$x'^T E x = 0$$

where E is the Essential Matrix:

$$E = \hat{T} R.$$

Another important matrix encountered in epipolar geometry is the Fundamental Matrix. The major difference between them is that an Essential Matrix is defined on a set of normalized coordinates while Fundamental Matrix does not need the cameras to be already calibrated. If we know the calibration matrices for both cameras, they being A_1 and A_2 we can calculate the Fundamental matrix from Essential and reverse by the formula:

$$F = A_2^T E A_1^{-1}$$

Once a minimum of 8 matching pairs of points in the two images have been detected by a feature tracking algorithm the essential matrix E can be computed using the “Eight-point Algorithm” [19]; and from it the position and orientation of robot camera in respect to a fixed camera can be recovered using Singular Value Decomposition (SVD) method.

4 CONCEPTS ABOUT ROBOTICS

Robotics and Automation are the fusion between computer science, mechanical engineering, electrical engineering and electronics. The difference of what a robot is and what an automaton is, is still subject to debates and each engineer has their own opinion about the subject. From my point of view, to declare an electro-mechanical system to be a robot, it needs the ability to adapt its *modus operandi* to (try to) cope with changes in the environment.

Regardless of the exact definition, robotics and automation both rely on controlling mechanical components with software via electronics. Therefore, a major part of the robotics consists in motor control, voltage conversion and trajectory calculations to allow a mobile robot to move at the intended destination.

In this chapter, we are looking at a few aspects of robotics that has been used in our work that resulted in building of our robot ROBI-1.

4.1 Differential Drive Kinematic

In the proposal from our first paper and our early experiments, we attempted to use a robot with Ackerman steering and not with differential drive. Our work was plagued with problems derived from this choice. First of all, we had a very hard time finding a decent mechanical platform with reliable Ackerman steering for our experiments. We ended up “hacking” remote controlled vehicles from various toy-shops and they proved to be unreliable. To make matters worse, all of those RC-cars we experimented with used proprietary steering servo-motors (not the standard hobby grade servos that are well documented) and we wasted a huge amount of time reverse-engineering those servos

(burning a couple of them in process), just to find the overall platform to not be reliable enough for our needs.

Even after we got it to work, we had found that the trajectory restrictions on Ackerman kinematics makes the robot very difficult to control on the domestic environment where it just ended up bumping into furniture. Long story short: We found out the hard way why no robot intended to operate in domestic environments (that we can find references about) has ever been built using an Ackerman drive. We ended up buying a differential platform that worked as expected from day one.

A Differential Steering Platform has two motor wheels commanded independently by two electric motors, and a third free running wheel which also freely rotates on the vertically mounted axis. Because of its free running, and ability to pivot to follow the motor wheels, the third Free Wheel has no role in steering and was exclusively added for the stability of the platform.

The electric motors are in most of the cases DC motors due to their ability to be easily controlled in speed and torque by variation in the applied voltage and because the natural presence of DC from batteries. In most of the cases a reduction gear-box is used to decrease the angular velocity and increase the torque.

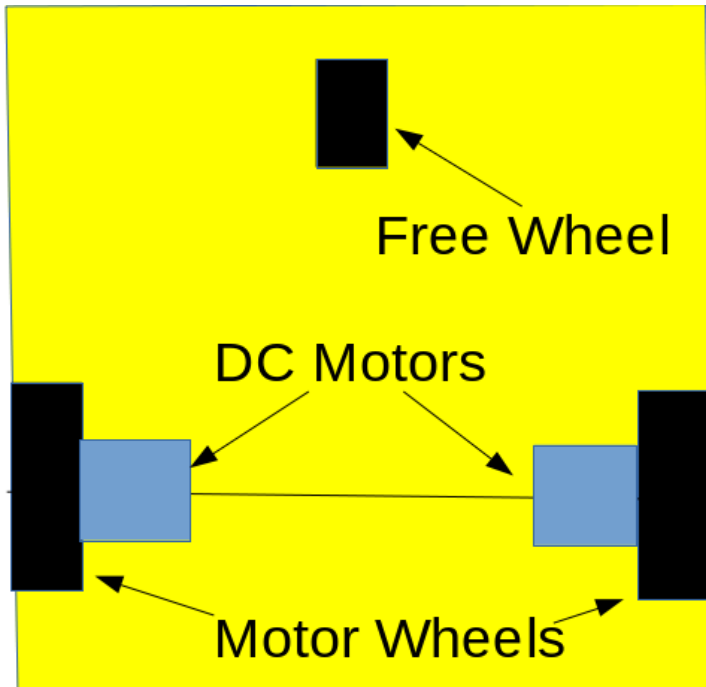


Figure 6 Classical Differential Drive

One of the first things we had to do was to derive the control equations for our differential drive which we present here. As a result of kinematic derivation we get two sets of equations: The Direct Equations allowing us to calculate where the robot ends if commanded with a well-known pair of velocities, and the Control Equations which tells us what velocities we should request in order to achieve a given trajectory.

For deriving the kinematic equations, we select the center of coordinate at the center between the two motor wheels. Because the third wheel has no functional role in steering, it is not represented in the kinematic model and only the two motor wheels are taken into consideration. In the diagram below the left and right motor wheel of the robot are located at points P_L and respectively P_R , therefore the robot is facing the x axis positive direction.

Because the wheels have different velocities in the same amount of time, the robot will move on a circle centered in the point C. In the amount of time Δt the center of axis between wheels will travel from the point O to O' while the wheels will travel to the points P_L' and P_R' respectively. In the following derivation we will use the following notations: The distance between motor wheels |P_LP_R| called the width of the train is referred to as **w** while the radius of the circle of turn |CO| will be referenced as **R**.

39

$$|arcL| = V_L \Delta t = \alpha(R - \frac{w}{2})$$

$$|arcR| = V_R \Delta t = \alpha(R + \frac{w}{2})$$

Without losing generality we can assume $\Delta t = 1$ and after elementary transforms gives as the direct set of equations:

$$\alpha = V_L - V_R / w$$

$$R = w \frac{V_L + V_R}{V_R - V_L}$$

$$\beta = \frac{\alpha}{2}$$

Expressing V_L and V_R from Direct equations we get the set of Control equations, which allows us to calculate what commands we need to send to each wheel to reach the destination point with the desired velocity of the platform:

$$V_R = V_O + \beta w$$

$$V_L = V_O - \beta w$$

Where V_O is the intended instantaneous velocity of the robot, defined as the velocity of the median point between the 2 wheels.

4.2 Controlling DC Motors

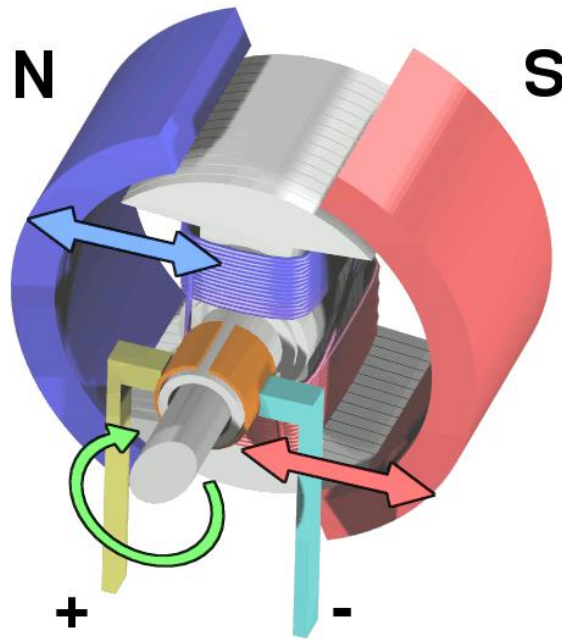


Figure 8 Simple DC Motor. (Courtesy Wikimedia Commons, Author: Webcaplet)

Each motor wheel of the robot is spun by a DC motor. Most DC motors for hobby applications use the classical design of a permanent magnet stator and a rotor switched by a rotary commutator mounted on the rotor axis.

The movement is generated by the interaction between the magnetic field of the stator and the magnetic field created by the electric current passing through the coils of the rotor.

A highly prized characteristic of the DC motors is the linear curve, i.e. the direct proportionality of the speed of the motor with the voltage applied for a given torque, direct proportionality of the torque with the current through the windings and linear

relation between the speed and the torque for a given voltage.

These particularities combined with the fact that the direction of rotation can be controlled by polarity of the power, makes DC motors highly prized for controlling applications especially in vehicles, tools and robots.

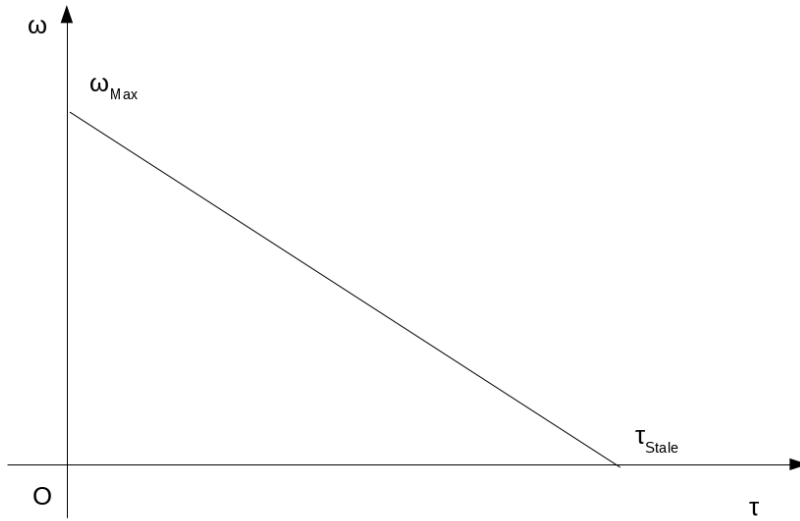


Figure 9 Speed-Torque Characteristic Curve for DC Motor

Each DC motor is characterized by an EMF constant k and its magnetic flux Φ . If I is the current through its stator windings and V the voltage applied, the equations for the torque and angular velocity are:

$$\tau = \frac{k}{2\pi} I \Phi$$
$$\omega = \frac{V - RI}{k\Phi}$$

From the equations above we see that when the motor has no load, it can reach a maximum speed $\omega_{Max}=V/(k\Phi)$ while the maximum torque $T_{stale}=kV\Phi/(2\pi R)$ is achieved when the motor is prevented to rotate by an overload.

The easier way to control a motor would be to insert a variable resistor (or resistor like circuit) in series with the motor, however this has a huge disadvantage. When the resistor is not toward zero a lot of the energy will be wasted on the resistor itself heating it. To solve this problem, most modern motor controller's employs Pulse Width Modulation (PWM) technique.

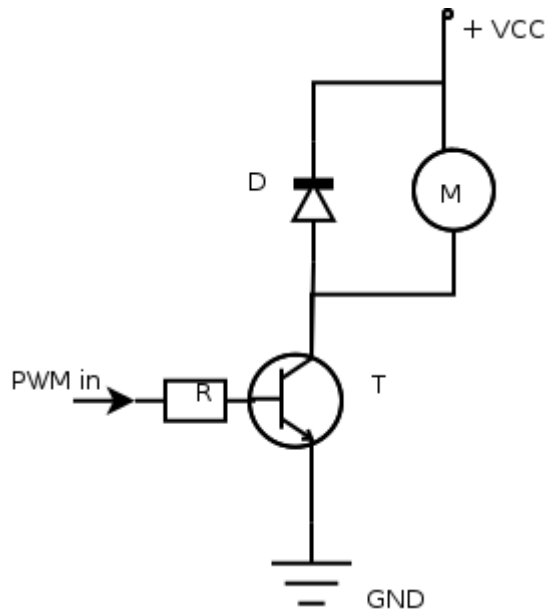


Figure 10 Controlling a DC Motor with a Bi-Polar Transistor

The Motor M is connected in the collector of a transistor T which receives PWM input via an optional resistor R. The role of the fast diode D is to protect the transistor against high EMF voltage induced into the coil of the motor when the voltage goes back to zero. The signal applied to the transistor on the base consists from a set of pulses (rectangular waveform) with the same frequency but with various fill factors (also known a duty cycle). The signal varies very fast between 0V (forcing the transistor to be cutoff), and a voltage high enough to send the transistor in saturation making the transistor to act as a

switch.

The fill factor (duty cycle) represents the percent of how long the signal is set to high in respect to the whole period of the signal.

The frequency of the signal is selected to be high enough so that due to inductance inertia the current in the motor cannot follow the variation of the signal, practically the coil inside the motor is being used to integrate the signal into a voltage that is equivalent to:

$$V = \text{DutyCycle} * VCC$$

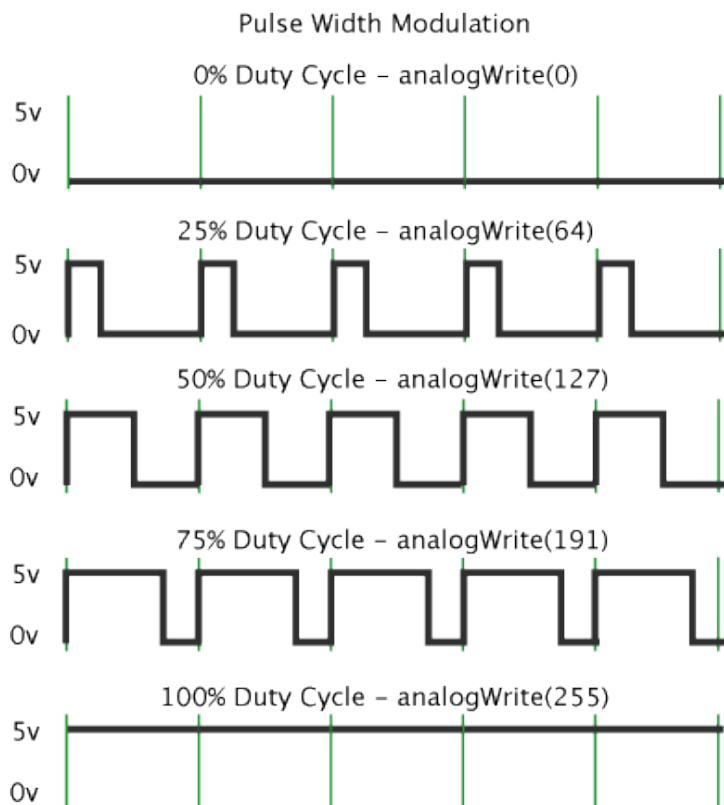


Figure 11 PWM Modulation (Courtesy to Arduino community from an article written by Timothy Hirzel)

Besides controlling the speed of rotation, the ability to control the direction of rotation is

essential for most robot operations. Since reversing the direction of rotation of a DC motor implies reversing the polarity of the applied voltage, a typical method employs an H-Bridge.

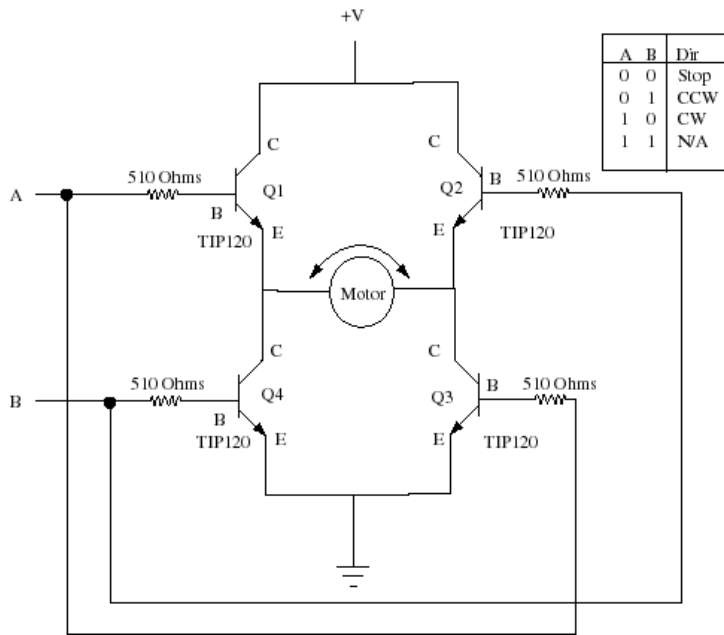


Figure 12 Typical implementation of an H-Bridge. (Courtesy Next Electronics, Greece).

The typical H-Bridge operations uses 4 transistors (operating in switching mode) to control the direction of current through the motor. For example, applying a high voltage on control point A and low voltage on B brings the transistors Q1 and Q3 in conduction and Q2 and Q4 in blocking state, forcing the current to flow through the motor from Left to Right. Similarly having A-Low and B-High forces the current to flow from Right to left having Q2 and Q4 in conduction while Q1 and Q3 being blocked. Nowadays, H-Bridges are available in integrated circuits including all the control logic inside a single chip. Various vendors sell ready to go modules containing the circuit and the protective diodes, dedicated logic to select the direction and PWM input for speed control and even

voltage stabilizers for powering up the TTL control circuitry. We are using such a module that is built around the Dual Bridge L298N for controlling our robot.

4.3 DC-DC Buck Converter

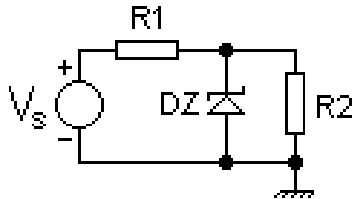


Figure 13 Typical Zener Circuit. (Courtesy Wikimedia)

Most electronics need stable 3.3V or 5V; however, in order to drive robot motors often a battery with a higher voltage is required. Therefore, every robot needs circuitry to step down the voltage for the electronics and guarantee a fixed level.

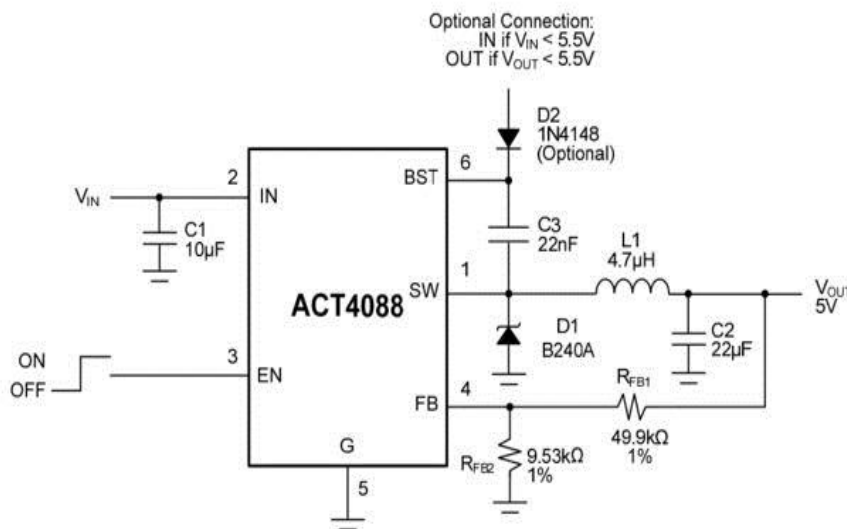


Figure 14 Typical Buck Converter. (Courtesy Instructables, Author: xKOBAYASHIMARUx)

The easiest way to achieve this, is to implement a linear regulator using a Zener diode. The drawback of a simple Zener regulator is its degree of high inefficiency. For example, providing 5V from a 12V source with a Zener circuit will never achieve a conversion efficiency above 40% at its peak, practically wasting over 60% of the power. On a device relying on batteries this amount of waste it is unacceptable.

Buck DC-DC converters achieve efficiency from 90% to 98% by using switched mode conversion taking advantage of the inductance inertia. An electronic (often a FET transistor) switch is controlled by monitoring the voltage on the load connected in series with an inductance. When the switch allows the current from the source to flow the inductance stores energy in its magnetic field and releases it on the load via a diode when the switch is disconnected. Due to their efficiency Buck converters are ubiquitous in battery powered equipment.

4.4 Controlling Servo-Motors

Servo-Motors are composed of a DC electric motor, gear-box, an angle encoder (often a resistive potentiometer) and control circuit board all of them into a very convenient enclosure. Unless regular motors, servos are usually not providing continuous motion but proportional positioning often in the range 0...180°.

Most hobby grade servos are connected via three wires: Ground, Vcc and Control line where the commanded value of the angle is encoded as a PWM signal. The desired angle is commanded by the value of the fill-factor (duty cycle) of the PWM signal.



Figure 15 A Servo-Motor Disassembled. (Courtesy to Seattle Robotics Society)

The servo motor is powered by the voltage applied between Vcc and Ground lines and the Control line it is used to specify the desired angle.

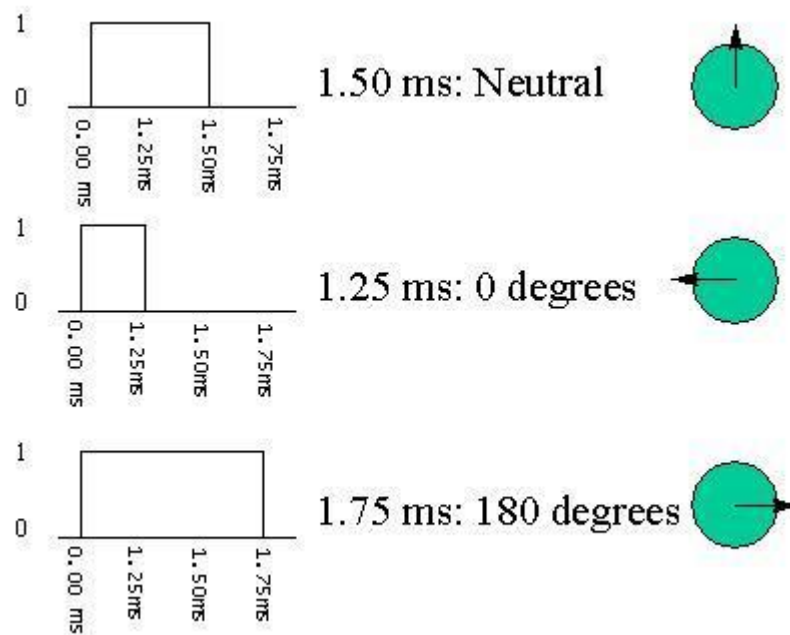


Figure 16 Servo Angle Control via Fill Factor.(Courtesy to Seattle Robotics Society)

The Control board inside the servo, usually implements a PID controller comparing the reading from the potentiometer with the prescribed value and adjusting the current passed to the motor to correct any discrepancy.

Various sizes of servo motors are available nowadays on the market and they are essential parts for any robot.

4.5 PID Controller

The Proportional-Integrative-Derivative (PID) controller is a method used in the process control to maintain a prescribed value (setpoint) of a functional parameter of a process (often referred in literature as a plant).

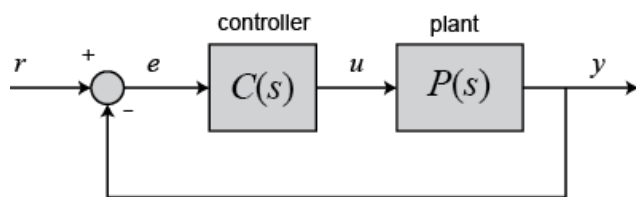


Figure 17 Feedback Control System

The PID controller is a feed-back mechanism continuously comparing the measured process parameter and the desired value. The difference is called the error and the controller tries to minimize the error by adjusting one or more input values into the process.

The output of the process y is measured or estimated and subtracted from the desired value r . The difference e is used as the input of the controller which in turn generates the control signal for the process u .

In PID controller the control signal is calculated as:

$$u(t) = K_p e(t) + K_I \int_0^t e(x) dx + K_D \frac{de(t)}{dt}$$

Where the parameters K_P , K_I , K_D are the weight coefficients for the proportional, integrative and respective derivative terms. These coefficients determine the behavior of the controller and are either calculated from the exact parameters of the plant or they are subject to tuning.

In the case of digital control systems, like the one used in our robot, the integral is implemented as a summation of the product between error and the sampling time.

5 SYSTEM ARCHITECTURE

Originally presented in our first paper [60] the system design consists on a set of independent software modules communicating with each other over the house network. The modules are communicating with each other via our sDOMO communication protocol presented in [62, 63].

5.1 Overview

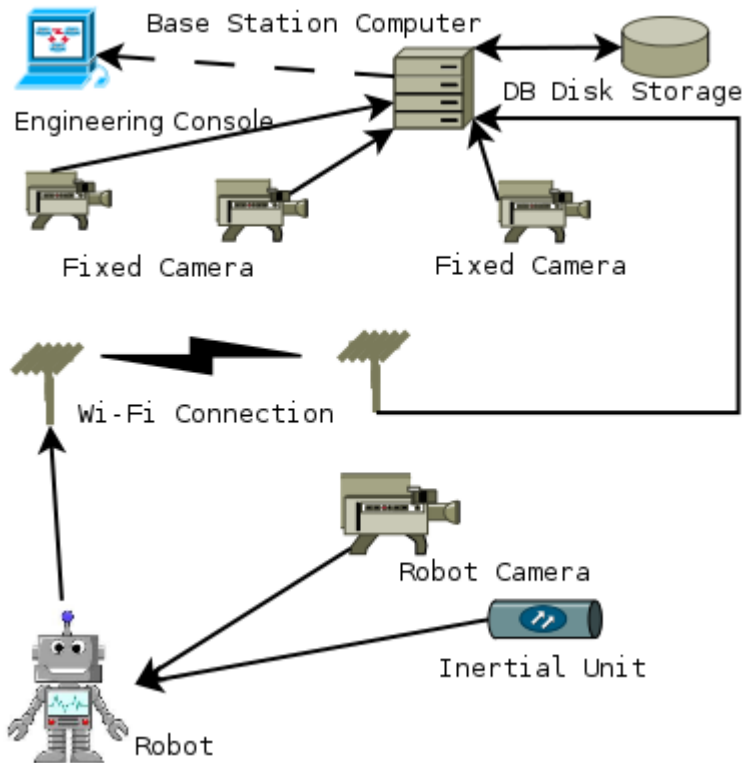


Figure 18 Overview of the System

The proposed system consists of a fixed unit (computer system referred to as Base Station) connected to a number of wired and/or wireless IP cameras overlooking the operating space (inside of the building) and one or more mobile units (robots) navigating inside the building. The communication between Base Station and robot it is using

existent WiFi network inside the house.

Each robot is equipped with a (monocular or a stereo) camera, and optionally other sensors like ultrasonic proximity sensors or Inertial Measurement Unit consisting of MEMS accelerometers and gyroscopes. After originally experimenting with an Ackerman Steering we ended up switching to the classical Differential Drive which proved more appropriate for navigating on indoor environments.

The video feed from the fixed cameras is continuously received by the Base Station. The Base Station uses the video streams from fixed cameras to perform object tracking and recognition, and to maintain the current 3D model of the environment by keeping track of the objects or inhabitants.

The images from the robot camera are processed on board by the embedded computer which sends in real time to the Base Station only a status vector. The robot is continuously doing on board optical flow calculations which are used to immediately react to potential collisions or to stay on the prescribed trajectory when out of sight from the fixed camera.

Base Station can request from the robot either the latest image or a specified sub-region of the latest image in order to do epipolar calculations. Upon a successful matching, the Base Station provides the robot with better position estimates. The robot is capable of dual mode navigation: autonomous (reactive mode) and guided (map based).

Reactive mode navigation is used exclusively for short distances when the robot is outside of the view of fixed cameras. It mainly consists of maintaining the prescribed direction of movement by processing the Optical Flow while looking at the floor or walls.

In guided navigation, a component of Robot Control Software running on the

Base Station it is using the map of the environment to prescribe robot trajectory. It is important to know however, that even in map based navigation the ability of maintaining a given trajectory still depends on the on board optical flow processing since contrary to our original expectation, the ambiguity in tracking proved to be way too big to be able to control the robot trajectory by tracking alone.

5.2 Software Architectures

A modular software architecture has been developed where different tasks are assigned to specialized modules. The software modules communicate between themselves with the newly developed sDOMO protocol.

The images from fixed cameras are processed by Camera Module (CM) with a correspondence of one running module for each fixed camera. Each Camera Module, running on the Base Station, is responsible for image pre-processing and tracking of moving objects. At every frame CM emits a Notification, processed by the Situation Awareness Module (SAM), containing basic status information of each tracked object.

SAM maintains a map of the environment, robots, and non-robot animated entities (humans, pets, or other moving objects) and as much information as the system is able to gather about the relatively fixed but movable objects (chairs, tables etc). When a new moving object is detected SAM can request from CM sub-images of moving objects for deeper analysis and eventually image recognition.

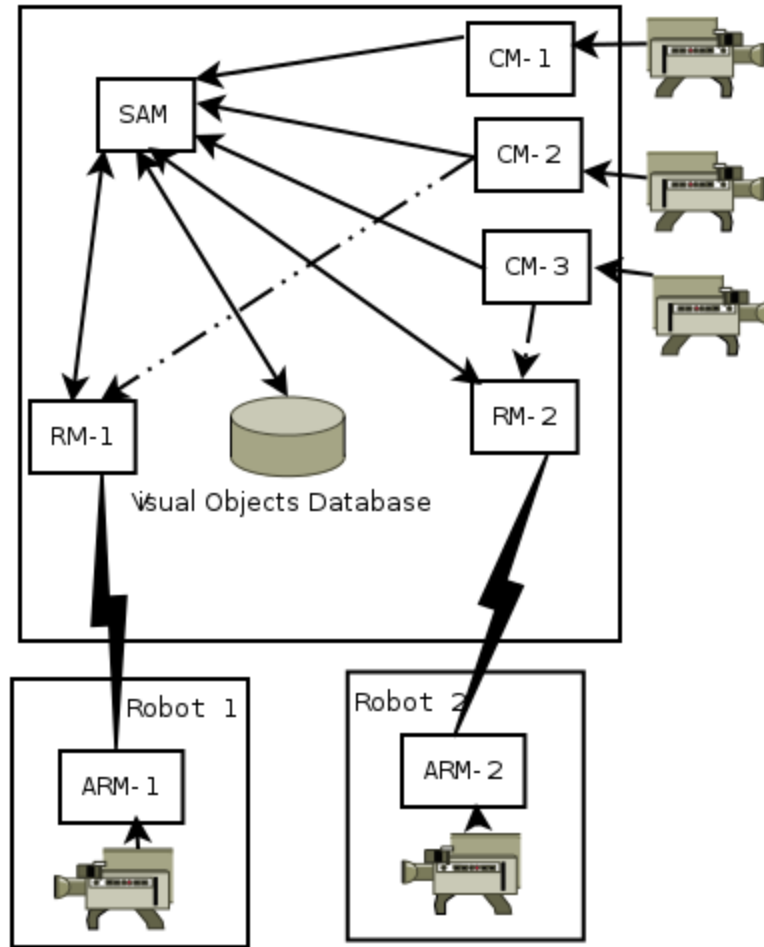


Figure 19 Main Software Modules

For each Robot a process called a Robot Module (RM) also runs on the Base Station. The RM is in constant communication with the software running on the embedded computer, on board of the robot called an Autonomous Robot Module (ARM). The RM is responsible for mission planning and doing epipolar localization if required by matching the image captured by ARM with images requested from CM. Practically, the robot's visual processing and control software is distributed between base station (RM) and mobile unit (ARM).

Our Autonomous Robot Module, runs on the “Raspberry PI 2” embedded computer board mounted on the robot. It continuously captures images from the RPI Camera and performs optical flow calculation. It uses the results of OF processing to maintain the prescribed trajectory it receives from RM and answer to RM requests for images or extended information.

During a typical navigation sequence, the RM interrogates SAM for a map of the grid between the current position and the target, and it plans a path. With the path selected, the RM commands ARM to maintain a particular trajectory. The robot navigates along the prescribed path driven by a PID controller using the direction feedback from the processing of the optical flow from the mobile camera.

RM constantly receives tracking information in the room as coordinates from SAM. They are used to periodically recalculate the robot trajectory, but this is done only when the distance from the last determined point is much larger than the uncertainty in location. As can be seen in the original paper, the initial idea was that tracking from the fixed camera can be used as real-time feedback for the robot position. The experiments we have done with this idea proved it to be unworkable because two reasons: Perspective error and blob fragmentation error. The fix we attempted was to use the robot camera OF processing for trajectory feedback and rely on the CM tracking only for rough periodic localization.

During navigation however, it's possible that obstacles unknown to SAM may be encountered. For example, a table that has never been moved from its place has been interpreted as a pattern on the floor when the grid was built. When approaching the table, the optical flow on ARM detects it as obstacle, the robot stops and relies the information

to the RM. The RM can now map the object using epipolar geometry and send the information to SAM to update the occupancy grid. Then the navigation re-starts with a new path planning.

5.3 Major Messages into the System

CMs uses HTTP to acquire images from IP cameras at the camera's maximum speed (around 6 frames / second), between modules sDOMO protocol is used to exchange information. There are two types of information exchanges employed by sDOMO: Notifications sent driven by an external event (example: a period of time passed or a moving object was detected on a previous scene) and Direct Messages used in the form of Request/Response to provide equivalent of method calls.

For every processed camera frame where at least a target is within view, a CM broadcast an *ActiveTargetsMsg* Notification containing a list of targets and basic information about them. SAM can ask with a Direct Message for extended info about a target and CM replies with an *ExtenderTargetInfoMsg*. SAM or RM can also use *RequestCMImageMsg* to request a whole or sub-image of interest and the requested is honored via the *CapturedCMImageMsg* reply.

For each *ActiveTargetsMsg* received, SAM performs a homographic transform from camera to room coordinates and the location of the targets in room coordinates are broadcast via *SAMTargetsMsg* Notification. Direct messages are used for model query and updates.

ARM broadcasts *DynamicBasicInfoMsg* received mainly by RM and contains current speed and direction of movement as commanded and detected by the robot. The

movement commands are done by the lonely direct message *DynamicBasicCmdMsg* sent by RM. This is a somehow odd message since it has no reply associated with it, but it is confirmed by an out of tempo *DynamicBasicInfoMsg notification* sent right away.

CapturedARMImageMsg is a reply ARM sends in response to RM requesting an image from robot camera and the request *RequestInformationMsg* from RM is being honored by ARM with either *RobotIdentityMsg* or *ConfigurationMsg* depending on what info has been requested.

Developer console is a GUI based applications that can subscribe to all the notifications above and can send requests for testing and debugging purposes. It can also spoof RM commands toward ARM allowing the robot to be moved by remote control by an engineer.

5.4 Environment Model

SAM is required to keep a live map of the operating environment. The map is represented as an occupancy grid representing the floor. Objects and people are modeled as prisms or as assemblies of prisms occupying a particular spot on the occupancy grid. The size of the grid cell is chosen to be about the size of the smallest object the robot is capable of manipulating. For each object besides the position and surface a radius of uncertainty is maintained which is expressed in multiple cell sizes because the objects not visited closely will have uncertainty in position that may be higher than the cell size.

We are using homography to project the images captured from fixed camera onto the room model. The homography matrix is calculated from the correspondence of well-known points in the room coordinates versus the position of them on the camera images.

Having the homography matrix calculated for each camera we can easily translate the tracking information from the fixed cameras into absolute room coordinates to broadcast them to the Robot Module.

Right now, the only information stored about the objects in the occupancy grid is their height. For future enhancements we envision a fully 2.5 D model based on surfaces. The objects will be represented as prisms of elevated surfaces from a generator plane. More precisely for each surface we store the Euler Angles (φ, θ, ψ) defining the generator plane in which the elevation grid is located, the center of the elevation grid (x, y, z) and on each cell the height from the plane to the respective surface. Besides elevation, each cell contains information about the color, texture or sub-image mapping.

6 sDOMO PROTOCOL

Created originally specifically to support this distributed architecture, sDOMO rapidly evolved during development as a general purpose protocol for home automation systems. Our vision regarding the whole intelligent home as an integrated domotic system working together to keep the inhabitants comfortable, the robot being just a mobile component of the smart house of the future. An integral part of our vision is the concept of a self-contained domotic system which is able to solve most of the problems without any reliance on the internet or vendor's sites. Most of this chapter is based on our two papers [62, 63] presenting sDOMO protocol.

6.1 Overview

sDOMO protocol proposes a hybrid communication structure with a House Hub acting both as a central registry for all the devices and also providing reliable and secure communication between devices; and optional direct device to device communication channels. Besides the home automation devices and the House Hub, the system contains a few special software components like House Intelligence Unit (HIU), Internet Gateway and Administrative Console used to control or complete the functionality of the system.

A Device is the generic name for a node in a domotic network. It can be an embedded device like a microcontroller based thermometer having Ethernet connectivity, a serial device speaking sDOMO over SLIP or PPP and connected to a serial-to-lan converter or a program running on a laptop or tablet used to display data or to manage the

sDOMO network. Bluetooth and ZigBee are also potential candidates to be considered in the future. The network associated to a particular house is called a Domain and is defined by a Domain Name. A Device is identified by the Device Unique ID (DUID) which is a 20 bytes alphanumeric string constructed based on a set of rules to insure uniqueness among vendors and/or hobbyists. Once a Device joins a Domotic Network by being accepted to a Hub, the Hub issues a Device Session ID (DID) and a Session Key is used to authenticate messages and eventually encrypt the communication between the Device and Hub. The type of devices is defined by the Spec ID. This is a string of up to 128 characters that is define as a URL from where the Spec File can be retrieved. For example, if a Device advertises a Spec ID of: *mezonix.com/sDOMO/devs/dev1* that means that a XML file defining the device can be downloaded either from:

mezonix.com/sDOMO/devs/dev1/device.spec or from
www.mezonix.com/sDOMO/devs/dev1/device.spec either by HTTP or HTTPS.

A Spec File is an XML file describing the device, linking to user and developer documentation, defining connection requirements, and the interfaces of the software objects implemented in the device. For example, an indoor video camera can specify that it transmits sensitive data therefore the Hub may decide to require encryption. What types of encryption are available for a given device, their strength, and order of preference is also specified in the Spec file. For privacy enforcement, a Spec File also contains a list of the interfaces that state this device must connect as a client in order to perform its tasks. From the software architecture point of view, a Device is composed of one of more (up to 255) Objects identified by a contiguous number between 0 to NoOfObjects. Each Object implements an Interface whose Definition File if referred by the Spec File.

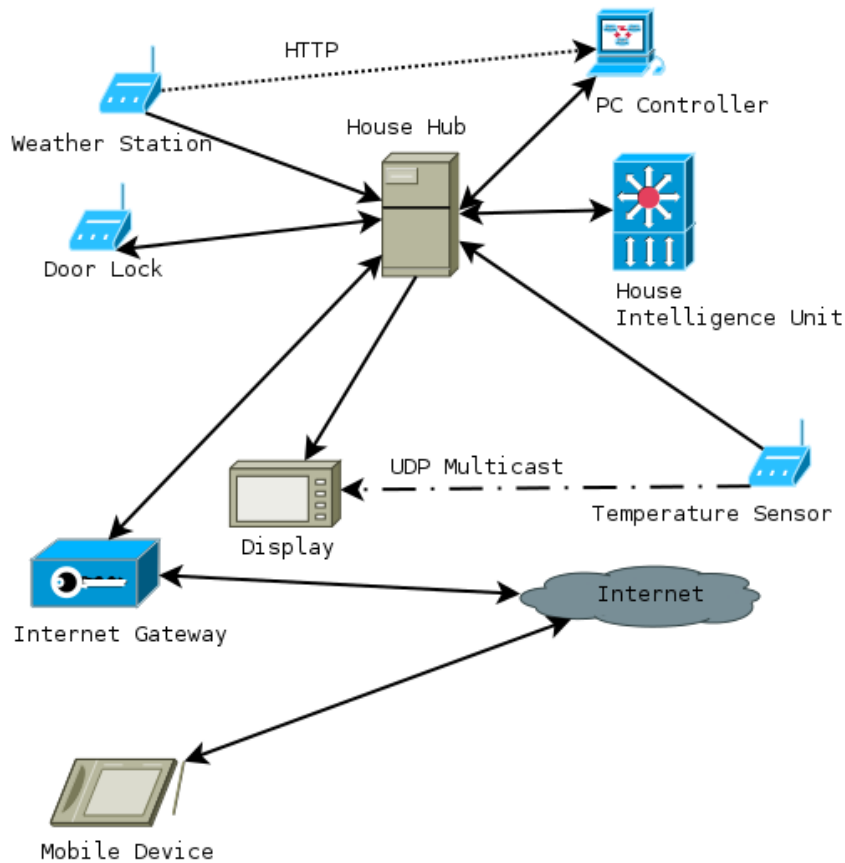


Figure 20 sDOMO - Hybrid Communication Architecture

The Interface Definition File (IDF) is another XML file defining the list of all Messages understood and/or sent by a given Object, the way they can be associated to provide method calls and the events that triggers them. The Interface Definition file is used by the developer tools to generate code for a Client that connects to an Object or for the Stub that implement the respective functionality. It is also used by the Hub for privacy enforcement and by the House Intelligence Unit for alerts and adaptive automation.

Each Device has a Device Main Key which is used by the Hub to encrypt the Session Key when it is delivered to the Device. A Hub that will accept a particular device must know the Device Main Key and its DUID, which is usually uploaded by an

administrative application scanning a QRCode or RFID tag attached to the device or typed in by the user.

Two sDOMO Devices can communicate either via Messages routed by the Hub either talking directly to each other or bypassing the Hub in what is called Streaming. Messages are blocks of data up to 4GB in size (subject to Hub/Device memory limitations) transmitted between Devices. Messages are carried as a payload in Packets. An sDOMO Packet is sent into a single UDP Datagram over the local LAN. Packets are sent between a Device and the Hub only, they are the backbone of connectivity and they can carry fragments of Messages as their payload. Messages are routed by the Hub from the source to destination Device.

There are two different types of Messages: Notifications and Direct Messages. A Notification is an sDOMO message sent by a device and received by any number of devices that were subscribed to it prior to the moment it was sent. A Direct Message is a one to one communication sent by an Object, part of a Device specifically towards another Object of a connected Device. Besides carrying Messages and acknowledgments for them there are various packets for device discovery, configuration, connection maintenance etc. The full definition of the packets structure and a C++ library for working with them is provided on the companion website for this paper.

A Virtual Device is a set of services provided by the Hub to the connected devices via Direct Messages and Notifications. In order to facilitate the process of discovery the Virtual Devices have well known DID's allocated into a reserved range from 1 to 999 in which only Virtual Devices and Devices with Special Duties are allocated. The Virtual

Device No. 1 have the object #0 designated as the DeviceManagerInterface allowing other devices to list the connected devices and their objects and get extended information about them. The AdministratorServices object allows for a console program to upload devices keys and edit their parameters. The AnonymousServices allows a device with level of trust 0 (without a recognizable Main Key) to log in as a program with special access or administrative rights using a user/password credentials pair or presenting a PKI Certificate signed with an authoritative Private Key. The House Intelligence Unit (HIU) is a special device acting as a meta-rules processor and is responsible for handling out of ordinary conditions, issuing alerts and perform complex automation tasks coordinating multiple devices based on scripts.

6.2 Data Communications

6.2.1 Overview

The preferred method of communication envisioned by our design is for devices to communicate with each other by Messages routed via the House Hub. The Hub receives packets, validates their authenticity, decrypts data payloads if necessary, assembles their payload into messages and delivers them to their destination device. While delivering messages, the Hub will enforce security and privacy rules on behalf of the devices, allowing small devices to benefit from the same level of protection as larger devices.

To support direct communications sDOMO allows for each device to advertise to other components in the network their own external/proprietary interfaces in this case sDOMO hub acting exclusively as a rendezvous server, allowing other devices to discover which peers are on the network and how to talk with them. This feature is

implemented in order to allow vendors a seemingly transition from their current protocols to sDOMO, or to support special high throughput applications like High Definition Video Streaming for which the House Hub would just introduce unnecessary delays. The protocol allows for multiple computers running Hubs for a single Domain however from the software point of view this is transparent to the devices.

The sDOMO architecture defines a house-centered self-sufficient network model. Internet connectivity is allowed exclusively via an Internet Gateway which isolates the sDOMO network from the cloud. The Internet Gateway provides encrypted communication with the outside world and require authentication of the users requiring access. The access to information is controlled via Access Control Lists indexed in the user id. In order to allow remote sensors and sDOMO Devices, the Internet Gateway provides sDOMO tunneling toward a specified IP address. Supplementary, regular web application can run in the Gateway allowing controlled access via a regular browser to specific functionality.

There is no restriction for a sDOMO device to have other interfaces incompatible with sDOMO. For example, a Power Meter can have a totally proprietary interface to be accessible by the power company while also exporting an sDOMO device in order to inform the household monitoring software about their energy consumption. The sDOMO specifications are responsible for defining a house-centrist architecture and to not interfere with proprietary interfaces of leased devices.

6.2.2 Discovery and Configuration

A Device starting up will use Multicast to the proposed sDOMO Local Multicast destination (224.68.12.8 port 1881) sending a “NewBabyBorn” Packet containing it's DUID, Spec ID and previous Domain and Given Name if any. A Hub that matches the Domain (if present) and wants that Device to join its network will send an “AdoptionOffer” Packet specifying an adoption bid between 0 and 255. The Hub that sends the highest bid will be selected by the Device to join by sending a “NetworkJoin” packet. The Hub accepting a Device replies with a “JoinAcceptedPack” which contains the Devices new DID and Session Key.

The reception of “JoinAcceptedPack” triggers the Device to send the “ConfigurationSummaryPack” containing some configuration parameters that can overwrite the default values specified in the Device Spec File. For each Object of the new device the Hub query for that object information and the Device replies with an “ObjectInfoPack”. Once all the Objects have been configured the Hub sends a Configuration Complete Packet which switches the Device to full operational mode. From this moment the device can start communicating with other devices via Notifications or Direct Messages. Having a Session Key received the device signs all outgoing packets and validates the signature of on any incoming packets.

A Device whose Device Main Key is unknown by the Hub will receive an adoption offer with a bid of 0. A regular device will ignore any bid with value of 0. However, an Administrative Console program or other GUI based device that can prompt the user to type in a username and password to join the hub at trust level 0 and use

Anonymous Services to upload its Main Key encrypted with the user's credentials. Such GUI based application are then used to scan and upload to the Hub the DUID / Key pair for any new device that will be connected to the network.

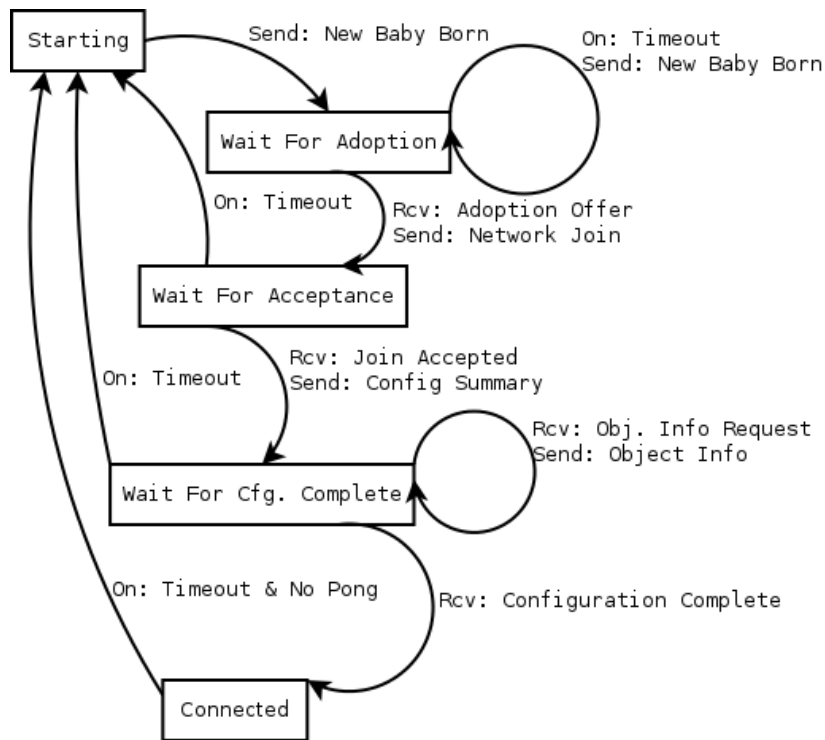


Figure 21 Device State Diagram while Connecting to the Hub

A second type of Main Key upload procedure was developed for the usage of sDOMO as a Robotic Systems Building Protocol. This is the case, for example, when an army unit is deployed on the battle field receives a new payload for one of their unmanned vehicles. Plugging in the payload will trigger a set of credential exchanges between the UAV and the new equipment, they then verify each other's identity and then the payload will upload its main key to the vehicle's hub. This procedure acts in the following way: the new device is connected with level of trust 0 and then uses the Anonymous Services to

upload a certificate for the device public key and requests the certificate of the Hub for verification. Following a successful credential validation on both sides, the device uploads a newly generated Main Key to the system encrypted with Hub's public key and signed with the Device's private key. This procedure however is applicable only for devices having non trivial computing power on board since it involves CPU intensive public key cryptography. The devices and the hub must be prepared to handle multiple levels of certificates. As presented in the hypothetical example above, the manufacturer of the vehicle may not know about the manufacturer of the payload and reversal. However, both manufacturers will be able to provide an army certificate for the key they used to sign the key of the device and vehicle respectively. This dual level of certificates allows full validation since both the vehicle and the payload knows the armies public key.

6.2.3 Packets and Messages

An sDOMO Packet is carried on a single UDP Datagram. Each packet contains a header followed by a variable payload section. The Packet ID specifies the function of this packet. DID is set to 0 for packets that have not yet assigned a Device ID like "NewBabyBorn" or "AdoptionOffer". Once a session is initiated the DID specifies the assigned Device ID. Packet numbers are unique values incremented at every sent packet, they are used to eliminate eventual duplicates on the network and to prevent reply attacks. Signature Type specify the HMAC algorithm used sign the packet. No signature=0, SHA1 HMAC=1, SHA256 HMAC=2. Encryption Type specify which encryption algorithm is used for the Packet: 0-No encryption, 1-XOR, 2-AES128, 3-AES256 are currently defined. The signature value is dependent on the signature type, for no signature the length is zero, i.e. the packet body starts immediately after the

encryption type. For SHA1 based HMAC the signature is 20 bytes long and for SHA256 it is 32 bytes. Numeric values are packet in network order i.e. big endian format.

The Packet Body is also a collection of network order fields. Strings are packed as zero terminates ASCII string, the byte 0 being required at the end of the string. Arrays are prefixed with the size encoded as variable-length code then the specified number of items follows. sDOMO does not send floating-point values over the wire but recommends to the message designers to use fixed point format represented as integers; for example, our thermostat demo sends the temperature in mili-Kelvin packed as an unsigned integer of 32 bit size which is enough to encode any temperature reachable on Earth with precision more than enough for most practical applications.

Messages are sent as a sequence of one or more Message Carrier Packets. Each packet contains the start position of the current packet in the final message. Packets arriving in expected order are not ACK-ed until the final packet arrives and then the whole message is ACK-ed with a single ACK packet. However, any packet out of expected order triggers a negative ACK containing the expected sequence. The Hub will receive a whole message and ACK upon reception to the sender before being delivered to the intended recipient. It is important to notice that therefore receiving an ACK for a whole message does not mean that the receiver got it. That means a confirmation of delivery is required, the designer of an Interface must implement it at the message level.

6.3 Security and Privacy Considerations

Home automation and robotic systems bring a whole new dimension to the problem of security. In a smart house, a potential attacker can take control and alter the physical

aspects of the environment, potentially even inflicting bodily harm to the residents. In sDOMO we oppose the view of having each device individually connected to the Internet and we propose a House Centered, Self Sufficient domotic system. Putting the whole house network behind a firewall and controlling the access from outside via the Internet Gateway is the first step in security and privacy of the residents. However, as long as the desktop and mobile operating systems that needs to access the home network are still vulnerable, the threat of an intruder infiltrating behind the firewall is a likely possibility. To guard the domotic system against this probable scenario, a two tiered packet level system and message level security measures had been designed as part of the sDOMO protocol.

6.3.1 Packet Level Security Considerations

After a new connection has been established and a Session Key has been downloaded from the Hub to the Device each sDOMO packet exchanged between a Device and the Hub is signed with a Hash based Message Authentication Code (HMAC). The HMAC is calculated with the formula:

$$HMAC(K,M) = H(K \mid H(K \mid M))$$

where H is the Hash function used, K is the Session Key and M is the message. The operator \mid specify byte stream concatenation. The Message subject to hashing consists on all the fields in the Packet Header (with the exception of the Signature itself) and all the fields on the Packet Body. In order to prevent reply attacks, when the highest “PacketNumber” of the incoming or outgoing packets is approaching the overflow limit, the Device shall re-send a Network Join packet that will renew the “SessionKey” and

reset the counters for “PacketNumber” to 0. By making sure that the combination (PacketNumber, Session Key) is unique and by not accepting Packet Numbers lower or equal than the previously received one, the system is protected against reply attacks.

To download a session key a Device is asked to generate a new 64-bit random number CR every time it sends a Network Join packet toward the Hub. The Hub will in turn generate its own 64-bit random number HR and encrypts the Session Key with a Download Key calculated as:

$$DldKey = SHA1(CR \mid Device\ Main\ Key \mid HR)$$

The HR is sent by the Hub toward the Device along with the encrypted key in the Join Accepted packet. The existence of both random numbers insure that the DldKey is unique every time a download happens and this prevents a reply attack in this stage in the connection process when a sequence number is not yet initialized. The fact that the DldKey is unique also allows usage of very small devices that have no encryption capabilities other than XOR. Since the encrypted SessionKey is a random number without any structure to reverse engineer having a size equal with the SHA1 hash, and since the Main Key is not known to the attacker the simple XOR with the unique DldKey is an approximation of one-time-pad encryption.

Streaming Interfaces as specified above are direct data connections between two Objects from different devices over the network, without the involvement of the Hub in communication process. The Hub however is involved in managing the security keys of the system. A procedure for allowing the Hub to manage and distribute the keys for streaming interface is being developed. While sDOMO allows for unsecured Streaming

Interface this is a feature that needs to be thought about very carefully. For example, if an attacker can spoof a thermometer sending data to a tablet, the worst that could happen is for the residents to get confused. However, if the same thermometer is also connected to the A/C control unit the results can have more serious implications. If a hacker sends to A/C unit a fake high temperature when the house is cold putting chilling in overdrive an eventual sleeping diabetic patient unable to properly sense the temperature may go into hypothermia. Therefore, for any control action secured communication via the Hub or encrypted Streaming Interface is highly recommended.

A unique security feature of sDOMO protocol, as far as we know, is the prevention of device kidnapping by the threat of disclosure of the attacker. If a dishonest neighbor or contractor is solicited to help with the installation of a new device, the installer will have access to the new Device Main Key in order to upload-it into the Hub. If the bad guy installs a Trojan Horse masquerading as a fake hub on the resident's laptop there is nothing a cryptographic protocol can do to prevent the fake hub to take control of the device. However, as a deterrent for this kind of situations sDOMO specifications requires that the Network Join packet, containing the address of the Hub intended to be joined, to be sent by Multicast or Broadcast if the underlying network technology permits it. It also requires for any Hub to log all the Network Join events it receives and informs the House Intelligence Unit of anything it heard on the local network about something joining the hub that is not known to be part of this domain. That is, if a Trojan Horse attempts to hijack a device, this results in immediate disclosure of the Trojan and counter measures are going to be taken by the House Intelligence Unit.

6.3.2 Message Level Security Considerations

Attacks on a domotic system are not necessarily done by outlaws penetrating the network but also by the legitimate software or devices. We are living in a world where many companies, some of them very large, generate most of their revenue from collection and selling personal information to whomever is willing to pay for it. Some of these companies have already entered the field of Domotics and Robotics. While putting a small-print note in the user agreement may give to the vendor legal rights to snoop on other devices, our approach to privacy does not agree with this practice and sDOMO is designed to take some steps toward protecting user privacy by protocol design.

The Interface Definition Files specify for each message the Security Level. Security level 0 are messages which even if sent maliciously to a device have totally benign results while the messages at level 3 must be permitted only by the authorized system administrator. Similarly, for outgoing messages, a level 0 means that any information in a message can be made public without any concern, while a level 2 or 3 respectively means that this piece of information must be kept secret and received only by authorized devices and administrative programs respectively. Similarly, any Device connected to the system has to have an associated level of trust from 0 to 3. The level of trust 0-Anonymous, is assigned to devices that connect to the Hub but which do not have a Main Key uploaded into the Hub. Beside Hub's Anonymous Services very few devices, if any, are expected to accept level 0 messages. When a device D1 sends a message to another device D2, or when D1 submits a request to subscribe to a notification emitted by D2 the message/request gets assigned a priority of:

$$MP = \min(\text{Device Trust Level}, ACL(D1, D2))$$

where $ACL(D1, D2)$ is returned from an Access Control List maintained in Hub's Database. If $MP \geq \text{Security Level}$, then the request is granted otherwise an error packet is sent back to the requester with an access denied error code. The ACL entries are assigned on a device to device basis, therefore a device wishing to snoop on the notifications sent by other devices will be denied unless the permission has been explicitly granted.

To facilitate the constructions of ACL's by regular home-owners without training in computer security, sDOMO requires that any device that needs to communicate as a client with another must list the interfaces it needs to access in the Device Spec File. The <client> tag entry must also provide the level of access required for that device, if granting this right is mandatory for the functionality of the device and a clear text description of the reason why this access is required. The Console Application used to scan a new device will download the spec file and will initiate a user friendly dialog for granting the rights. If a particular interface is not listed on the spec file, no ACL entry will exist for that particular pair and therefore the access will not be possible for anything but Security Level 0. Of course a dishonest manufacturer can list in the Spec File all the possible devices it wants to snoop on, and here comes the first line of defense: Shame. It does not require a high level security training to understand that a thermostat does not need access to all the indoor cameras and microphones to "regulate the temperature inside the house". It is hopeful that at least a small number of users take their ire to the blogosphere to make the manufacturer's life harder.

Of course, only reliance on shame alone is not good enough to protect the people privacy therefore sDOMO introduces a third XML file for Standardized Expert Advice. The Hub will maintain a list of web-sites and a database with PKI Certificates of experts publishing device reviews. These expert reviews will be presented as a signed XML files for each device that has been reviewed and will contain a set of proposed alternative ACL's each of them with an explanation of what privacy problems it solved and what functionality shortcomings it will generate. The Console Application used for installing a new device will automatically download the reviews and present the home owner with alternative configuration options based on the compromise privacy/functionality they are willing to make. The Expert Advice Files can be automatically scanned by the online stores and provide ratings of the devices before purchase. This direct feedback can have the effect of making the vendors more privacy conscious.

6.4 Additional Services

sDOMO protocol has been designed to be compact, fast and efficient to allow small footprint devices while conserving network bandwidth therefore in order to perform the complex functions required by a domotic system additional services and information are needed. To handle these requirements a set of configuration files and additional services are implemented beside the House Hub and home automation devices.

6.4.1 XML Files

The Device Spec file contains documenting features, capability specifications and default configuration values, links to interfaces exported by the device (as a server) and to interfaces imported by the device (acting as client). Documenting features allows for

alternate language specifications for Device Name, short description and link to URLs for both user and developer documentation; icons for representing the device in management GUIs etc. Capability specifications contain the list of the configuration parameters of the Device, their valid ranges of values and default values for them. For example, a sensor for the door or windows is powered by a coin cell battery will want to stay in sleep state for up to 24 hours to preserve battery life. This device needs to specify in the Spec File that it can go idle for up to 90000 seconds at a time otherwise the Hub may attempt to ping it after 10 seconds of inactivity and disconnect it after a failure to respond to 3 consecutive pings.

Interface Definition Files are referred from the Device Spec file and describes all the Messages and additional data structures used to build a Message as understood by a given software Object used in the device implementation. In addition to this, the Interface File specifies how the Direct Messages defined above are combined to provide Remote Method Calls. The Interface Definition file it is used by an Interface Compiler that parses it and generates classes used to communicate with the device (Proxy) or implement the specified service functionality (Stubs). Manufacturer proposed rules for HUI are also listed here.

A third XML file used by our domotic system is the Standardized Expert Advice File. This file is used by the specialists analyzing the devices existent on the market in order to eliminate potential privacy threats to users. Beside alternative ACL and explanations for their use the expert advice files can contain alternative default settings for the device (overriding parts the Spec file) and conditions when they are a better use than the one provided by the manufacturer. Another entry that may be added to the expert

files will be the rating of the device in accordance to various criteria to allow buyers to make more informed decisions while shopping for a new device. Scripts for House Intelligence as well as Unit and Internet Gateway can also be offered as part of the expert advice files.

All the files above are XML files which are signed using Public Key Cryptography. We are experimenting currently with “GnuPG clearsign” signatures but on a commercial system XML signature as specified by the W3C XML-Sig proposal are expected to be available too. Every sDOMO Hub is expected to maintain a database of PKI certificates along with a list of servers for expert advice and methods for querying them for a given file.

6.4.2 House Intelligence Unit and Internet Gateway

The House Intelligence Unit (HIU) is a special device having a close relationship with the Hub with which share access to the Hub Database. HIU is responsible for handling out of ordinary conditions that need special attention. This handling is achieved by a set of processing rules allowing HIU to monitor notifications emitted by various devices, and when certain conditions are met emit alerts or handle the condition by either using plugins or scripting. To aid the automated building of the HIU rules for a device, the manufacturer can add into the interface definition file potential alerts which upon a device start-up are loaded by the HIU. Besides rules from interface files, HIU can have its own set of rules added by the console from a system administrator or from the expert advice files for this device.

```

<intelligence>
  <meta_alert notification="CycleCompleted" criticality="Low" domains="House" type="text">
    <condition>(msg.Counter >= 8) and (msg.Counter % 4 == 0)</condition>
    <text lang="en">Washing machine needs your attention</text>
    <text lang="ro">Masina de spalat are nevoie de atentie dumneavoastra</text>
    <append_msg_text msg_text="Text" />
  </meta_alert>
</intelligence>

```

Figure 22 Intelligence Tag for Washing Machine Example

In the simple example from the figure above, HIU subscribes for the “CycleCompleted” notification emitter by the washing machine. Based on the condition tag, when 4 minutes passed and the notification is still being broadcasted HIU will start emitting a text alert with the first line of text: “Washing machine needs your attention” and having on the next line the content of the field Text from the notification sent by the washing machine. Since HIU alerts are most likely displayed by all GUI based devices into the house, the distracted housekeeper will be notified about the washer’s need for attention while watching her favorite show on the smart-TV.

By specifying various criticality levels and domains of actions, HIU can alert a nurse about the fall of a resident as detected by a computer vision enabled surveillance camera or send to the home-owner vacationing out of town a text message in the same time with calling the fire-fighters with a prerecorded message if the smoke detector and heat sensors simultaneously detects activity in a room. Scripts in HIU can also monitor the humidity level of the soil, query the weather channel web site for the forecast and decide whether to turn on the irrigation or not in the garden, etc. HIU is also the first line of defense against device hacking, being the one alerting the users and the company installing the system about the attempt of device hijacking as detected by sDOMO device hijacking prevention schema.

The Internet Gateway is the only point of access from the outside world into the domotic system. The house automation network contains sensitive information and can harm the residents if misused with intention or by accident. However, the home owner should be able to watch his surveillance cameras when a motion alert text message is received or a nurse should be able to take control of a robot from an independent living assisted house when an emergency condition has been detected. To allow for these contradictory requirements, the Internet Gateway will be able to tunnel sDOMO messages towards devices located outside the house network. The communication is taking place over an encrypted network connection and the tunneling connection is opened only for limited amounts of time subject to periodic re-authorization between the remote device and the gateway. The original authorization for opening the connection must be done via a separate communication channel like HTTPS or SSH and a Tunneling Session Key is issued for the other end of the tunnel (the Remote Device Service). The RDS in turn will establish an encrypted communication with the Gateway encrypting the channel with the Tunneling Session Key and authenticating with the gateway with a separate key pre-assigned to that device. Once the tunnel is opened the Device software located in the mobile device will be able to connect to the Hub like any regular sDOMO device from the local network.

6.4.3 Data Integrators and Adapters

While some microcontrollers like AVR used by Arduino Uno provide persistent EEPROM memory the microcontroller can be used to store persistent data and configuration parameters, other microcontrollers of interest for small sensors like “STM32 Value Line” do not have this capability. To allow these kinds of devices to have

persistent configurations, a new type of packet, “ConfigurationParamsPack” has been added to the protocol. This packet is sent by the Hub to the device immediately after connection during the configuration phase and is also sent by the device toward the Hub when it needs to update a particular persistent parameter. The Hub stores the configuration parameters for each device in the Hub Database allowing either the Administrative Console or the HIU to modify them. If an allowed program modifies a parameter, the HUB will send that immediately to the device which can be then reconfigured on-the-flight if the device firmware implements that capability.

Unless some other popular protocols in embedded world, originally sDOMO did not provide a Topic-Based method for integration of messages across devices, under the assumption that HIU can handle any type of choreographing between multiple devices. While this is still true, driven mainly by DDS, the Topic-Based data model is gaining traction especially with people having programming experience in the defense industry. To avoid overloading HIU with these new capabilities, we decided to implement a separate service to provide this functionality: The Topic Manager Server. This server software runs on the Base Station and creates a set of Virtual (software only) Devices that will subscribe to data streams from real devices. The virtual devices provide data integration driven by configuration files or dynamic requests and expose the same interface as a regular sDOMO device. That way, Topic-Based integrated data crosses multiple devices can be accessed using the regular sDOMO API without any modification to the protocol or devices.

Very similar to the Topic Server, an Adapter Server is being proposed with the only differences being that it will collect data from devices that speak other protocols not

sDOMO. Originally we proposed that manufacturers will add sDOMO protocol as an optional alternative to their current protocol allowing a device to speak sDOMO if it's in an sDOMO network otherwise it uses the manufacturer protocol. However, we found this idea to be difficult to be accepted, since for the manufacturers to have financial incentives to do this work there must be a relative large base of sDOMO networks. But in the same time, one cannot build sDOMO networks without having sDOMO devices. The Adapter server is the tool that we hope to provide the resolution to this dilemma. The Adapter Server allows construction of sDOMO networks without having any native speaking sDOMO device, while at the same time it will offer all the benefits that our protocol has to offer. A device speaking a different protocol is going to talk exclusively with the Adapter in its natural language while the adapter presents itself as an sDOMO device offering the same functionality as the original device. The Adapter Server is a container managing a set of plug-ins (drivers) that implements the specific protocol of each device to be integrated.

7 MULTI-THREADED MESSAGE DISPATCHER

To implement House Hub and House Intelligence Unit from sDOMO system presented in our previous paper [62] we designed a Multi-Threaded Message Dispatcher (MTMDispatcher) framework to support Messages and sDOMO Packet processing. The framework has been designed to be reusable for other applications that require message processing and is being offered as open-source.

The core idea behind the design of this framework has been the suitability for Safety and Mission Critical Application development therefore attempting to aid with a few of the challenges encountered during application development: Deadlock Prevention, Separation Of Concerns and Software Testability.

7.1 Problem

The typical problem this framework addresses is the problem of multiple devices sending data in messages toward a central message processor (MP) which has to process the information and eventually send messages toward the devices in response. In the figure below the main architecture is presented.

The Devices are software components able to send messages, the system accepts multiple devices of the same kind as well as different kind of devices. A “kind” of device is characterized from the types of messages that it emits and receives. Some devices can be just simple software components either one way like a logger or two ways like a database or other type of data store. Since devices send messages asynchronous from one another and some messages can be just re-routed to other device with little or no processing, it makes sense for the message router and processor to operate using multi-

threading in order to make best use of CPU cores and achieve higher throughput.

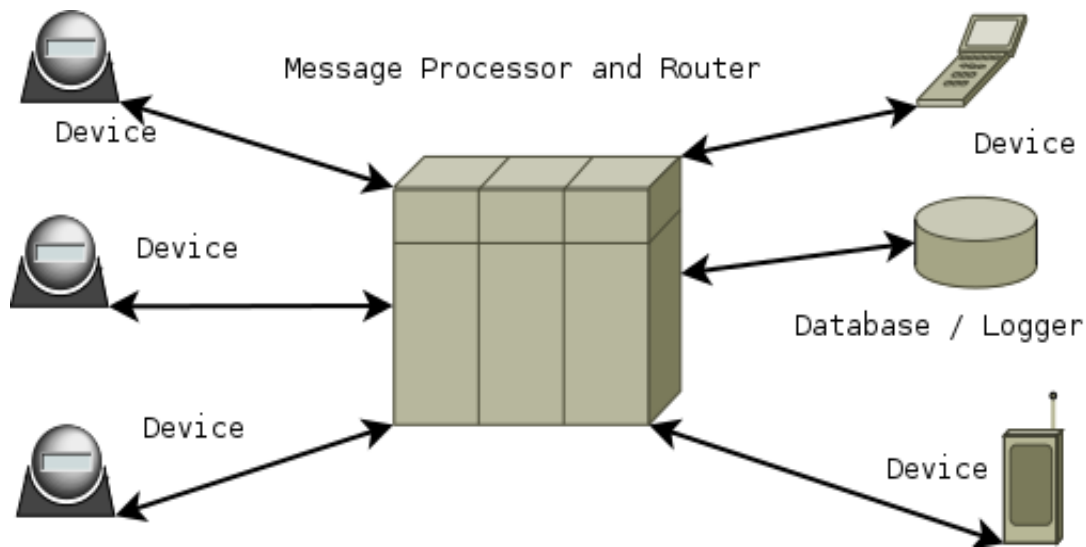


Figure 23 Multiple Message Sources and Shared Data Talking to One Message Processor

Because the message processing may require access to shared data, mutual exclusion has to be implemented in order to avoid race-conditions and whenever multiple threads and mutexes are employed there always exists the possibility of deadlock.

Having the software engineers constantly switch their attention between the business logic and the implementation details (like for example the deadlock-prevention or validating pointers) opens opportunities for mistakes to slip in. The principle of Separation of Concerns recommends to clearly separate the two, allowing the programmer to focus on and address a single class of concerns at a time.

Despite most design and implementation efforts, errors are highly likely to slip in and software testing it is the most common way used to detect them in order to be eliminated. Unit testing emerged as a very good testing strategy allowing small units of the program to be independently put into a test harness and exercised independently into

a controlled way. Unfortunately, unit testing is neither easy nor cheap when the program has not been designed with unit testing in mind, because usually each unit makes references to other units and this increase in cascade making good tests harnesses notorious hard to write.

7.2 Proposed Solution

As depicted in the sketch from Fig. 24, the main entities of the system are a set of message sources S_1, S_n which asynchronously add Messages into the Priority Queue. The Message Dispatcher owns one or more threads which extract the next Message (in the order of priority) from the queue. Upon successfully validating a Message the Dispatcher looks-up all the Message Handling Entries registered for this particular message. A Message Handling Entry consists of a Message Handler Function (Handler 1() ... Handler m ()) and a tuple of references to Shared Data Objects ($D_1 \dots D_p$).

For each Entry, the Dispatcher will lock the mutex associated with each Data Object using the “partial ordering deadlock avoidance algorithm” as proposed by Dijkstra as solution to “Dinning Philosophers Problem”. Once all the resources are acquired, the Dispatcher calls the function handler passing a reference to Data Objects as parameters to the function.

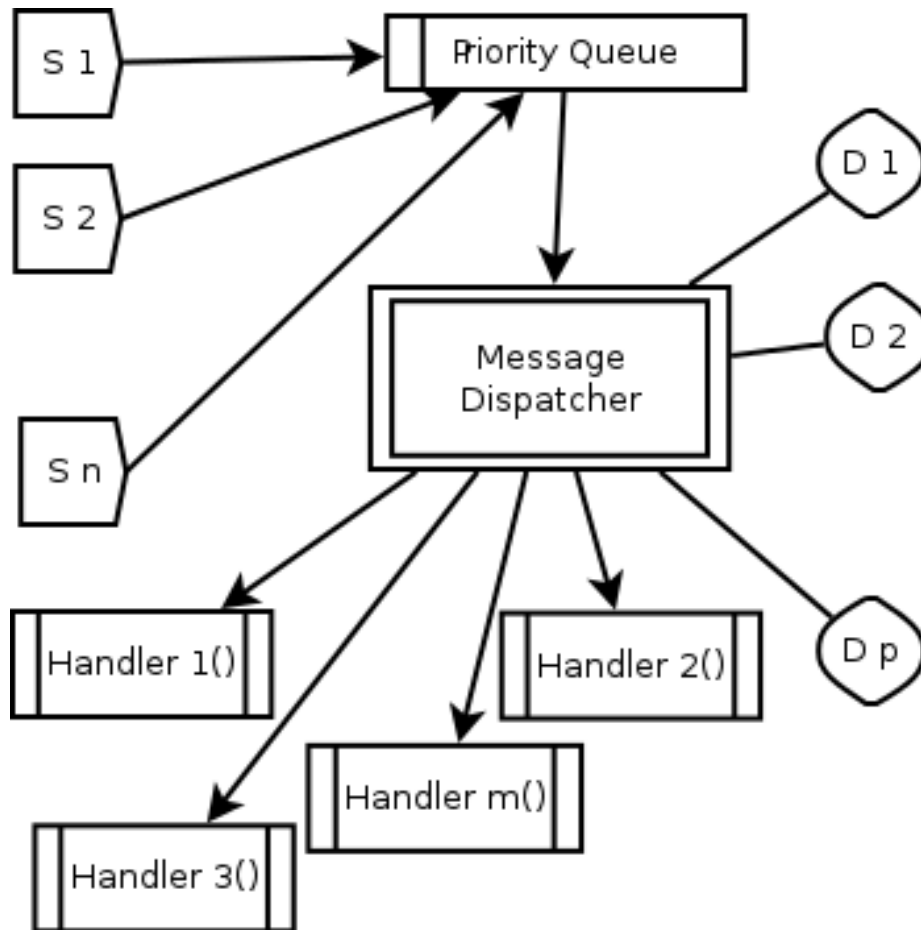


Figure 24 Dispatcher Main Components

7.3 Mission Critical Application Support

The proposed design has a set of features to provide support for development of applications that are vital for an organization or system.

7.3.1 Dealing with Race Conditions and Deadlock Prevention

The main goal in designing this framework has been the ability to allow multithreaded message processing while making sure the access to shared resources would never result into a deadlock situation that will make the system unresponsive and unable to perform

its mission critical role. The framework implements a deadlock avoidance procedure that guarantees a deadlock-free dispatching how long the accesses to Data Objects are non-blocking, i.e. implementing request/completion asynchronous operations. The algorithm for deadlock avoidance works as following:

1. All Data Objects are made inaccessible from regular user code using DataProtector template class, this makes race condition impossible since any attempt to access a Data Object outside of the framework control results in a compiler error
2. For each Message that needs to be handled a function handler must be declared and registered with the Dispatcher
3. Handler registration specify for each Handling Function the set of Data Objects that should be bound to it parameters during a call
4. When a Message is handled, the Dispatcher will lock the Data Objects in the order of their unique locking priority, avoiding the possibility of deadlock by the partial ordering solution
5. The references to Data Objects are retrieved by the ExecCaller object created by Dispatcher which have Friend Relationship with DataProtectors and passed to the Handler Function as parameters.

7.3.2 Support for Separation of Concerns

Separation of Concerns is a design principle in software engineering that asserts the need to minimize the amount of time the mind of the programmer performs context switches

like for example between high-level business logic and low-level implementation details.

It is believed from the psychology studies that these constant context switches are a weak link in the process of focus, allowing programming errors to slip in. The presented framework design attempts to aid the programmer in the task by taking a small set of tasks on its own and enforcing others.

The fact that messages are sequenced in a priority queue guarantees that lower priority processing will not delay critical messages from being handled. Once the software engineer determined the priority of each message, either role based or by RMS, the framework will take care of handling the proper task with the appropriate priority without further programmer's attention.

Instantiating each of the Data Objects under the control of a DataProtector prevents the programmer from accessing them directly enforcing them to rely on the framework in order to access each Data Object. This eliminates the need for the programmer to care about Critical Section problem outsourcing it to the framework. As a matter of fact since all the handlers registered for a particular message are called sequentially under the context of a single thread this also eliminates the need for the programmer to write Message Handler Functions to care about multithreading entirely, from the programmers point of view there is no difference between the fact that a particular handling function is called from the Multithreaded Dispatcher or just called from a regular function into a mono-threaded program. All the synchronization and deadlock avoidance procedures are hidden inside the framework.

7.3.3 Support for Unit Testing

Having all shared Data Objects constructed under a protector, forces the programmer to declare the required shared objects as parameters to message the handler function in order to be provided by the framework. As a result, all message handler's functions are self-sufficient pieces of code that can be tested individually in a test harness that just passes the required parameters to the handler subject to testing. Because the framework also takes care of all the multithreading and synchronization issues hiding this aspect from the handler programmer, all the message handler's unit tests can be performed into a single-threaded easy to use environment.

7.4 Design Details

Two interfaces serve as the base for the Data-Protectors. LockableObjectInterface is the base for any class that the MTMDispatcher class is supposed to lock it before calling the handler and release it after. DataProtectorInterface, it is a template abstract class parameterized with a data type that will be passed to the message handler function. The DataProtectorInterface have two protected member functions returning pointers to a LockableObjectInterface and the data type used to instantiate the template.

An auxiliary template interface SelectiveDataProtectorInterface serves as the base class for registering arrays of shared data-objects in order to pass to the handler one of them based on some information from the incoming message that is being processed.

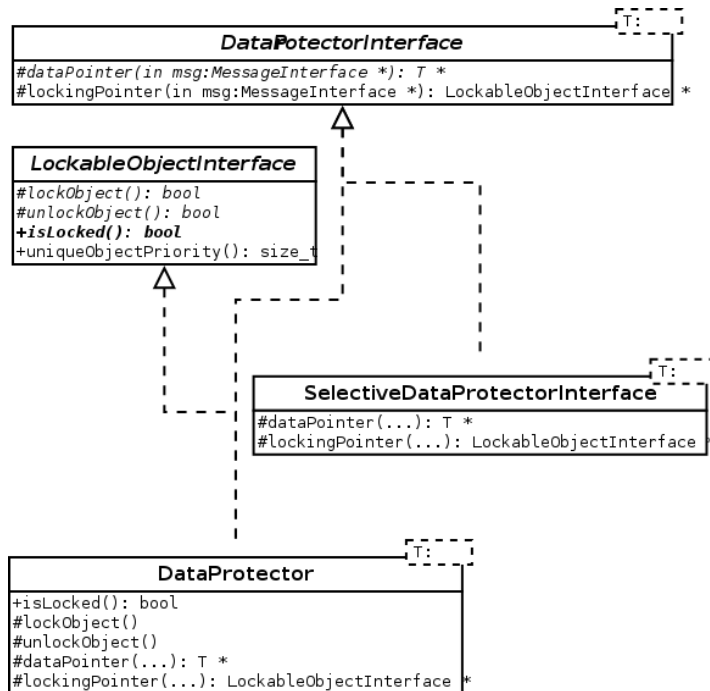


Figure 25 Class Diagram related to Safe Data Access

When a handler function is being registered, the references of the classes extending DataProtector template class or SelectiveDataProtectorInterface are being passed to the registration procedure. The framework uses a friend relationship with the protectors in order to access the methods that provide a pointer to data or associated locking mechanism.

The abstract class MsgSourceInterface is the base for all the objects that will send messages to be handled by function handlers. A message is a class inheriting an instantiation of template Message with two integer parameters, CategoryID and MessageID, and then defining their own data. Message Sources have the ability to enqueue into the Dispatcher MsgHandle which references an actual Message that needs to be sent. When the Dispatcher dequeues a message reference, it uses it to get access to

the actual Message via the method getMessage() from the MsgSourceInterface. This twostep access (using a handle that resolves to message instead enqueueing a direct pointer to the message) have been implemented to address two important problems: event cancellation and messages instantiated in special memory segments.

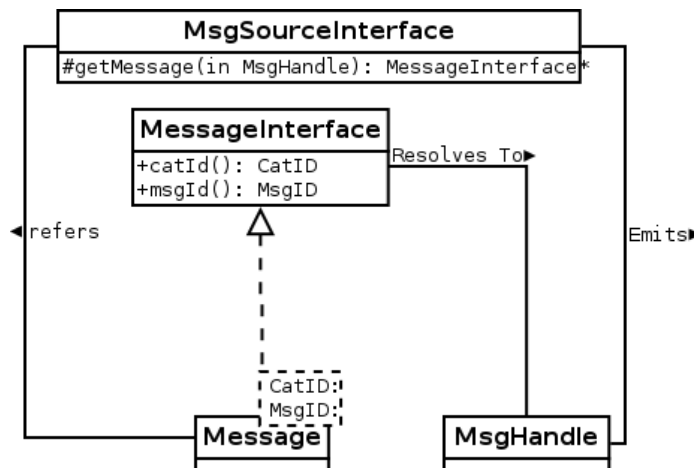


Figure 26 Class Diagram for Message Management

Event Cancellation is best understood considering a time-out timer started when the request is lunched and which calls a time-out function if the answer has not yet received in time. If the response is received the reception handler will cancel the timer. However, if the queue is not empty when the reply is received, the message is enqueued at the end of the queue and until it will be served it is possible that the time-out event will be also enqueued to be executed later. Without the two step look-up both reception and time-out handlers must perform extra accounting steps to keep track of a particular request/time-out pair since just canceling the event have no effect the event being already enqueued as a message. With a two-step look-up, when the answer handler is processed it cancels the timer and when the t/o event reaches the execution state the source will just return a null

message avoiding the time-out handler from taking place, so event cancellation is achieved without any external coding from the point of view of the application programmer, in direct accordance with the Separation of Concerns principle.

Dual step look-up also allows large messages to be kept in a memory managed by the Message Source itself which can, for example, manage blocks of data in shared or non-uniform memory blocks. When the look-up of the handle is performed the right block can be mapped into the process address space and a pointer being returned. Enqueueing directly a pointer to the memory would require the memory to be mapped early and stay idle for all the period the pointer it is enqueued or would require data copy into process memory.

With every call to the template method registerHandler of the MTM_Dispatcher a new DeferredCaller entry it is added to the map indexed on the pair CatID,MsgID. The DeferredCaller entries holds the pointer to the function handler to be called and references to the data protectors associated with the data that needs to be passed to the function.

The Dispatcher starts one or more dispatching threads. Each thread will dequeue a MsgHandle and from the owning message source a pointer to the actual message is retrieved. If the resolved pointer is not null, each DeferredCaller entry associated with this message is called with the message. Beside the Message pointer, a pointer to the DispatcherLocker object owned by every thread is passed along to the call(...) method of the DeferredCaller. The DeferredCaller uses the DispatcherLocker and the Functor it holds to instantiate on the stack an ExecCaller functional object. The ExecCaller

performs object locking in accordance to partial ordering solution and then call the actual function handler with the values retrieved from the protectors.

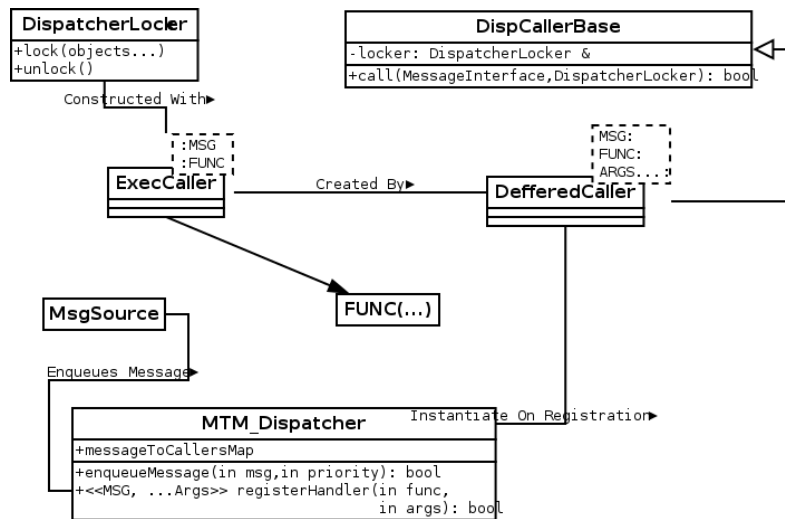


Figure 27 Class Diagram for Message Dispatching

The rationale for having the intermediary **ExecCaller** instantiated on the stack instead of allowing the **DeferredCaller** itself to perform locking is to assure that the same **DeferredCaller** can be simultaneously called from two or more threads. Calling the functor with the expansion of the Tuples as a variable list of arguments while holding a lock on all the arguments requires that the object performing the call do locking before call and hold a pointer to the Locker to perform unlocking after the function handler returns. Keeping the pointer in the **DeferredCaller** would make impossible for it to participate simultaneously in different threads since each thread have a different **DispatcherLocker**. Creating the intermediate object **ExecCaller** on the stack solves the problem in an elegant manner.

After all the handlers have been successfully called, the dispatching thread

releases the Message data with the source and waits on the MsgHandler queue for the next message.

7.5 Typical Usage

Using the MTMDispatcher framework to implement a Message Handling Application consists into a set of standardized steps:

1. Defining the Messages that are being processed by the application. The Framework defined messages as parametric templates with two integer parameters, named as CategoryID and MessageID allowing flexibility in mapping the messages ID coming from various device types
2. For each Message defines the Shared Data Structures required to process it. The Shared Data usually is a struct element grouping together various pieces of data that make semantic sense to be associated from the point of view of the business logic
3. Write function handlers for each message, having as first parameter a reference to the Message class and followed by references to all shared data objects that need to be accessed by the function handler
4. Instantiate Data Objects inside a protector as Protected Data Object variables
5. Write Message Sources Servers as active objects inheriting MsgSourceInterface
6. Instantiate Message Sources
7. Register the handlers and the corresponding protected data objects with the Dispatcher

8. Call the method start() of the Dispatcher
9. Call the start() method for all the Message Sources

There is a possibility for a Data Object to extend the MsgSourceInterface allowing Messages to be posted when certain conditions are met, as a matter of facts most of the Data Objects would probably be implemented this way allowing a three step process that brings the Separation of Concerns as “first class citizen”. More precisely, message Handlers functions can be divided in three categories: Incoming Handlers, Business Logic Handlers, and Outgoing Handlers.

When an incoming message comes from an external source, the set of Incoming Handlers will just receive the data and unpack-it into the appropriate data objects. As a result of changing the state of data objects, they emit various business logic messages like posting an alert condition, requesting an adjustment to another value, etc. These messages are handled by the BL set of Handlers which implement domain specific knowledge to assess and react to BL events. Either as a result of processing BL or by timers, a set of Outgoing Request Messages are emitted which are used by the Outgoing Handlers to pack and send the data to external devices. The clear separation between the operations of Handling External Data and Business Logic processing allows different team members to focus on their specific tasks reducing the cross-domain coupling.

8 MULTI-PARADIGM OBJECT TRACKER

This chapter presents the MP-Tracker algorithm developed to provide real-time tracking data about the objects moving in the view of a fixed camera. The algorithm has been presented in the paper [61] presented at ACM RACS in 2014.

8.1 MP-Tracker Introduction

An important component of the software architecture for domestic robot navigation presented in our previous paper [1] is the external vision (exovision) based Object Tracker located on the Camera Module (CM) on the Base Station. The Camera Module is a piece of software associated with each fixed camera overlooking the scene being responsible for image pre-processing and tracking of moving objects. At every frame the CM broadcasts the status of each tracked object in a Tracking Message, to the Situation Awareness Module (SAM).

The Tracking Message contains a list of tracked objects (referred to from now on as the Targets) with the specified location, movement vector, and the size of the bounding rectangle around the tracked object. Upon request from other components CM will provide a variety of information including a polygonal approximation of the Target, its Fuzzy Histogram, or even partial or full images. SAM integrates information received from multiple CM and builds a 3D model of the environment in the form of a live-map. The model built by SAM is used to provide navigation and localization assistance to the robot.

The role of the Object Tracker as a part of the CM is to provide real-time tracking information of the objects “seen” by each fixed camera to aid the robot in determining its

own position and to avoid collisions with other moving objects, persons, or pets.

The main problem with the tracking operations is Data Association, i.e. if at the moment \mathbf{t} we are having a set of N targets $T_i^t, 0 \leq i < N$ and at moment $\mathbf{t} + d\mathbf{t}$ we detect M blobs $B_j^{t+dt}, 0 \leq j < M$ the problem consists of determining a set of P pairs $P \leq \min(N, M)$ such that a pair (T_i^t, B_j^{t+dt}) have the meaning that the target \mathbf{i} at \mathbf{t} moved at the position where the blob \mathbf{j} has been detected at the moment $\mathbf{t} + d\mathbf{t}$. Once an association has been made the position and other information about the target are updated with the new data acquired from the new detection and then we can talk about the target \mathbf{i} at the moment $\mathbf{t} + d\mathbf{t}$. A Track is defined as a sequence of consecutive positions occupied by a target during its life time, i.e. the ordered sequence $\tau_i = (T_i^{t0}, T_i^{t1}, \dots, T_i^{tk})$ with $\mathbf{t0}$ and $\mathbf{tk} = t0 + k \cdot dt$ respectively being the moment when a target has been taken into account (target initialized) and removed from the tracking accounting (target destroyed).

8.2 Method

We took a multistage approach to the problem by generating a set of “one to many” hypothesis of association between Targets and Blobs and later refining them in subsequent steps using various image processing and data association methods.

The main program flow is given in Fig.2. After the image capture background subtraction is performed using the OpenCV’s class *BackgroundSubtractorMOG* which uses a Gaussian Mixture Model of the background and the result of the subtraction is a binary image representing the modified foreground as a set of blobs. The contours of blobs are found and an array of data structures is built holding for each blob the position, area, bounding rectangle and an RGB space Fuzzy Histogram. A data structure part of

this array will be referred from now on as a Blob. The histogram is built from the RGB values of the pixels inside the found contours providing an easy way to compare the visual aspect of two blobs.

A second array of data structures called Targets is used to keep information about detected objects. Each Target contains a Kalman filter user for tracking the movement, a list of associated blobs and trajectory history. On every frame, each Target uses the Kalman filter to predict the expected position into the new frame then it lays a claim on the newly detected blobs that may be located around the expected position. A claim, named in code a Hypothesis, associates one Target to one or many Blobs and a confidence score.

Based on the prediction from the Kalman filter on each step an Expected Bounding Rectangle (EBR) of the target in the next frame is calculated and all the blobs intersecting the Expected Bounding Rectangle are selected to be part of the hypothesis for this Target.

Because with each Kalman filter we have an uncertainty in location, a blob can intersect more than one EBR resulting in an ambiguous Hypothesis. The next steps are dedicated to refine the confidence and provide hypothesis disambiguation.

On the confidence refining step we attempt to eliminate the claimed Blobs by a Target that are not grouped together close enough to form a single object. For some Hypothesis that will also result in disambiguation, however the next steps are purposely performed to eliminate any remaining ambiguities.

```

While (true):
    capture Frame
    Background Subtraction
    extract Contours, build Blob Array
    for each (existing Target):
        predict next position and uncertainty using Kalman filter
        create a Data Association Hypotheses
    Attempt to disambiguate Hypothesis array using:
        Fuzzy Histogram Matching
        Area Matching
        Lucas-Kanade Optical Flow Matching when needed
        MSER Segmentation to break ambiguous Blobs
    Pick unambiguous hypothesis having confidence > threshold1
    Pick ambiguous hypothesis having confidence > threshold2 while no contender exists with confidence > threshold3
    Run Second Chance Tracking Algorithm
    Attempt Target Splicing
    form pairs from Leftover Blobs in current & last frame
    assign score to each pair based on:
        Fuzzy Histogram Matching
        Area Matching
        Lucas-Kanade Optical Flow Matching
    Create new Targets from pairs with score > threshold1
    save all unmatched Blobs for next frame Leftovers
    for (all unmatched Targets):
        increase the uncertainty rectangle & increment age
        if (trajectory took them out of frame or spliced) delete it
        if (age > age_threshold) delete it
    for(all matched Targets):
        set age to 0
        update Kalman filter position and uncertainty equation

```

Figure 28 MPTracker Algorithm

The first attempt of disambiguation employs Fuzzy Histograms. For each blob

claimed by more than one target a score is calculated comparing the histogram of the ambiguous blob with all the blobs previously associated with a target in the previous frame. The score from histogram matching is combined with a score calculated from the matching of the area of the blobs in question using a weighted sum. The weight attributed to the area score is much smaller than that of the Fuzzy Histogram score because the area can vary due to occlusions much steeper than other visual characteristics.

For all the ambiguous claims that fell below the confidence threshold and were not picked, Lucas-Kanade's optical flow tracking is used to update the confidence in the claims, and a second run of the matching algorithm is used to pick Target with Blobs pairs based on the newly updated confidence.

Finally, the last attempt at Hypothesis disambiguation employs the Maximally Stable Extremal Regions (MSER) operator. A sub-image surrounding each Blob that was ambiguously claimed up to this point is segmented using MSER algorithm and an AND operation is applied between the segmented regions, and the original binary image of the ambiguous Blob. The resulting set of contours are used to update the old Blob and the rest are added as new Blobs, then the Hypothesis are recreated from scratch and updated using the steps described above with the exception of Lucas-Kanade matching. Since the resulting new blobs will all be within the boundaries of the old Blob there is nothing new LK can find and therefore its usage at this point will be a gratuitous waste of CPU power. After the hypothesis updates the data association procedure is run again to pick new association Hypothesis.

The Second Chance Algorithm investigates the possibility that the association

between Target and Blobs has not been possible up to this point, because multiple blobs are too close together both in position and color to be distinguished even after MSER segmentation, or because Kalman prediction failed to estimate their position. Raw Lucas-Kanade matching is run on bounding rectangles of the last known position of the Target and an estimated Searching Rectangle whose calculation is described later in the paper.

The Target Splicing Algorithm investigates the possibility that unassociated Targets might have just been fragments of a bigger Target that was tracked individually because of partial occlusions and now their blob merged. Target Breaking Algorithm looks at the targets whose new bounding rectangle grew much faster than the sum of areas of the associated blobs. It attempts to find targets that were mistakenly associated with blobs moving in different directions than the rest of the Target and remove them from it.

All the unmatched Blobs in the current frame up to this point are compared against all unmatched blobs from the previous frame using both Fuzzy Histogram and Lucas-Kanade Sparse Optical Flow and the good pairs are used to initialize new Targets. Any blob that has not been matched up to this point becomes part of the list used to initialize targets in the next frame. While this is similar to MHT method, it is important to notice that we never keep more than a single frame of Leftover Blobs for matching therefore avoiding the exponential growth of hypothesis specific to MHT.

8.3 Target Modeling

Each Target contains a Kalman Filter used to model and predict the movement of the targets in time.

The Kalman state equation:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ dx_{t+1} \\ dy_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_t \\ y_t \\ dx_t \\ dy_t \end{bmatrix}$$

is the standard equation for a body in motion in 2D space without a control signal. The Kalman filter is used to a predict / update cycle: On every frame the filter predicts where the new position of the object should be and if the center of a blob is found within an expected rectangle around the predicted position a level confidence is associated with this prediction and the filter is updated with the new measurement. The expected rectangle center is provided by the Kalman filter while the size of it is provided by a confidence equation and previously detected bounding rectangles of all the Blobs that are part of the Target. At every update step, the distance between the predicted center and the real center is calculated and the uncertainty in position is updated using the exponential averaging

$$\begin{pmatrix} x_{sz} \\ y_{sz} \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix} \cdot \begin{pmatrix} x_{sz} \\ y_{sz} \end{pmatrix} + K \cdot \begin{pmatrix} 1 - \alpha & 0 \\ 0 & 1 - \alpha \end{pmatrix} \cdot \begin{pmatrix} dx \\ dy \end{pmatrix}$$

where the vector [dx dy] is the distance between the predicted and measured center, while the resulting “sz” vector is the size of the Uncertainty Rectangle.

The Uncertainty Rectangle expresses the uncertainty of the location of the predicted center of the Target in the next frame and is used to calculate the Expected Bounding Rectangle (EBR) of the next predicted position. More precisely if the Target currently has a bounding rectangle TBR, EBR is going to be the rectangle having:

$$\begin{aligned} center &= (x_{t+1}, y_{t+1}) \\ size &= (x_{sz} + width_{TBR}, y_{sz} + height_{TBR}) \end{aligned}$$

Beside the position and location uncertainty, a Target object contains the last measured area, bounding rectangle and a fuzzy histogram of all the blobs associated with the Target in the last frame. Both of them are used to calculate the confidence in a match. Given the A_t and A_b the area of the Target and all the considered Blobs respectively the confidence in matching by area is calculated as:

$$c_A = \frac{\min(A_t, A_b)}{\max(A_t, A_b)}$$

The fact that each Target can be associated with more than one Blob allows us to handle Target fragmentation that is occurring when a Target passes in front of background spots having similar color and texture as the Target or when it is partially occluded by small objects in front of it. Since a Target needs to have a well-defined center in order to be used in Kalman filter calculation and determination of EBR, the center of a multi-blob Target is calculated as:

$$\begin{aligned} A &= \sum_{i=0}^{N-1} a_i \\ x_T &= \frac{1}{A} \cdot \sum_{i=0}^{N-1} x_i \cdot a_i \\ y_T &= \frac{1}{A} \cdot \sum_{i=0}^{N-1} y_i \cdot a_i \end{aligned}$$

where x_i , y_i and a_i are the coordinates and respective the area of a blob composing the Target.

Another type of problem arises when two targets come so close together that their blobs get generated by background subtractions forming a single bigger blob. If the conjoined blob is part of the two already initialized targets, one of them is going to lose tracking then an increase about its uncertainty rectangle will follow. When the Targets separate the matching algorithms will find the blob within its (very large now) EBR and

usually Lucas-Kanade and other matching functions will be able to correctly assign the blob back to the right Target.

A much more difficult problem arises when a new object with a color close to an existing Target but is a size bigger than it enters the Target's EBR. The new object fails to initialize as a new target and will be adopted by the existing one, resulting in false tracking. The only partial solution we have at this moment to this problem is when the new blob is much bigger than the existing one. We can avoid some false tracking with much bigger objects by imposing a restriction that we will not disambiguate a hypothesis that makes a given blob grow 2 times or more unless the two subsequent blobs overlap. However, an object that is just marginally bigger than the existing Target and that has a relatively close color can still generate false tracking. This remains an open problem as of this moment. If the new object is smaller than the target, the algorithm will stay fixed with the current Target when they split and the problem does not arise at all.

8.4 Target Life Management

New Targets are created from the "LeftOver Blobs" i.e. from the Blobs that were not assigned to any existing Target by the end of the frame processing. At the end of each frame, all left-over Blobs are paired with all the leftover Blobs from the previous frame and then the pairs are refined consecutively by Fuzzy Histogram matching, Area matching and Lucas-Kanade matching. The best matching pairs over a certain threshold are selected to initialize new Targets.

Each Target keeps a frame number with the value of the last frame when a position updates i.e. a successful matching has taken place. Using the saved frame

number and the number of the current frame each Target has a calculated Age which is defined as the number of frames passed from the last successful update. A Target that failed to be updated for one or more frames is called a Lost Target. The age of a Target is directly correlated with the uncertainty in position. While after a successful update the size of the uncertainty rectangle is calculated as described above, for a lost target the size of uncertainty rectangle is each frame until its size equals frame size.

When a Lost Target ages over a certain limit, the Target is eliminated from the array of active Targets, this is called a Target being destroyed. If the same object is detected later, it will be reinitialized as a new Target and unfortunately all the previous trajectory information will be lost. A Target is also destroyed when it is Lost and the projected Trajectory is determined to be out of the frame. In that case we do not have to wait for the maximum age before Target elimination.

When a Target is Lost while its EBR is intersecting another Target's EBR the Target is marked as Occluded. Occluded Targets are permitted to reach an older age before they are removed from the system. Active research is currently done to an algorithm to detect occlusion with fixed objects (non-Targets) from the environment that are located between the Targets and the camera.

Finally, a target that is declared to be Spliced into another by the Breaking and Splicing algorithm is also destroyed immediately to avoid generating false hypothesis.

8.5 Fuzzy Histogram

Fuzzy Histograms (FH) are used as a fast method to increase confidence that a particular blob located within the expected rectangle predicted by the Kalman Filter is the

tracked object. For each detected Blob a Fuzzy Histogram is calculated automatically when a Blob object is created from a detected contour. Whenever a Blob is assigned to a Target the Blob's main data including the Fuzzy Histogram is carried inside the Target. The disambiguation algorithm for a Blob claimed by two or more Targets first makes use of the FH for a quick comparison. FH is also used as the fast way to filter away candidate pairs used for Target initiation.

Unless other work done with FH we are not converting the RGB color space to HSV space but we are using a three dimensional histogram for each color component of RGB space while also using a much larger interval between bars. That is, we are using only 4 to 6 bars for each color component and we normalize the values of the histogram in the interval $[0, 1]$ to make it independent of the number of pixels. While this approach is more sensitive to changes in brightness it is also more robust in matching variation in color and is faster.

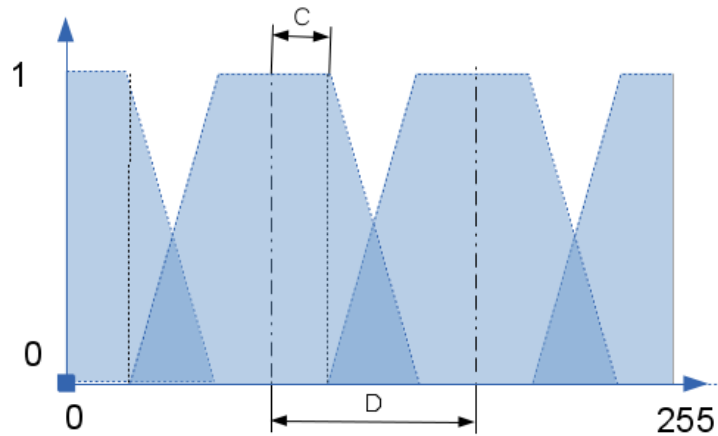


Figure 29 Fuzzy Histogram Membership Function

For updating FH with pixel values we are using a trapezoidal membership function allowing for small variation around the main histogram bars, as shown in Fig. 4.

If the pixel values are falling in the collar **C** vicinity of the histogram bar, then only the given bar is updated otherwise both bars bounding the pixel will be updated proportionally with the distance from the pixel to the neighboring bar collars.

Comparison of two histograms **A** and **B** is done by returning a matching score calculated according to the formula:

$$S = 1 - \frac{\sum_{i=0}^{N-1} |b^A_i - b^B_i| + |g^A_i - g^B_i| + |r^A_i - r^B_i|}{6}$$

where r,g,b are the normalized values for Red, Green and Blue respectively and **N** is the number of bars in the histogram.

8.6 Lucas-Kanade Tracking

The LK method provided by OpenCV library implements a sparse iterative version of the Lucas-Kanade optical flow with pyramids. This function is used to find a set of matching features (corners) in two consecutive images in order to increase the confidence that the object located around a detected Blob is a match for a given Target or the previous frame Blob.

If Fuzzy Histogram and Area matching methods did not provide enough accuracy to unambiguously pick or reject all the generated Hypotheses we use an implementation of Lucas-Kanade method to increase the confidence in a particular Hypothesis.

Because LK calculations are CPU intensive we are making two major optimizations in using it. First, we use LK only when it is impossible to disambiguate a Hypothesis without it, i.e. only after using Fuzzy Histogram and Area matching, if we

still don't have a clear cut on the set of data association hypothesis. Second, we are not calculating LK optical flow on full size image but we are cropping sub-images around the Blobs and the Target of interest and apply the algorithm for LK matching only on the selected sub-images as seen in Fig. 30.



Figure 30 LK Cropped Window and Mask

To perform LK matching we select 2 images of the size a bit larger than the maximum size of bounding rectangles of both Target and Blob(s) and on the Target image detect Shi-Tomasi features. We retain for matching only those features that are located either inside or at the borders of the binary masks of the blobs that are part of the Target.


```

lkMatchingScore(Target, Blob):
    rectSz=max(TargetRectSz, BlobRectSz)+SmallBorder
    tgtImg=ImageAroundTarget(rectSz);
    blbImg=ImageAroundBlob(rectSz);
    mask=binaryImageOf(AllBlobsInTarget)
    goodFeaturesToTrack= Shi-Tomasi(TargetRect)
    usedFeature = goodFeaturesToTrack & mask
    resFeatures=calcOpticalFlowPyrLK(tgtImg, blbImg, usedFeatures)
    return count(resFeatures)/count(usedFeatures)

```

Figure 31 LK Tracking Procedure

Then we calculate the vector of matching features using Lucas-Kanade optical flow method. The returned matching score is the number of features detected on the second image divided by the number of the featured that were passed to the matching algorithm, i.e. located inside or at the border of the Target's blobs in the first image. The procedure is illustrated in pseudo-code in Fig. 31.

8.7 Hypothesis Management

For each frame, the Targets already tracked will lay claims to all the Blobs detected by the subtraction of the current image from the Gaussian Mixture Model of the background. Each claim is called a Hypothesis and it is a triplet {TargetId, set<BlobId>, confidence}. The set of Blobs in the hypothesis is all the blobs that intersect the EBR of the Target.

It is quite possible at this point that if two or more Target EBR's intersect the same Blob than it will be assigned to multiple hypothesis. This is called an Ambiguity and it is the job of Ambiguity Resolving Algorithm to try to resolve them.

The original confidence associated with a hypothesis is calculated based on the size of the Uncertainty Rectangle calculated with formula (2). The bigger the Uncertainty Rectangle the smaller the confidence that is associated to that hypothesis when the claims are laid.

Ambiguity Resolving Algorithm will list all the Hypotheses that share one or more Blobs and for each shared Blob a score is calculated based on Fuzzy Histogram and Area. The common Blob is then assigned to the clear winner. If there is not a clear winner (i.e. having a score with at least 20% higher than the next contender) and if the size of the Blobs in question is over a minimal size required for LK to provide meaningful results, Lucas-Kanade matching is employed to update the confidence. After a particular blob has been removed from a multi-blob Target the confidence is re-initialized at the value resulting from EBR size and updated back with the score from FH and Area for the remaining Blobs. LK is not used again at this point until required because of the remaining ambiguities.

The confidence in the hypothesis is never assigned from scratch but is updated from the previous one based on the formula:

$$Confidence = (1 - \alpha) \cdot Confidence + \alpha \cdot score$$

where *Confidence* is the confidence already assigned to the Hypothesis and the score is the result from the last test performed. The alpha coefficient is dependent of the level of trust on the particular test. The value for alpha is small for the Area Match because the area of a detected blob can vary wildly due to occlusion and high for FH and LK match which have proved to provide high quality results.

There are two methods for picking a hypothesis in order to perform data association between Targets and Blobs: Unambiguous picking and Ambiguous picking.

Unambiguously picking is the method of first choice. We select a hypothesis with a confidence over a certain threshold such that no Blobs associated with this hypothesis are claimed by any other hypothesis. It is employed early on after just FH and Area updates. If ambiguities persist another attempt for Unambiguous picking is attempted after LK update.

Ambiguous picking is used as the solution of last resort before labeling all the remaining Blobs as leftover and resort to Second Chance Algorithm. Ambiguous picking selects Hypothesis with confidence above a given high-threshold given that no other competing hypothesis containing a Blob shared with this one have a confidence over a low-threshold. All the hypotheses that were not accepted by the Ambiguous picking will be discarded since no further Hypothesis processing will happen after this point.

8.8 Second Chance Algorithm

The Second Chance Algorithm assumes that matching between the Target and Blobs failed because either multiple Blobs are located too close for the Background Subtraction and Segmentation to differentiate between them; or because the Target took a movement incompatible with Kalman Filter prediction. This latest case can happen for example when a ball hits a wall and the trajectory diverges significantly from what Kalman Filter's state equation can handle. The Second Chance Algorithm will use the Search Rectangle defined as the rectangle containing the Target if it would move from the previous known position at what is assumed to be the maximum speed. More precisely,

if the last confirmed position of the target center was (x,y) and the Target was contained in a rectangle with dimensions (w,h) the Search Rectangle is centered at (x,y) and has dimensions:

$$(sW, sH) = (w + 2 \cdot vX \cdot dt, h + 2 \cdot vY \cdot dt)$$

where vX , vY are the maximum expected speed (in pixel / second) for a Target on the respective coordinates and dt is the duration of a frame.

Here the algorithm makes use of innate knowledge about the environment, in the form of a function provided by the Settings class which based on the position and size of an object will estimate a maximum speed expected for that object. The method assumes that small blobs are farther away while very large blobs are closer to the camera, and returns an expected maximum speed for each Blob. This expected maximum speed is used in the calculation of the search rectangle as described above.

The Second Chance Algorithm relies on brute-force Lucas-Kanade matching to find the image of the last known Target into the Search Rectangle. At this point we may not have distinguishable Blobs to update the tracking based on their center. To update the Kalman filter with the new position estimate $E(xE,yE)$ we first calculate the center of mass of the LK matching points in the new frame $C2(x2c, y2c)$ and in previous frame $C1(x1c, y1c)$ then we calculate the point E such that the offset from E to $C2$ is the same as the offset from L to $C1$, $L(xL,yL)$ is the previous known position.

$$\begin{aligned}
(x1_c, y1_c) &= \frac{1}{N} \cdot \sum_{i=0}^{N-1} (x1, y1)_i \\
(x2_c, y2_c) &= \frac{1}{N} \cdot \sum_{i=0}^{N-1} (x2, y2)_i \\
(xE, yE) &= (xL, yL) + (x2_c, y2_c) - (x1_c, y1_c)
\end{aligned}$$

9 VISION BASED NAVIGATION

A camera mounted on the robot offers a whole new set of opportunities for the robot to navigate the environment. Among them we will be focusing on localization of the stationary robot by recognizing multiple landmarks with well-known positions, localization of the moving robot by repeated angle measurements from a single known landmark and maintaining a prescribed trajectory by using a PID (Proportional-Integrative-Differential) controller using as feedback information the optical flow vectors measured while looking at the floor.

9.1 Multiple Landmark Localization of a Stationary Robot

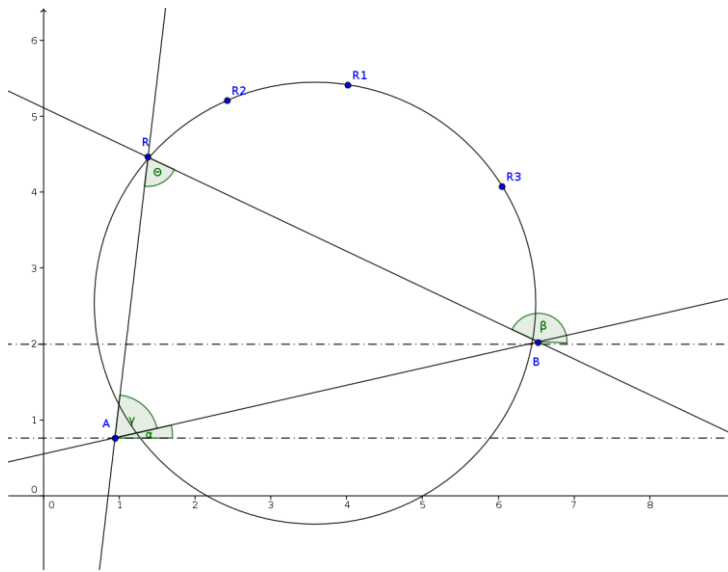


Figure 32 Geometric Locus of a Robot Keeping the Same Angle of Sight between Two Landmarks

In multiple landmark localization, the robot is able to recognize a set of already known landmarks (doors, windows or wall mounted pictures) in the environment and knowing

their location on the map of the room will be able to calculate its own position by measuring the angles of view between the known landmarks. It is easy to see that only two landmarks are not enough for localization if only their two dimensional position in the room coordinates is known.

Let's consider two landmarks A and B with their well-known position $A(X_A, Y_A)$ and $B(X_B, Y_B)$ in the room coordinates. The robot located at point $R(x, y)$ sees them at the constant angle θ .

From the Figure above we can derive the set of equations:

$$\begin{aligned}\tan\alpha &= \frac{Y_B - Y_A}{X_B - X_A} \\ \tan(\alpha + \gamma) &= \frac{y - Y_A}{x - X_A} \\ \beta &= \alpha + \gamma + \theta \\ \tan\beta &= \tan(\alpha + \gamma + \theta) = \frac{y - Y_B}{x - X_B}\end{aligned}$$

Using the trigonometric identities to split the last equation and replacing the values from the previous relations in it, after algebraic manipulations, we obtain the equation of the geometric locus of the points $R(x, y)$ which are able to see the landmarks A and B under the angle θ as:

$$\begin{aligned}x^2 + y^2 + \left(\frac{Y_B - Y_A}{\tan\theta} - (X_A + X_B)\right)x + \left(\frac{X_A - X_B}{\tan\theta} - (Y_A + Y_B)\right)y + (X_A X_B + Y_A Y_B \\ + \frac{X_B Y_A - X_A Y_B}{\tan\theta}) = 0\end{aligned}$$

Which is the equation of a circle, therefore the robot R can be located on any point around this circle having the center (X_0, Y_0) and radius r :

$$X_0 = \frac{X_A + X_B}{2} - \frac{Y_B - Y_A}{2 \cdot \tan\theta}$$

$$Y_0 = \frac{Y_A + Y_B}{2} - \frac{X_A - X_B}{2 \cdot \tan\theta}$$

$$r = \sqrt{X_0^2 + Y_0^2 - (X_A X_B + Y_A Y_B + \frac{X_B Y_A - X_A Y_B}{\tan\theta})}$$

The equations above allows us to define half of the geometric locus of the points able to see the two landmarks under a constant angle.

Beside the circle (X_0, Y_0, r) there is another circle that is also a valid solution to the locus problem above. It is the circle that has the same radius r and has its center at the reflection of point (X_0, Y_0) on the segment AB as depicted bellow in the figure (33).

To find the reflected point we write the equation of the segment on the mid-point of the segment AB which should contain the point (X_0, Y_0) and calculate the (X'_0, Y'_0) such as the midpoint of AB is also the midpoint of (X_0, Y_0) (X'_0, Y'_0) .

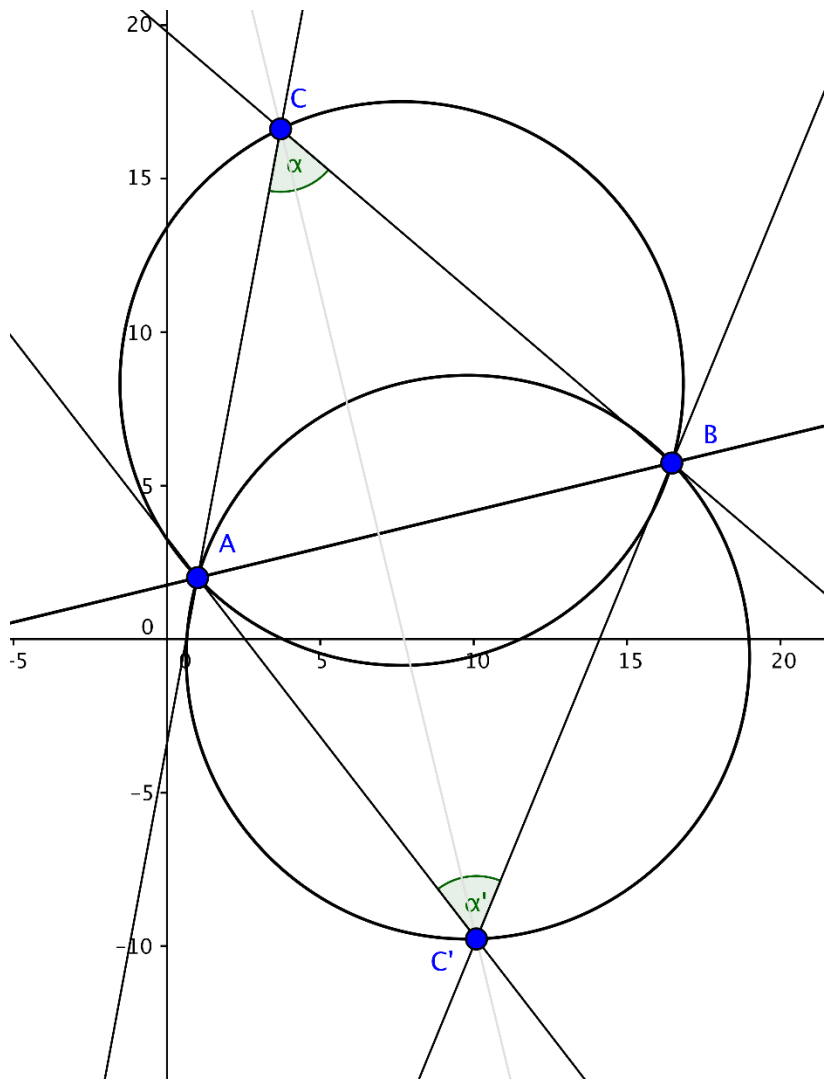


Figure 33 Complete Locus as a Reunion of Two Circles

The position of the robot on the above circle is however subject to a restriction stemming from the fact that in robotic vision, landmarks have a distinct identity and we can put the restriction based on the map that one landmark must be seen in the right side of the other if the robot is located upside-up on the room. It is also possible to eliminate all the segments of the arch of the circle that will fall outside of the geometry of the room. This

can reduce the localization possibilities and may ease computations.

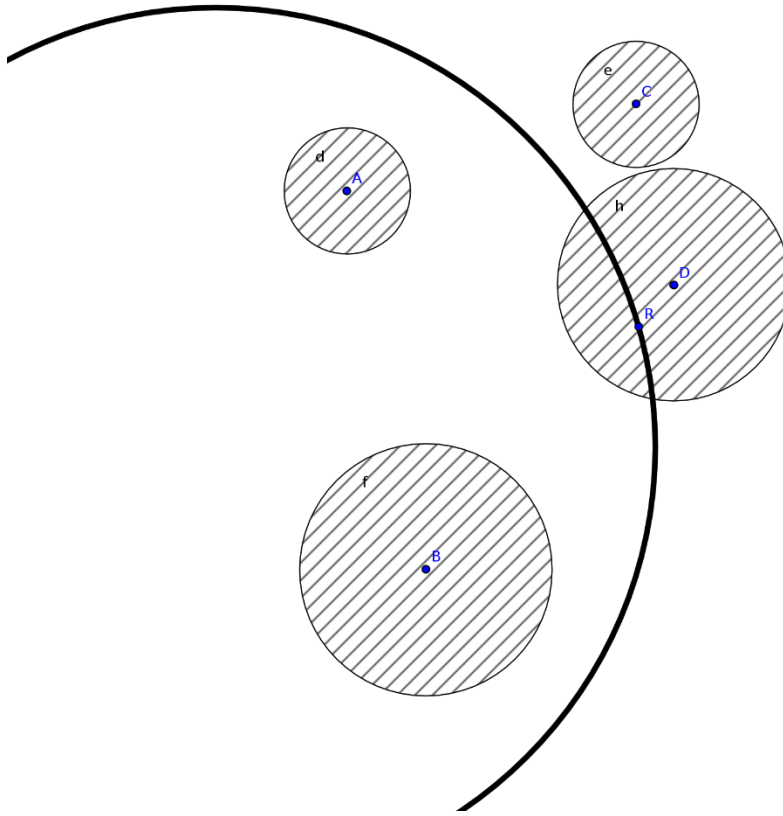


Figure 34 Disambiguation Using External Blob Tracking Information

Unfortunately, even with the above restrictions the location of the robot can be anywhere on the allowed arch, therefore supplementary information is required for position calculation.

One of the possibilities can be the usage tracking information from a fixed camera, as provided by the MP-Tracker algorithm. For example, in the figure on the side, the tracking algorithm detected four targets A, B, C and D and their uncertainty in location, while the camera on the robot used two landmarks to calculate the potential location along the large circle depicted in Fig. 34. Since the only blob intersecting the

locus is the one associated with the target D, the robot can now unambiguously identify itself as being target D located somewhere around point A on the allowed arch. While this is not a precise point location, it can be good enough for many practical applications.

However, if the robot has the ability to detect a third landmark in the environment, we can theoretically collapse the space of solutions to a set of maximum 8 points that can therefore be disambiguated by restrictions and/or external information. In the derivation of the equations below, we ignore the reflections and we consider a single circle as a locus for each pair of landmarks. On the real algorithm however we will consider all the reflections, resulting in a set of 8 distinct cases that each of them shall apply the equations that are being derived below.

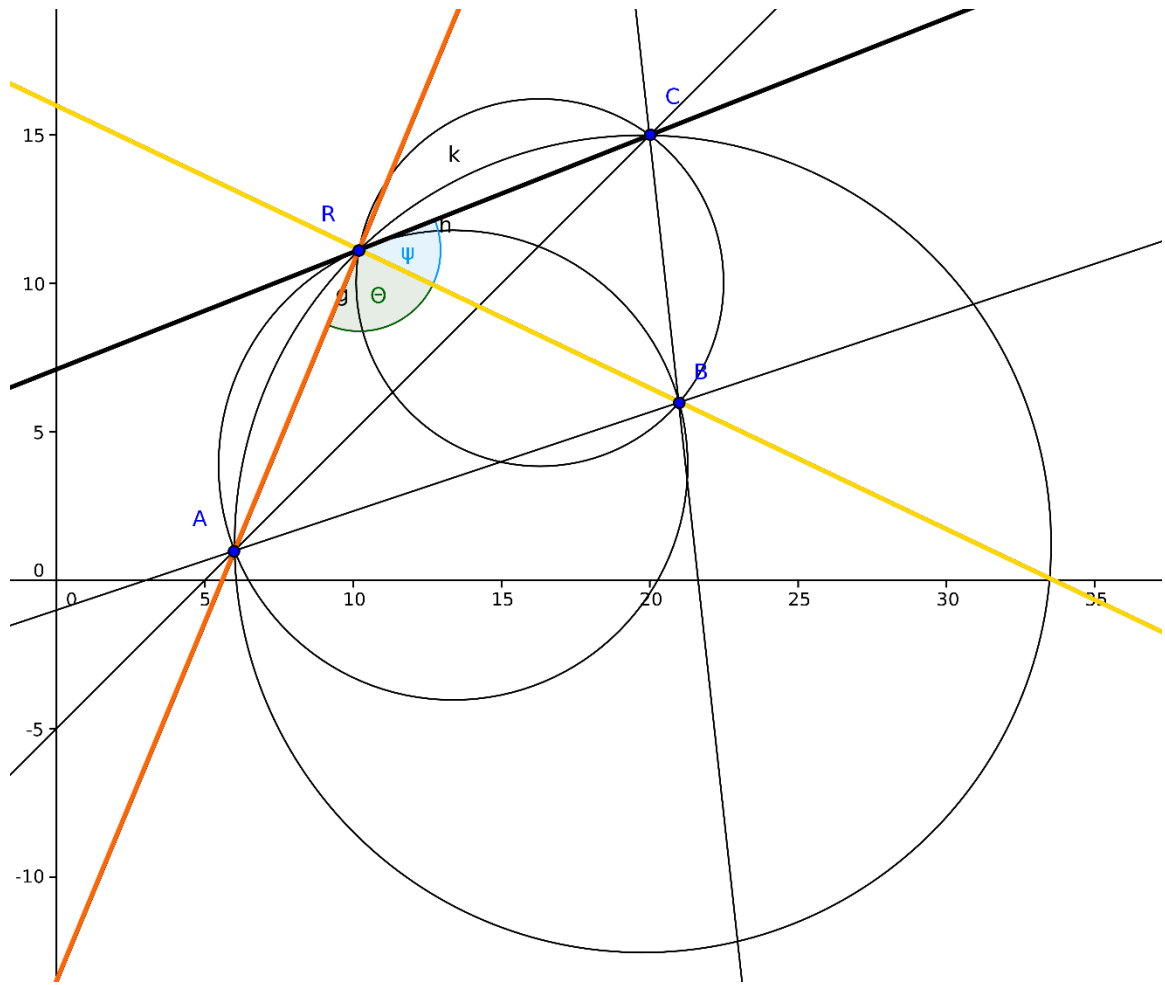


Figure 35 Exact Localization with Three Landmarks

In three landmarks localization, the robot located at the (to be determined) point R sees three landmarks A, B, and C and measures the angle Θ between A and C, Ψ between B and C and finally the angle $\Theta + \Psi$ between A and C. Each of the three known angles and their associated landmarks allows us to use the equation derived above to write a system defining the three circles determined by each triad (angle, landmark coordinates) as:

$$\begin{aligned} x^2 + y^2 + m_1x + n_1y + p_1 &= 0 \\ x^2 + y^2 + m_2x + n_2y + p_2 &= 0 \\ x^2 + y^2 + m_3x + n_3y + p_3 &= 0 \end{aligned}$$

Where the parameters m , n and p can be calculated from the two measurements of angles and known landmark position:

$$\begin{aligned} m_1 &= \left(\frac{Y_B - Y_A}{\tan \Theta} - (X_A + X_B) \right) \\ n_1 &= \left(\frac{X_A - X_B}{\tan \Theta} - (Y_A + Y_B) \right) \\ p_1 &= X_A X_B + Y_A Y_B + \frac{X_B Y_A - X_A Y_B}{\tan \Theta} \end{aligned}$$

$$\begin{aligned} m_2 &= \left(\frac{Y_C - Y_B}{\tan \Psi} - (X_B + X_C) \right) \\ n_2 &= \left(\frac{X_B - X_C}{\tan \Psi} - (Y_B + Y_C) \right) \\ p_2 &= X_B X_C + Y_B Y_C + \frac{X_C Y_B - X_B Y_C}{\tan \Psi} \end{aligned}$$

$$\begin{aligned} m_3 &= \left(\frac{Y_C - Y_A}{\tan(\Theta + \Psi)} - (X_A + X_C) \right) \\ n_3 &= \left(\frac{X_A - X_C}{\tan(\Theta + \Psi)} - (Y_A + Y_C) \right) \\ p_3 &= X_A X_C + Y_A Y_C + \frac{X_C Y_A - X_A Y_C}{\tan(\Theta + \Psi)} \end{aligned}$$

To solve this system, we subtract one of the equations from the other two. The resulted system:

$$\begin{aligned} (m_2 - m_1)x + (n_2 - n_1)y + p_2 - p_1 &= 0 \\ (m_3 - m_1)x + (n_3 - n_1)y + p_3 - p_1 &= 0 \end{aligned}$$

represents the equations of the two lines determined by the points of intersection between first circle with each of the other two, in the figure above the straight lines: AR and BR.

These two lines will intersect at the location of the robot, point R which is therefore

described by the coordinates:

$$x = \frac{(n_3 - n_1)(p_2 - p_1) - (n_2 - n_1)(p_3 - p_1)}{(n_2 - n_1)(m_3 - m_1) - (n_3 - n_1)(m_2 - m_1)}$$

$$y = \frac{(p_2 - p_1) + (m_2 - m_1)x}{n_1 - n_2}$$

For a robot in our line of work, which is operating inside a house with a well-known floor-plan, the preconditions to know exactly the position of a certain number of remarkable landmarks (windows, doors, room corners, picture frames mounted on the wall or active landmarks that the robot can control) is easily achievable.

The algorithm that uses these equations for landmark based localization will calculate the pairs of two mirror circles for each pair of landmarks and will form all 8 possible combinations. For each combination the equations above will be calculated. In the first step of elimination we will discard the solutions of the circles that do not intersect. This is required since the subtraction of two circle equations generates the equation of the **radical-line** (or **power-line**) of two circles which is an equation that exists even if the circles does not intersect. However this case has no meaning in our localization so it should be discarded up-front.

From the remaining results we eliminate the cases where the point defined by the intersections of the radical-lines does not reside on all three circles, because the only acceptable solutions to our problem has to be located at the intersection of all three circles. To do this, we evaluate each point result from the equations above to verify each circle equation. Finally, we check the remaining points against the geometry of the room

and any external inputs we have in order to eliminate all the unsuitable solutions.

9.2 Localization of a Mobile Robot Using a Single Landmarks (Running Fix)

In this scenario, the robot is able to maintain a straight line during navigation and is capable to measure the distance it travels by an accurate odometer. The position of a landmark $L(x_L, y_L)$ is also known and a set of angle measurements from the robot to the landmark are taken along the way.

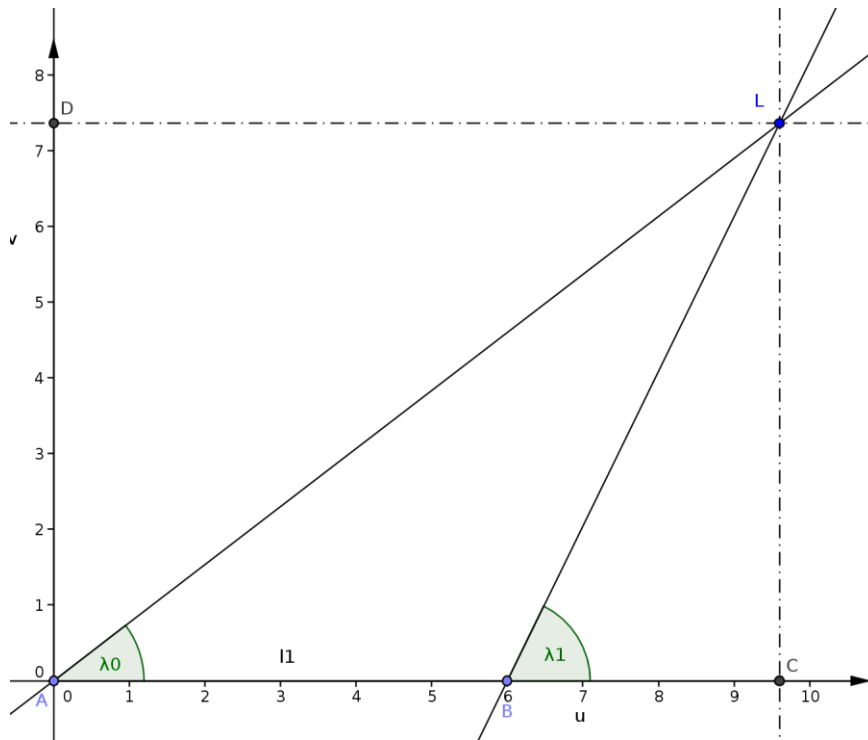


Figure 36 Moving Robot and One Landmark in Robot Coordinates

More precisely, the robot sees the landmark L at an angle λ_0 from its direction of movement, then it moves in a straight line for a distance l_1 as measured by the on-board odometer, then it determines the landmark to be visible under the angle λ_1 . The situation

from the robot point of view, i.e. in robot coordinates, is depicted in the figure bellow.

Considering the system of coordinates (u,v) having as origin the original location of the robot we have can write the trigonometric equations:

$$\tan\lambda_0 = \frac{L_v}{L_u}$$

$$\tan\lambda_1 = \frac{L_v}{L_u - l_1}$$

From this system of equations, we can calculate the position of the landmark from the robots point of view as:

$$L_u = l_1 \frac{\tan\lambda_1}{\tan\lambda_1 - \tan\lambda_0}$$

$$L_v = l_1 \frac{\tan\lambda_1 \tan\lambda_0}{\tan\lambda_1 - \tan\lambda_0}$$

Switching back to room based coordinates where the position of the landmark $L(L_x, L_y)$ is known the above results defines the position of the robot as the geometric locus of the points R located at L_u distance on any straight line tangent at a circle of radius L_v around the landmark L. This definition is in itself a circle therefore the position of the robot cannot be fully known from this single observation.

It is easy to see that taking any more measurements along the way just generate new equations that are a linear combination of the original two points, therefore they add nothing new in respect to localization. External information is therefore needed to collapse the robot position to a single point.

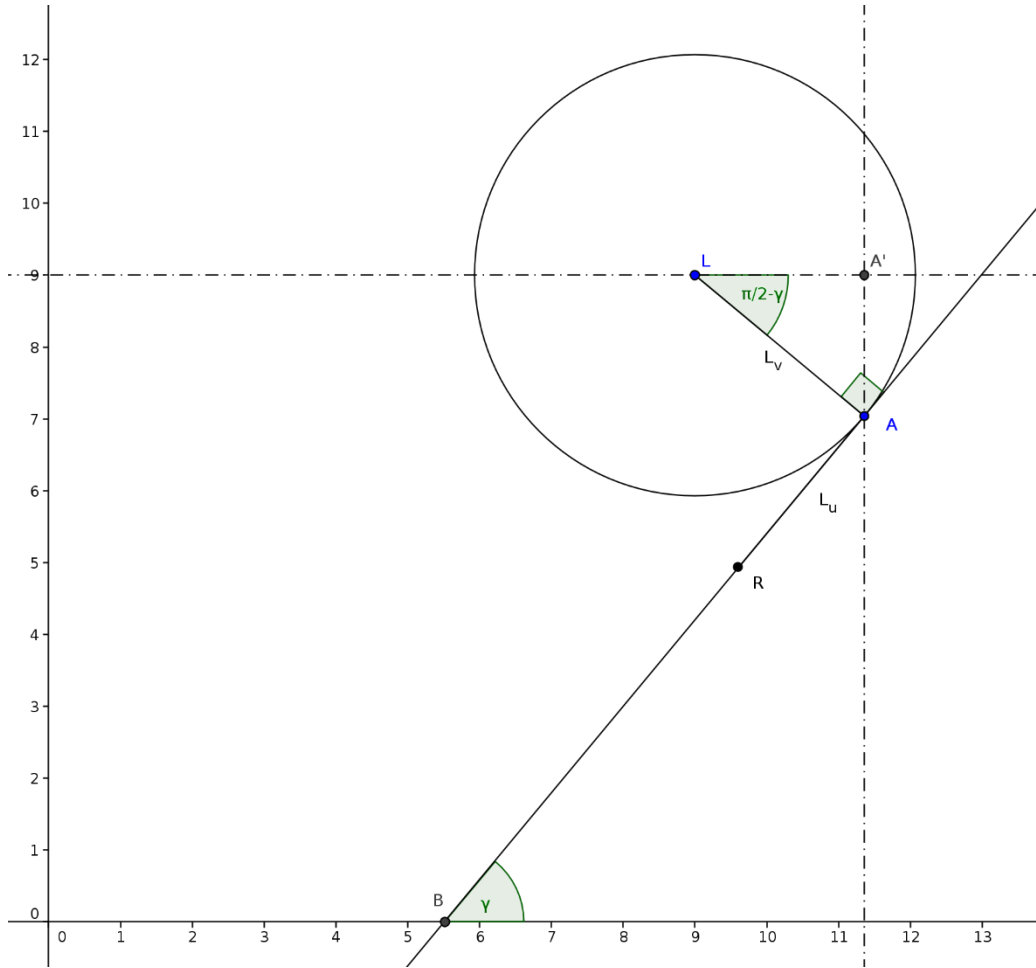


Figure 37 Moving Robot and One Landmark in Room Coordinates

One possible source of such information can be the direction of the movement which for a robot moving outdoor can be obtained via a digital compass (magnetometer) or from an indoor robot by averaging the tracking information from an external camera.

If the angle γ specifying the direction of robot movement is known, from the triangle ALA' we can calculate the position of the point of the tangency between the robot line of movement and the circle on which this point can be located based on the difference in landmark position:

$$|AA'| = L_v \sin\left(\frac{\pi}{2} - \gamma\right) = L_v \cos\gamma$$

$$|LA'| = L_v \cos\left(\frac{\pi}{2} - \gamma\right) = L_v \sin\gamma$$

$$A_x = L_x + L_v \sin\gamma$$

$$A_y = L_y - L_v \cos\gamma$$

and these relations then allows us to calculate the exact robot position R in the first spot as:

$$R_x = L_x + L_v \sin\gamma - L_u \cos\gamma$$

$$R_y = L_y - L_v \cos\gamma - L_u \sin\gamma$$

Once the first position has been determined, the robot can easily calculate all the subsequent positions on the trajectory.

As an interesting fact we can notice here that in maritime navigation a graphic method to determine the position of a ship on the map only by a single landmark and the reading of the magnetic compass, time and ship speed over the water is called a “Running Fix”. Practically, the ship captain solves the equations above with a ruler and compass on the map.

9.3 Visual Odometry by Optical Flow

One of the important problems in robot navigation it is the ability to estimate the change in position over time, also known as odometry. In the case of our robot, the on-board camera can successfully be employed for this task.

9.3.1 Method

Once we determine the velocity vector field from the robot camera point of view, we can back-project it using the Homography Matrix onto a model of the flat floor. The result is a vector field representing the movement of the robot in reference to the floor. This vector field can be used to calculate the robot's movements in reference to the floor, using the method we present here.

As already derived in the chapter, in differential kinematics sub-chapter, the robot is moving in a circle with a radius determined by the difference in the wheel speed. When the two wheels of the differential drive have the same speed the radius of the circle is infinite, therefore the robot is navigating a straight line. We derive here the general equation for the case.

In visual odometry we have to calculate both the position of the center of the rotation and the angle of rotation. Therefore, a minimum of 2 different point pairs must be detected by the Optical Flow algorithm to allow calculation. In practice however, a larger number of correspondence are needed, to calculate the parameters for all possible sets of 2 points then eliminate outliers and average the position to be able to cope with the high level of imprecision in the visually acquired data.

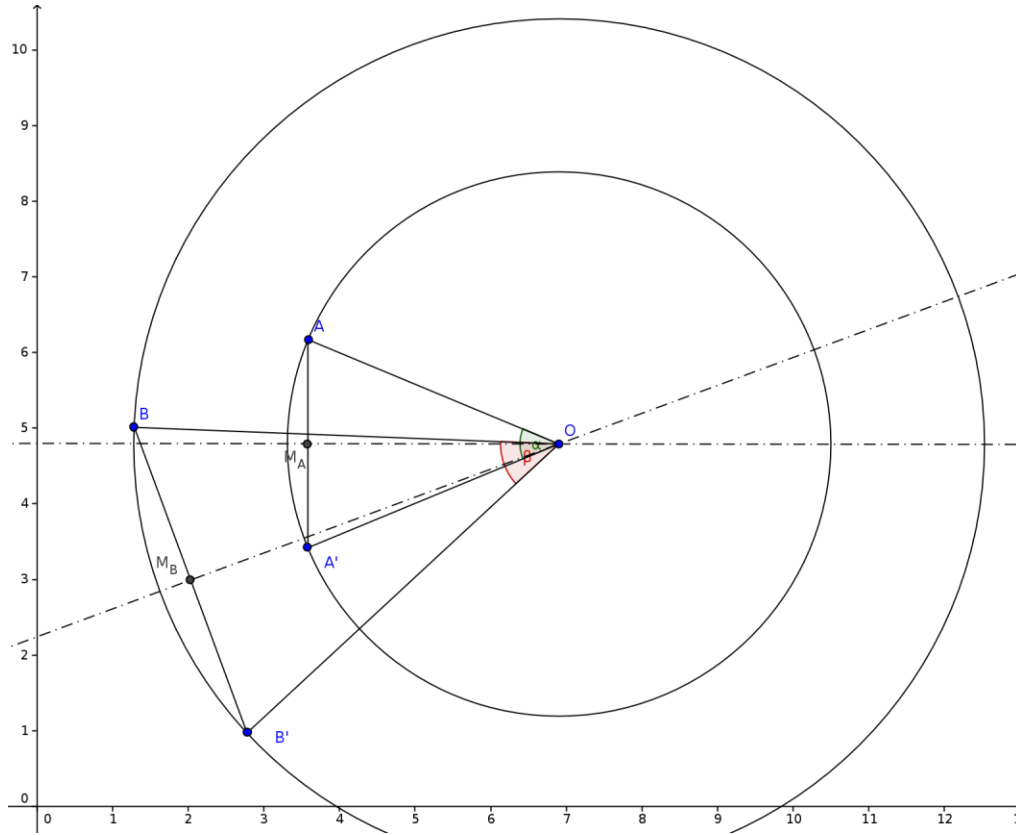


Figure 38 Calculating Visual Odometry

In theory however, a single pair of independent points is required for calculations.

As seen from the side figure, we detected two points A and B which in the subsequent image we found them shifted at the position A' and B' respectively.

Because of the differential drive, we know that the robot moved on a circle around the, to be determined, center of rotation O.

To determine the position of the point O we first calculate the slopes and the mid points of the segments AA' and BB' respectively, as:

$$m_A = \frac{y_A' - y_A}{x_A' - x_A}; x_{MA} = \frac{x_A' + x_A}{2}; y_{MA} = \frac{y_A' + y_A}{2}$$

$$m_B = \frac{y_B' - y_B}{x_B' - x_B}; x_{MB} = \frac{x_B' + x_B}{2}; y_{MB} = \frac{y_B' + y_B}{2}$$

with these parameters we can write the equations of the two perpendicular bisectors of the segments AA' and respectively BB' to be:

$$y - y_{MA} = \frac{-1}{m_A} \cdot (x - x_{MA})$$

$$y - y_{MB} = \frac{-1}{m_B} \cdot (x - x_{MB})$$

These two lines intersects at the point O(x_O,y_O) whose coordinates are calculated by solving the system of equations above resulting in:

$$x_O = \frac{m_A m_B (y_{MB} - y_{MA}) + m_A x_{MB} - m_B x_{MA}}{m_A - m_B}$$

$$y_O = y_{MA} - \frac{1}{m_A} (x_O - x_{MA})$$

The remaining problem is to calculate the rotation angle. In the image above this is either the angle α or β since regardless of the point in question they all rotate with the same amount in the same given interval of time.

Therefore, for example from the triangle AOM_A we have:

$$\tan \frac{\alpha}{2} = \frac{|AM_A|}{|OMA|}$$

which results in:

$$\alpha = 2 \cdot \text{atan}\left(\frac{\sqrt{(x_{MA} - x_A)^2 + (y_{MA} - y_A)^2}}{\sqrt{(x_{MA} - x_O)^2 + (y_{MA} - y_O)^2}}\right)$$

9.3.2 Optical Flow Vector Classifier Algorithm

One of the problems we faced during implementation of visual odometer was coping with the errors generated by the optical flow detection algorithm. These errors in finding the right corresponding pairs appears in the form of vectors having a magnitude or an orientation “out of place” when compared with the large majority of the vectors surrounding them. They are called Outliers and unless eliminated they can significantly reduce the accuracy of calculations.

To help with the task of outlier’s elimination we developed our Vector Classifier algorithm that inspects the optical flow vector and attempts to provide as output a smaller set of vectors that are equivalent with the average movement into the image minus the errors.

The core idea in developing this algorithm was the fact that outliers are a small percentage of the overall optical flow vectors and they have direction and/or sizes that are quite different from the rest of the vectors nearby.

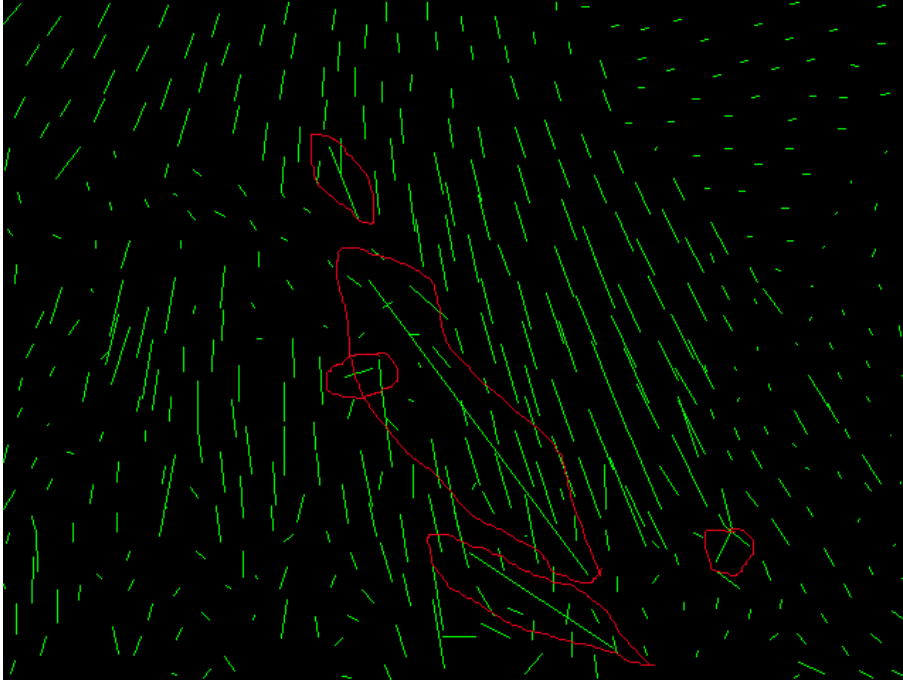


Figure 39 Optical Flow (velocity vector field) from a Forward Moving Camera Looking at the Floor with a Few Marked Outliers

In the beginning, we are dividing the camera field of view into a number of cells and we treat the vectors that originate from one cell as a group. For each vector group we use k-mean clustering algorithm to partition the set of vectors in k (in our case 3) groups such that each vector is located into the group with the nearest mean value. We eliminate the groups that either have too few elements or whose mean is too different from the other cluster from the same or surrounding cells.

For the remaining groups we calculate the mean vector and origin points which is the output of the algorithm. Prior to odometry calculation the vector field must be translated into floor coordinates to be agnostic of perspective camera projections. The perspective projection between the plane of the flat floor and the camera image is a homography described by the camera matrix. Before we pass the vectors to the visual

odometer algorithm, we apply a homography transformation to translate the vector space in the coordinates of the floor.

```
for v in opticalFlowVectors:
    cell=cells[v.x*XNo/width, v.y*Yno/height]
    cell.insert(v)

for cell in cells:
    cell.groups=calculate_k-mean(cell)
    group=lowestCount(cell.groups)
    if(group.size() < 3): delete group from cell;

for cell in cells:
    for group in cell.groups:
        keep=false
        for g in groups_in_surrounding_cells(cell)
            if(group.dx near g.dx && group.dy near g.dy):
                keep=true
                break
        if (not keep): delete group from cell
```

Figure 40 Pseudocode for Vector Classifier Algorithm

For homography transformation we use the OpenCV built-in function `perspectiveTransform(...)` to transform the vectors values output by the classifier and respectively `warpPerspective(...)` to transform images.

These two functions require the homography matrix to be known.

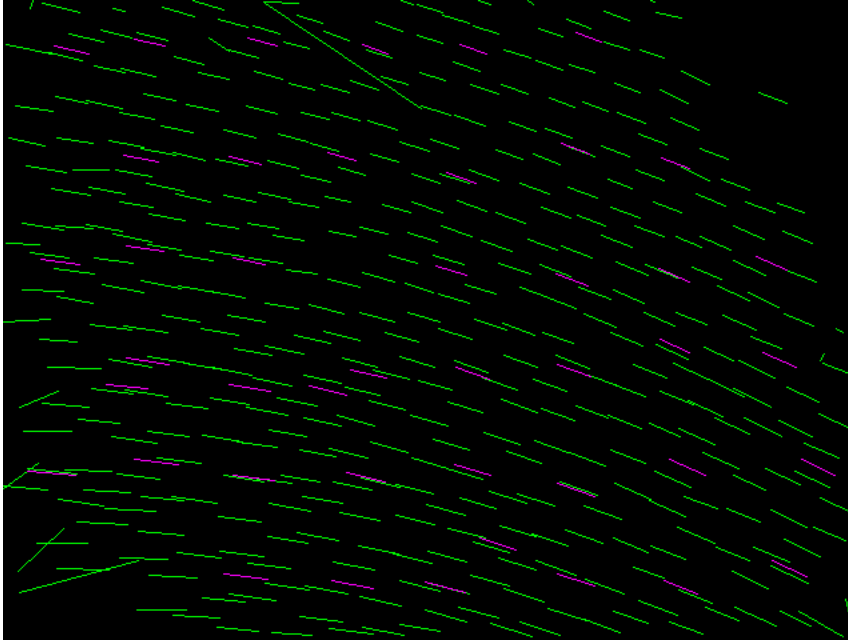


Figure 41 Optical Flow from a Rotating Camera Showing the Result of the Classifier (purple vectors). After a homography transform, “purple vectors” are ready for odometry computation as presented above

To “ease our lives” we used a little trick to allow the robot to be “taught” the matrix instead of us having to calculate it. OpenCV provides the function `getPerspectiveTransform(...)` which calculates the homography matrix from a set of four corresponding points. We take advantage of the model of flat floor and use a piece of paper positioned at the known distance from the robot. We command the camera of the robot at a particular angle and mark on the image the points corresponding to the corners of the paper, then we provide the real-world coordinates of the paper corners in meters from the center of the robot. Then `getPerspectiveTransform()` will calculate the exact correspondence between the real world coordinates and the image, we inverse the matrix calculated by the method above and save it in a database indexed by the angle of sight at which we commanded the camera. We repeat the procedure for a few angles of interest.

When in need of visual odometry, we just command the camera at one of the saved angles and retrieve its corresponding matrix from the database. Of course, this procedure would be in trouble outdoors on an irregular, sloped terrain. But for indoor robots always moving on a flat floor it is a quick and easy solution.

9.3.3 Homography in Speed Measurements

On the embedded board from the robot itself, the main computer vision processing task is to calculate the movement of the robot in order to provide an estimation of the direction of movement and speed. The results of this repeated set of measurements will be used to keep the robot on a prescribed trajectory for navigation since the experiments from tracking from fixed cameras showed that the imprecision of tracking measurement it is too big to be relied upon it for real-time PID feedback.

In order to ease the tasks for optical flow based measurements we are taking advantage of some prior-known or assumed conditions. First, we assume that being a domestic robot the environment we are operating into it features a flat horizontal floor. The second precondition is that the height of the robot is well known and that the servo motors running the pan-tilt mechanism of the robot camera will reliably position the camera reliably at the same angle for the same command. Based on these two assumptions we can take the short-cut to calibrate the robot camera directly by calculating the homography matrices for a set of well-known angles and store them on the robot flash card. When measurements need to be taken we just command the camera to one of the pre-selected angles and take the measurements.

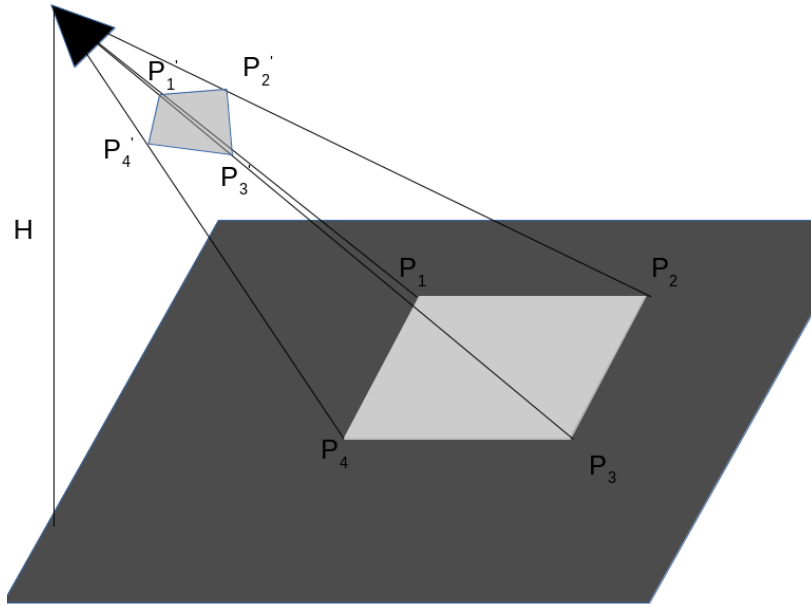


Figure 42 Robot Camera Calibration

While on the classic projection, there is no way to determine the real distances/sizes (the geometry is determined up to scale) in our case based on knowledge of the size of calibration figure and on the assumption of flat floor and constant height of the camera in respect to the floor, we are able to recover the full size of objects. Therefore, from the size of the calculated optical flow vector back-projected on the floor plane we are able to calculate the real speed of the robot.

Because there is always the possibility that moving objects can be present into the field of view of the robot the optical flow calculation will be taken with the robot camera oriented toward the floor to minimize the probability of false reading. We also implement an algorithm for clustering of the measurements and keep for the OF calculation only the major clusters that have a somehow similar orientation. This algorithm is also useful in eliminating the outliers inherent in any optical flow algorithm.

On the Calibration phase we are placing a rectangular sheet of paper on the floor (with corners labeled as the points P1, P2, P3, P4) and identify them in the picture as (P1', P2', P3', P4'). Using DLT algorithm we calculate the homography matrix H such that:

$$\begin{aligned}P_1 &= HP_1' \\P_2 &= HP_2' \\P_3 &= HP_3' \\P_4 &= HP_4'\end{aligned}$$

where the points P1...P4 are expressed in the real-world coordinates.

We are saving on the permanent storage the calculated H for each orientation of the robot camera that we plan to use in odometry. When performing navigation, the robot camera is commanded to one of the saved positions the H matrix retrieved and used to calculate the floor coordinate from the image points.

10 THE ROBOT – ROBI-1

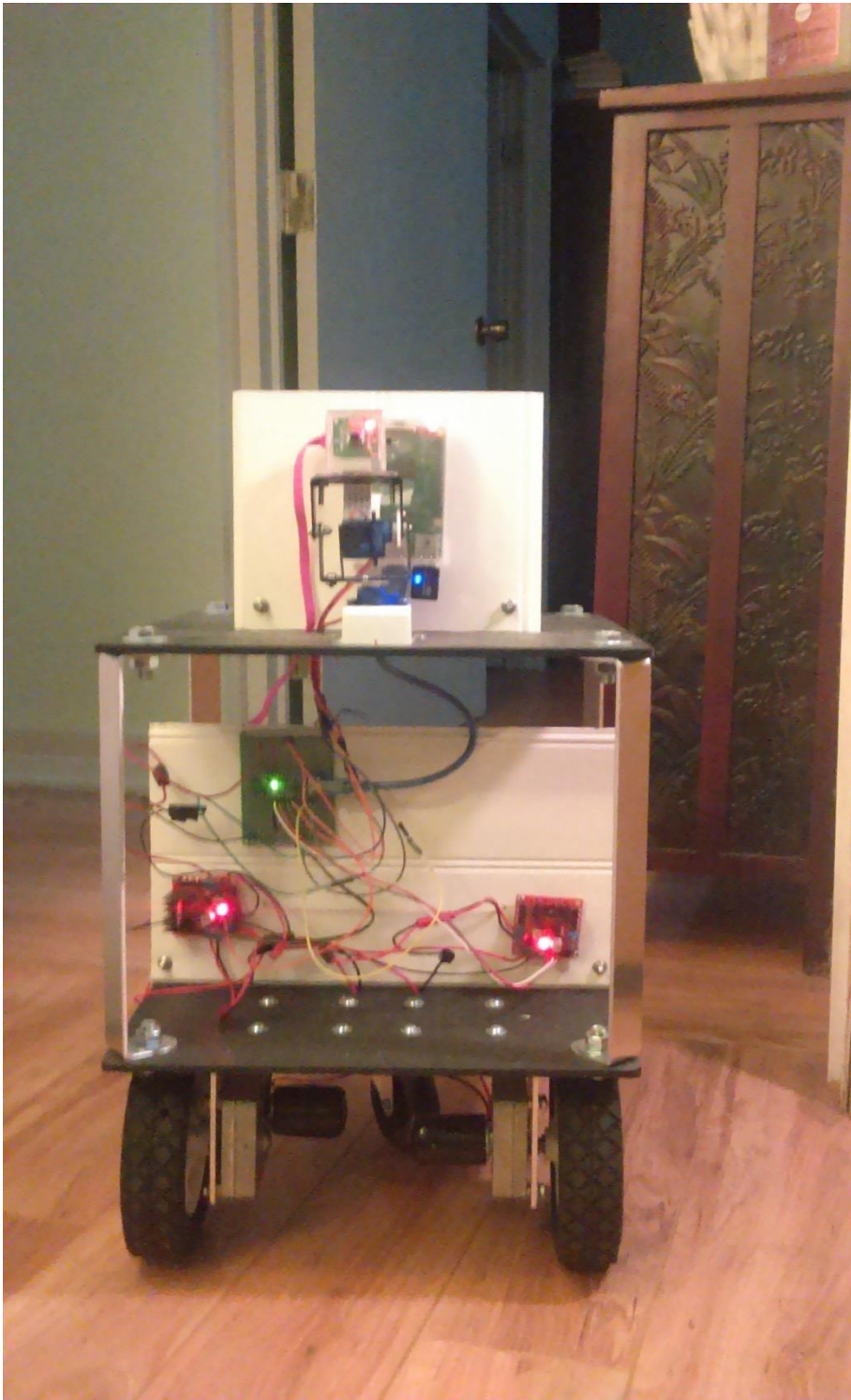


Figure 43 ROBI-1

ROBI-1 is the name given to the mobile robot built in order to test our system. The robot main embedded computer board is a Raspberry Pi 2 having a BCM2836 ARMv7 Quad Core Processor. It runs at 900MHz having 1GB RAM and 16GB mini-SD card as permanent storage.

Communication between robot and the rest of the system is taking place using sDOMO protocol over UDP/IP via an USB WiFi dongle connected to the RPI.



Figure 44 Raspberry PI2 - the Main Computer Board of the Robot.

10.1 ROBI-1 Hardware

Robi-1 has been implemented using MAX-97 base from Zagros Robotics. The base is a two deck with 10 inches' heights between decks and having the base a square with the edge of 12 inches. The base features a differential drive with two 12V DC-Motors capable of 20 in*lb maximum torque. The drive wheel has a 6-inch diameter allowing the robot to achieve a maximum speed of 39 feet/minute.



Figure 45 MAX-97 Base



Figure 46 Pan-Tilt Mechanism for Raspberry PI Camera

The 12 V battery is connected directly to dual H-Bridge Module build around the L298N circuit which is connected to the two drive motors. This module is also providing a 5V output which is used to power the micro-controller board and the servo-motors for the camera mount.

The 5MP RPI-Camera (2592 x 1944 max static resolution) is mounted on a mini pan-tilt module powered by two 9G mini-servo-motors. This allows the robot to adjust its line of sight about 60° vertically and 90° horizontally.

The WiFi USB dongle used to communicate with the rest of the system is identified as Ralink Technology RT5370 bought from CanaKits. Unless some other common WiFi dongles, this one is properly supporting multicast which is a requirement for sDOMO protocol.

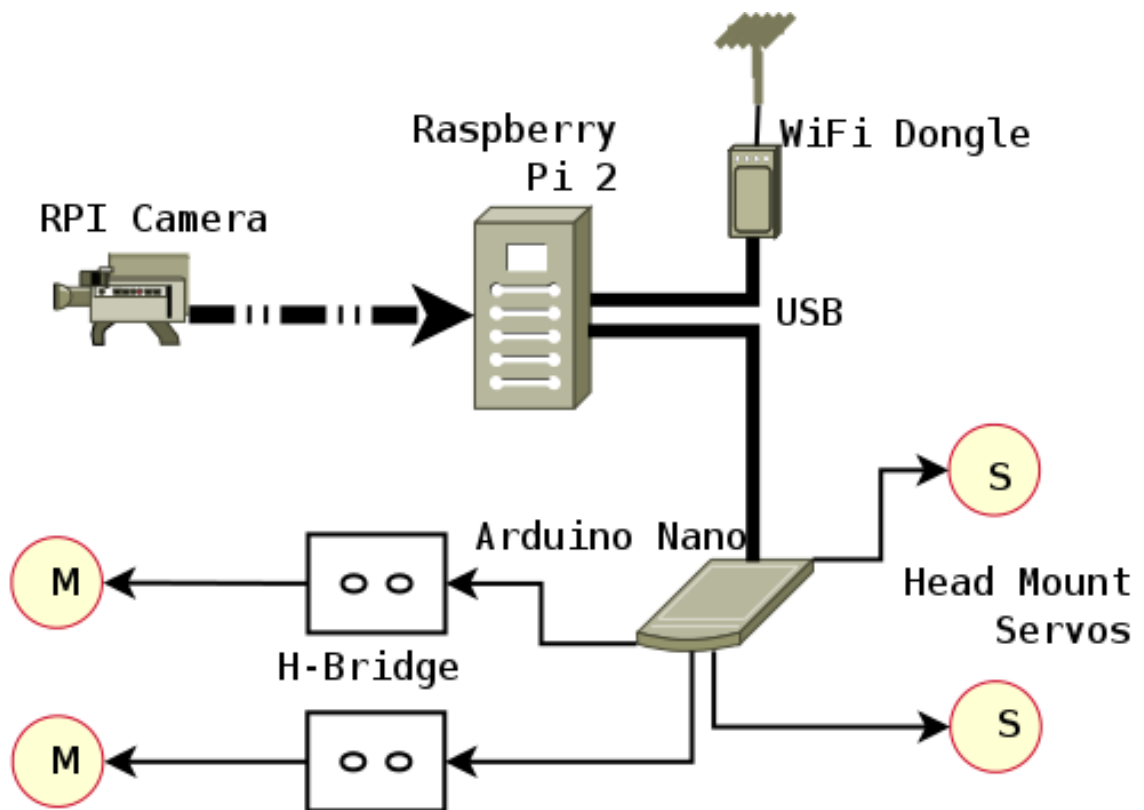


Figure 47 ROBI Hardware Block Schema

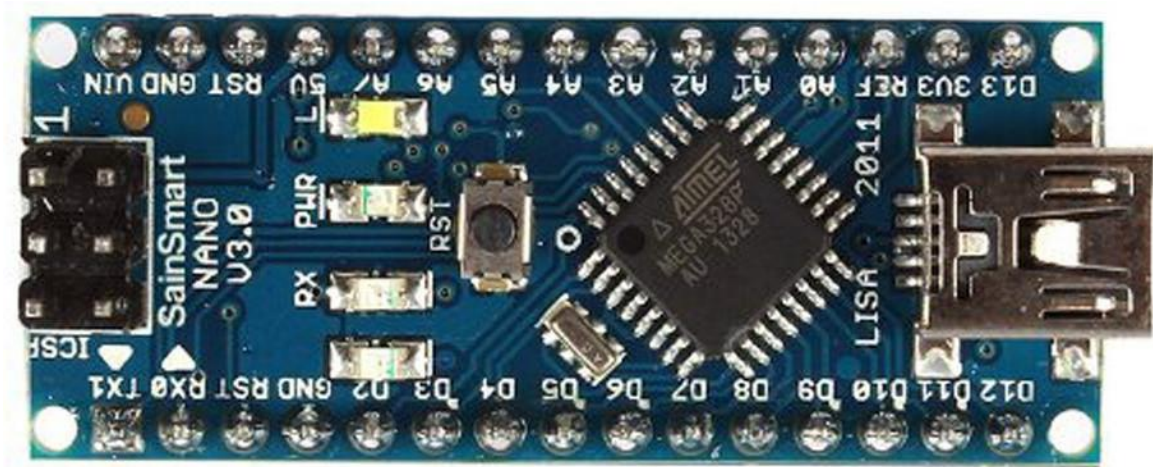


Figure 48 Arduino Nano Microcontroller Board

The PMW signals for controlling the servo motors from the camera mount and commanding H-Bridges for the wheels are generated by an Arduino Nano microcontroller board. The Arduino is connected via an USB cable to Raspberry PI.

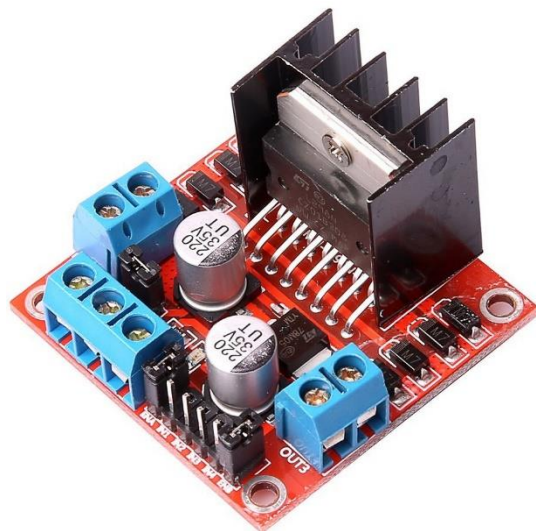


Figure 49 L298N Dual H-Bridge Module

In the implementation of the drive, we used two H-Bridge Modules built around the L298N circuit. While this module is practically a dual H-Bridge that can be used to drive

two independent motors on its own, we choose to use two of them and use a single bridge from each module. The rationale for that was to avoid the need for a cooling fan for the heat-sink of the L298N circuit. By using only half of the power the bridge is capable to control, the heat-sink that came with the module is good enough to dissipate all the generated heat without forced air cooling. And because of the low price of these standardized modules, it was just easier to use two of them instead of crafting a new mount for a cooling fan.



Figure 50 DC-DC Step down Converter

The 2 H-Bridge Modules along with the mini-servos from the camera mount are driven by PWM generated by an Arduino Nano version 3.0 board powered by an Atmel ATmega328 microcontroller.

Table 2 Wiring Table for Arduino Microcontroller

| | |
|-----------------|------------------------------------------------------------------------------|
| Arduino Pin 4 | Bridge 1 IN 2 |
| Arduino Pin 5 | Bridge 1 IN 1 |
| Arduino Pin 6 | Bridge 1 EN A |
| Arduino Pin 2 | Bridge 2 IN 2 |
| Arduino Pin 12 | Bridge 2 IN 1 |
| Arduino Pin 3 | Bridge 2 EN A |
| Arduino Pin 10 | Pan-Tilt Vertical Servo Signal (Orange wire) |
| Arduino Pin 11 | Pan-Tilt Horizontal Servo Signal (Orange wire) |
| Arduino GND Pin | Bridge 1 GND, Bridge 2 GND, Vert Servo GND, Horiz Servo GND (Brown wires) |
| Arduino 5V+ | Bridge 1 +5V, Vert Servo V+, Horiz Servo V+ (Red wires) |
| 12 V Battery - | Bridge 1 GND, Bridge 2 GND |
| 12 V Battery + | Bridge 1 12V+, Bridge 1 12V+ |

To power the Raspberry PI2 we used a 12 V 10 A/h sealed lead-acid battery which is often used in electric skateboards and telecommunication equipment. The battery is providing power to a 12V to 5V / 3A DC-DC converter which offers regulated voltage to the computer board via a Micro USB connector. The Wi-Fi dongle is connected to one of the main USB ports of the RPI2 which also provide it with power.



Figure 51 12 V Sealed Lead-Acid Battery

10.2 ROBI-1 Software Infrastructure

We used the same operating system Raspbian that came with RPI2 and we just installed development packages and other libraries that were needed. The version of Raspbian used features a Linux Kernel 3.18.7 compiled with PREEMPT-RT patches to provide real-time capabilities.

We compiled from sources OpenCV libraries version 2.4.10 and RaspbiCam library for OpenCV. This library is used to process images from the standard Raspbery PI Camera, we wrote an adapter around it to integrate it into Autonomous Robot Module.

Communication with the Arduino board is taking place over it FT232 USB-Serial converter available on the Arduino Nano board. This chip is presented on Linux as the serial port `/dev/ttyUSB0` and is opened and worked with using regular Serial Port system calls. We used for communication 115200 bauds with 8-bit character size 1 stop bit and no flow control.

The WiFi dongle is seen by the operating system as the interface `wlan0` and its IP address is automatically configured by the DHCP server on the local network. Unless other protocols that requires well known addresses, sDOMO employs its own procedure of discovery and self-configuration which allows us to just plug and play any computer into the robotic network regardless how its IP address has been configured.

For the firmware running on the Arduino, we used the Servo Motor library for controlling the camera pan-tilt mechanism. Because Servo library is internally using one of the microcontroller timers, using this library can interfere with the `analogWrite()` function used to generate PWM for the differential drive. Only certain configurations for pin

assignment can be possible that can allow the generation of usable PWM while the servo library is used. The assignment provided in the wiring table above where we are using pins 3 and 6 for H-Bridge PWM and pins 10 and 11 for Servo PWM has been proven to work. Notice that the pins 2, 4, 5, 12 in our set-up are simple GPIO lines and are not used for PWM generation.

The firmware written for Arduino is a simple loop reading inputs from the serial line which specify the speed of each motor wheel as a percent of the maximum speed possible and the absolute angles for the camera in horizontal and vertical set-up. As soon as a command has been received the firmware commands the hardware accordingly and also provides acknowledgement by sending back the current commanded values to the ARM. If no command has been received (on idle) the same previous report is sent back to RPI at 0.33 Hz.

In order to be able to control the microcontroller software both from a terminal during development and from Autonomous Robot Module during regular operations the commands we sent to the microcontroller are an ASCII string with a very simple structure to be easily parsed by the resource constrained microcontroller. The format of command consists on four 2 digit numbers each prefixed with a sign (+/-) contained between 2 special characters. The generic command line sent from ARM to Arduino is described as:

@sLLsRRsVVsHH&

Where LL, RR, VV and HH are 2 digits' percent of the commands to control the speed of Left and Right wheels and the Elevation and Azimuth of the Pan-Tilt mechanism respectively. Each number is prefixed with a sign denoted by s. The two special

characters @ and & are the delimiters of the command string and are used by the parsing procedure to detect the boundaries of the command.

The report from Arduino to RPI is in human readable format since on RPI we have enough computing resources to engage in parsing of more complex strings.

The RPI keeps sending the command string at 2Hz in order to keep the robot running. If no command has been received for over a second, the firmware stops the robot immediately. This is done to prevent the robot to keep moving and bumping into furniture, people or pets in the case that the main control software running on RPI crashes or otherwise became unresponsive. The software running on the Raspberry PI keeps sending the commands twice a second how long it intends to keep moving to make sure it prevents the firmware from stopping the motion.

10.3 ROBI-1 – Autonomous Robot Module

Autonomous Robot Module (ARM) is the software running on the RPI2 on the robot having as main functions:

- Communicate with Robot Module to receive commands and provide status and requested images
- Translate high level navigation commands into the command parameters for Arduino Microcontroller and send them to it
- Acquire images from mobile camera and calculate optical flow
- Process optical flow with the Vector Classifier algorithm and calculate visual odometry from it

- Use the odometry data to maintain a prescribed trajectory or rotate with a requested angle

Autonomous Robot Module is using sDOMO protocol to communicate with RM and with an Engineering Console used to test it during development. The main Messages implemented by ARM are:

- RequestInformationMsg, replied by ARM with either RobotIdentityMsg or ConfigurationMsg based on the content of request
- RequestARMImageMsg replied by ARM with CapturedArmImageMsg
- DynamicBasicCmdMsg used to specify speed and direction and CameraMoveCmdMsg to orient the robot camera, both of them are acknowledged with the same response DynamicBasicInfoMsg that contain the status of both the movement and pan-tilt mechanism
- CameraCalibCmdMsg it is used to perform camera calibration and will result in saving an entry into the Homography matrix to be used later for translating the Optical Flow Vectors in floor coordinates.

Maintaining a trajectory is achieved by 2 independent PID (Proportional Integrative Derivative) controllers one for the angle and one for the speed. The inputs for the controllers are the commands received by ARM from the RM and the results of the Visual Odometry calculation from the Optical Flow, the inputs are subtracted and the controller is trying to minimize the error between the prescribed and actual value.

11 EXPERIMENTS AND RESULTS

After an overview of the System Implementation and its status, some results and experiments are presented and ending highlighting the envisioned follow-up work.

The results of a few components like the sDOMO protocol and MP-Tracker algorithm are presented in details with the conclusion chapters from the published work where they were presented.

11.1 System Implementation



Figure 52 DCS-932 L WiFi Camera overlooking the room

To test the designed architecture and the software we used a room of 20 x 12 feet in

which we installed two DCS-932 L WiFi cameras in the two opposite corners of the room near the ceiling overlooking the room. The two cameras provide opposite images that can be used to build a 3D model of the room after homographic correction.

Data from the camera is being processed by a laptop with a quad core Inter I3 @ 2.3 Ghz running Mageia Linux. On the laptop for each camera we ran a Camera Module (CM) configured to capture data and run MP-Tracker algorithm. Besides running the CM software, the same laptop is also running the House Hub required by sDOMO protocol.

An E5200 dual core desktop at 2.5 Ghz is running Situation Awareness Module (SAM) and the Robot Module (RM) also on Mageia Linux. All the modules are communicating using sDOMO with messages routed by the Hub.

An optional Engineering Console is also running on a separate Laptop to monitor the system.

The software has been developed using C++ programming language (GNU g++ compiler) and OpenCV 2.4 libraries for basic Computer Vision algorithms. The Engineering Console has been developed using QT 5 cross platform GUI library.

The status of the implementation of system is as follows:

1. Camera Module implementation is complete. It acquires data from the cameras, runs MP-Tracker algorithm and broadcast over sDOMO the notifications with the moving objects. The other modules can request full images from CM as needed.
2. Situation Awareness Module exists as a minimal implementation. Right now it receives the notifications from CM's, can request images from CM and using for

each camera a Homography Matrix calculated offline by hand is able to convert the motion vectors in absolute room coordinates. Implementing the algorithms for automatically generating the 3D model of the room and the ability to automatically identify the obstacles in the room it is left to further work due to the huge complexity of this task.

3. Robot Module (the part of robot control software that runs on the base station) is implemented at the minimum required for the rest of the system to work. Right now it is just receiving absolute coordinate tracking information from SAM and attempts to use them to have the robot following a prescribed trajectory by running a PID controller on the trajectory. Also pending is the ability to use image recognition for Visual Landmark based Localization. This will be subject to follow-up work.
4. Autonomous Robot Module (the part of robot control software that runs on the robot itself) is capable to convert the commands from RM into hardware-control signals and drive the robot. It is also able to:
 - capture images
 - calculate optical flow and execute our vector classifier algorithm
 - apply homography transform to translate vectors in room coordinates
 - calculate visual odometry from them
 - It also responds to requested images from other modules
 - Receive commands to orient the camera and work in conjunction with

Engineering Console to calibrate the camera matrix.

There is work in progress to have a PID controller capable to utilize visual odometry to maintain a precise trajectory.

5. The robot hardware is fully implemented, tested and proven to work at nominal parameters.
6. sDOMO House Hub and application libraries are complete and work as expected.

11.2sDOMO Protocol

We successfully demonstrated the implementation of a native speaking sDOMO thermometer based on an Arduino Uno (2 KB SRAM and 32 KB Program Flash) a WS5100 “Ethernet Shield”, TMP-102 I2C temperature sensor and LEDs as placeholders for controls. This implementation proved that the sDOMO it is scalable enough to accommodate small devices as full nodes on the network while sending packets authenticated with SHA1 HMAC. The devices emit a notification either as a reply to a direct message from the GUI either once every second if no control message arrived. The notification message contains both the temperature in milli-Kelvins and the state of the three controls pins, connected in the demo to first 3 LED’s. The control message from the GUI contained a new state of the control pins which turns on or off the LED’s connected to the pins. The demo source code and schematic for the Arduino thermostat is available on the companion website.

The second demo from the companion website implements an elementary file transfer server and client pair which allowed us to perform a set of performance

measurements by comparing the download speed achieved by our demo against the FTP software that came with Mageia Linux (pure-ftp for server and lftp as client).

Table 3 Performance of sDOMO File Transfer vs. FTP Software

| File Size (bytes) | Transfer Speed MB/s | |
|------------------------------|----------------------------|---------------------|
| | <i>sDOMO Demo</i> | <i>FTP Software</i> |
| 11461 | 0.91 | N / A |
| 3785652 | 1.48 | 1.99 |
| 77594624 | 1.48 | 2.10 |

The measurements show that our non-optimized “proof of concept” demo program performed decently while compared with the mature, highly optimized FTP software for Linux despite the fact that the FTP it is a direct transfer between the client and server while in our demo the messages carrying the files were passed between the server and client using the House Hub as intermediary. If we also consider the fact that FTP it is an unsecured connection, vulnerable to a “Man in the Middle” (MiM) attack while all sDOMO packets were signed with SHA1 HMAC and verified at both ends making such an attack impossible, we see that our claims about the efficiency of sDOMO protocol are validated by this measurement.

11.3 Message Dispatcher

The presented framework has been used to rewrite the House Hub from sDOMO project in order to allow scalable processing of multiple devices once the original proof of concept implementation reached its limits. It is being used also in the implementation of House Intelligence Unit from the same project.

The framework implements unique features for mission and safety critical applications being able to offer compile time checking of errors in message registration, enforce the usage of a deadlock avoidance protocol that guarantees the system will not deadlock due to a programming mistake and enforce separation of concerns allowing the implementer to focus on the problem at hand instead of low level mutual-exclusion problems. Because the framework uses handler registration, messages and share objects that can be easily defined at any time MTM-Dispatcher framework it is highly extensible and can be successfully employed in projects that are envisioned to need to scale a lot in the future. The separation of concerns implemented by this framework allows each handler to be written as a standalone piece of code, avoiding coupling that reduces the scalability. This aspect of enforcing stand-alone handlers that are fully defined by their parameters, make the framework highly suitable for test-driven development which is a practice highly regarded in safety critical applications.

To assess the performance of the MTM-Dispatcher a set of tests has been run on a multiprocessor computer having 12 CPU cores. The main question to be answered by the performance testing was if the new multithreaded dispatching frameworks scales well with the number of dispatching threads. The test employed 10 Message handlers all of

them subscribing for the same message from a single message source that has been implemented both as an Active Object without the need to have the Dispatcher lock it during dispatching of the message and respectively as Data Object requiring the Dispatcher to lock it for the duration of dispatching. There were three tests run to assess the performances.

Test #1 had the handlers printing a message then idling for the required amount of time while Test #2 had the handlers performing CPU intensive calculations for the same amount of time. For Test #3 we use the same handler functions as for Test #1 but the Source emitting the message to be delivered to handlers was as of this time a Data Object which required the Dispatcher to lock it therefore preventing other threads to run on the same time. This is a degenerated case that transformed the MTM-Dispatcher behavior in something similar with Reactor framework. For each test we run the dispatcher 32 times with a number of dispatching threads from 1 to 32 with the same workload each time.

As can be seen from the graphic highlighted on Figure 53, for the tests #1 and #2 the amount of time required to terminate the work decreased very fast until all the available CPU's cores (12) has been used by the Dispatcher. After that the curve leveled as expected. There were no noticeable difference between the behavior of I/O and CPU intensive handlers that took the same amount of time to complete.

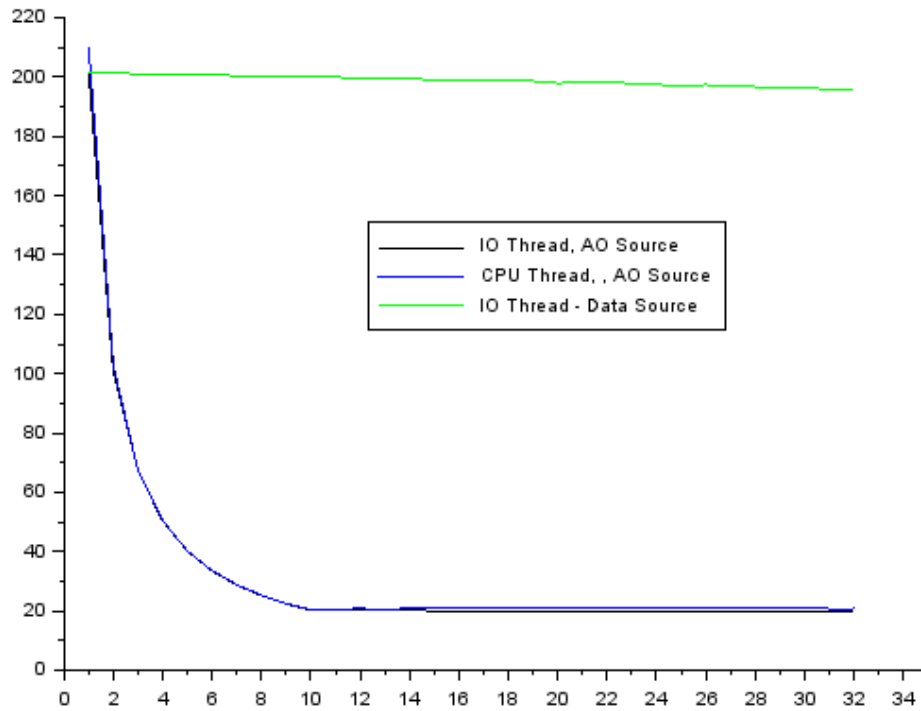


Figure 53 MTM-Dispatcher Performance Graph

By contrast for the Test #3 where we used a Blocking Source forcing all the threads to wait for the current one holding the lock, the curve is almost flat as we would expect also from the Reactor pattern which is using a single dispatcher thread. In theory, the same way as Reactor is using a single thread to perform all the dispatching in the degenerated case of MTM-Dispatcher we would expect the curve to be absolutely flat regardless of the number of threads employed.

However, a closer look at the graph above shows that even for this Test #3 there is an, at the first glance, unexpected very small improvement in performance with increasing number of threads. The explanation for this improvement is that besides the

work required to be performed by the handlers (on which the resources are locked), the Dispatcher itself has to perform some “house-keeping” overhead to manage the messages. While in the case of Reactor pattern this overhead is executed on the same thread as the handler, in the case of MTM-Dispatcher the overhead work performed before the resources are locked and after they are unlocked takes place on a parallel thread to the one currently holding the lock and operating inside the handler. Therefore, even in the absolute worst case scenario when due to resource management our dispatcher degenerate into Reactor behavior, MTM-Dispatcher still outperform Reactor due to the ability to parallelize the overhead work.

The venerable Reactor design pattern [56] has been with us for over 20 years and used to implement countless projects in mission critical applications, but today due to the advancements in C++ language we are able to provide a much better alternative that not only outperforms it in every aspect but also improves the safety and speed of code development by strong enforcement of the separation of concerns.

The only drawback to MTM-Dispatcher is that it requires advanced C++ techniques that are available only in the compilers that implements the C++ 2011 standard and newer, while the Reactor can be implemented in any older dialect of C++ language and even in less evolved languages like Java, C or Ada. There is however a follow-up effort to research Java Reflection technique as a potential means to provide help in porting a “light-weight” version of MTM-Dispatcher to Java. But as of this moment, C++11, C++14 and the forthcoming C++17 are the only languages in which MTM-Dispatcher can be implemented and even for the foreseeable future they can remain the only languages that can be used to write a complete (“heavy-weight”)

implementation of this framework.

As far as interpreted languages like Python, while they will never be able to support compile time checking of correctness (due to inherent nature of weak-typed interpreted language) they shall have relative little problems to implement deferred calling of a functional-object with a tuple, therefore a “light-weight” version of the dispatcher should be implementable in these languages too.

11.4MP-Tracker

The tests run showed our method to be able to track two RC vehicles and a person walking inside a room using an IP camera mounted at the corner of a room near the ceiling. The camera provided 640x480 JPEG images accessible via HTTP with an average frame rate of 5 frames a second.

During experiments, it was noticed that our optimization worked as expected. While tracking only one or two RC vehicles the LK matching algorithm is very rarely called, for over 92% of the time the tracking has been performed exclusively with FH and Area matching alone. Even small occlusions are being resolved without the need to invoke LK in over half of the instances.

For example, while tracking a single vehicle alone for a duration of about 800 frames a single invocation of LK matching has been performed when the RC car took a semi-circle at high speed. With two vehicles LK is being invoked mostly when vehicles EBR intersects.

Bringing a person in the scene changes the things radically due to the much larger

size of the person and more fluid changes in shape. Due to severe occlusions LK is being invoked around every next frame when the persons walk in front or in the back of the other targets.

The experiments showed that it is much easier to track vehicles than persons. Vehicle tracking has been showed to recover very easily from occlusions, while tracking of a person occluded by fixed objects often fails when the person came back into the view, creating spikes in CPU usage. The low frame rate provided by the IP camera is another source of problems for tracking the person. Often the person is able to turn fast enough such that into a frame the view is sideways while we have a front view in the next frame and LK matching fails to find enough corresponding points. A better IP camera capable of higher frame rates is expected to allow improvements in human tracking.

To perform performance comparison, a video 1100 frames @ 640x480 has been recorded to a file allowing us to run the algorithms without any network latency into a repeatable manner. We run the measurements on a Pentium E5200 @ 2.50GHz and compared MP-Tracker performance against Raw Lucas-Kanade Optical Flow with 400 Shi-Tomasi points distributed across the image.

The experiment shows that on the average MP-Tracker is about three times more efficient than Raw Lucas-Kanade Optical Flow calculation across the whole image. The spikes observed in MP-Tracker coincide with the person walking relatively close to the camera occluding both vehicles. In that case MP-Tracker lunches the MSER segmentation and restarts LK matching afterward to resolve remaining ambiguities.

Table 4 MP-Tracker Performance Comparison Summary

| | MP-Tracker | Raw LK OF |
|-----|------------|-----------|
| Min | 33 ms | 114 ms |
| Avg | 39.4 ms | 121.5 ms |
| Max | 289 ms | 171 ms |

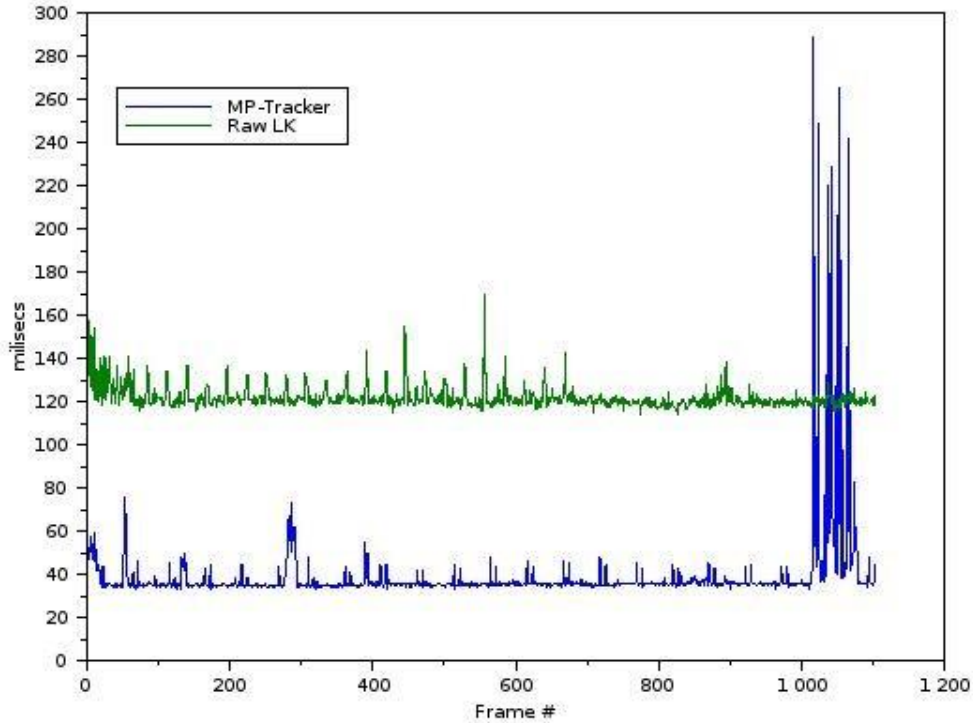


Figure 54 MP-Tracker Performance Measurement

Active research is being done to solve this set of problems with person tracking by exploring contour tracking and hierarchical region grouping an idea inspired from [7].

An alternative idea that is in research as of this moment is the ability to perform Target

merging when two or more Targets exhibit trajectories that can be interpreted with high confidence as a perspective projection of parallel tracks. In the main program flow, as shown in Fig. 2, this is referred as the Splicing part in Target Breaking and Splicing.

However, the fact that the average time for processing is below 50ms allows us to provide real-time tracking information for multiple objects while running multiple trackers connected to separate cameras on the same multi-core computer on the Base Station.

11.5 Localization Experiment Using Pre-existing Landmarks

The first experiment to be performed was localization using vision based detection of the already existing landmarks inside the house, (furniture and fixtures) using optical flow to attempt to identify a few features in the rooms that were selected and mapped up-front. The attempt used an algorithm derived from the standard application of optical flow to detect objects in two slightly moved images:

- Calculating Shi-Tomasi corners in the source image (where landmarks were mapped)
- Choosing only those that lies within the boundaries of the rectangle defining the landmark
- Use Lucas-Kanade algorithm to find the corresponding points in the new image
- Use the detected corresponding points to calculate the corners of the landmark in the new image

Unfortunately, the experiment failed to provide consistent recognition of the

landmarks with a repeatability good enough to perform localization, where for each position we needed four landmarks as explained in the localization chapter above. As can also be seen from the picture, the algorithm failed to reliably match three of the landmarks with a probability high enough to declare a landmark match.

Only a single landmark (the electric guitar) was perfectly identified over and over again in all the performed tests. The three drawer dressers were exactly at the opposite end, never being correctly matched by the algorithm. The two shelves were sometimes matched but not consistently enough to be reliably used for localization.

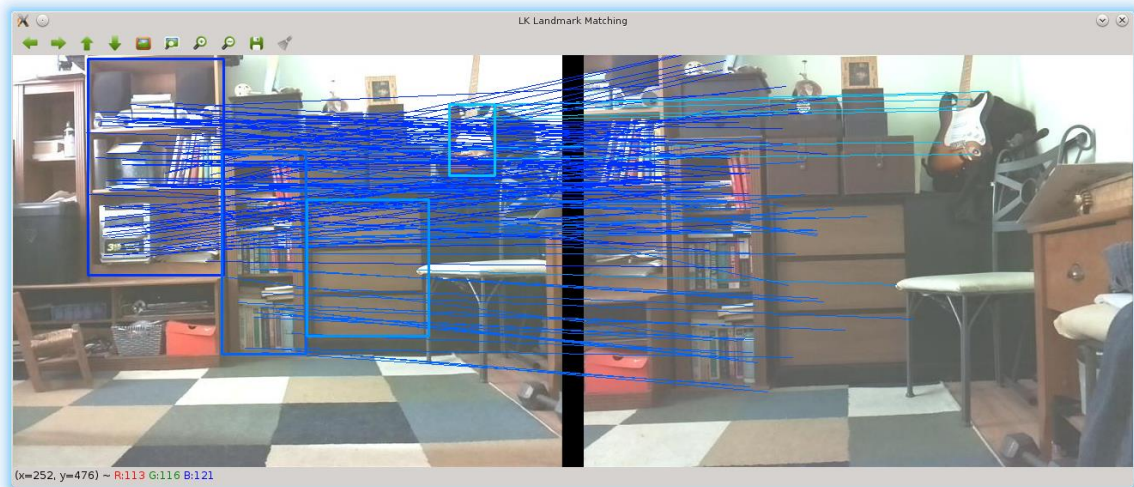


Figure 55 Lucas Kanade Landmark identification

A possible explanation for this results could be the fact that significant corner information were lost during data compression. Because LK matching is very CPU intensive, the work has not been performed on the embedded computer board on the robot itself but it was implemented on one of Base Station computers by the so called Robot Module (RM). The images captured by the robot camera had to be uploaded into

RM, and this was done using JPEG compression. JPEG compression divided the image into small cells and performed approximations on the content of the cells, these approximations introduced fake artefacts that the Shi-Tomasi corner detector could mistake for a true corner in the image. The dresser had relatively uniform colors making them get smoothed out of way to much by the compression algorithm that no feature of interest had been left out. The two book shelves having horizontal shelves and vertical books provided a lot of edges parallel with the JPEG cells allowing false corners to appear. On the other side, the electric guitar with its fluid forms gave a hard time to the JPEG optimizer which had to preserve most of the original artefacts of the image. This preserved good corners at the price of less efficient compression on that part of the image. The preservation of the original corners provided good ground for Shi-Tomasi corner detector allowing the guitar to be correctly identified in experiment over experiment.

An interesting lesson that was learned from this experiment was that the reliability of the algorithms using corners to identify objects is strongly affected by image compression and new algorithms have to be developed for this situation. This will definitely be an interesting line of follow-up research work.

11.6 Localization Experiment with Active Artificial Landmarks

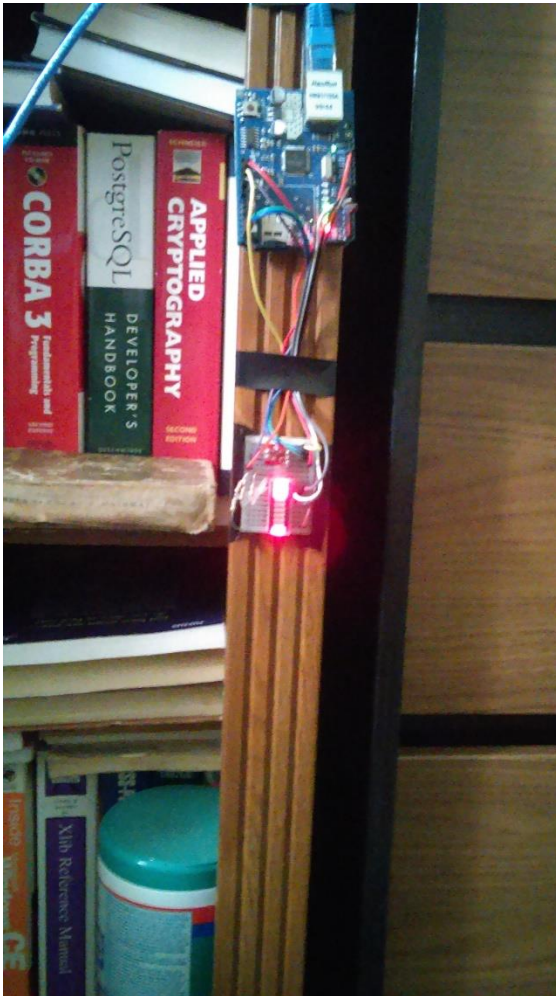


Figure 56 Active Landmark

After the hard time with previous localization attempts, an alternative idea for landmark localization has been to take advantage of the integrated domotic network we developed to allow the robot to control artificial landmarks that act as beacons.

The active artificial landmarks are devices speaking sDOMO protocol allowing the robot software to control their LED by turning them on and off. The active landmarks are best placed at the same height from the floor as the robot camera for optimal viewing

condition.

For this experiment, we reused the same hardware as in the sDOMO speaking thermostat that we developed for our paper [62] presenting sDOMO protocol at the IEEE TEPRA conference. The thermostat have four GPIO pins connected to red LEDs as a placeholder for relays that can be used to control the A/C units. We just tapped the thermostat on the furniture and therefore transformed it in an active landmark by adding an sDOMO Hub rule allowing the robot full control rights of the device.

The procedure for detecting the landmark consists in using the robot camera to take a series the images with a particular landmark as follow:

1. Turn landmark OFF and take first image $OFF1 = \text{RedChannel}(\text{image})$
2. Turn landmark ON and take first image $ON1 = \text{RedChannel}(\text{image})$
3. Turn landmark OFF and take first image $OFF2 = \text{RedChannel}(\text{image})$
4. Turn landmark ON and take first image $ON2 = \text{RedChannel}(\text{image})$
5. $DIFF1 = \text{BinaryThreshold}(\text{abs}(ON1 - OFF1))$
6. $DIFF2 = \text{BinaryThreshold}(\text{abs}(OFF2 - ON1))$
7. $DIFF3 = \text{BinaryThreshold}(\text{abs}(ON2 - OFF2))$
8. $\text{Result} = \text{BinaryThreshold}((DIFF1 - DIFF2) \& DIFF3)$

The result of this processing sequence has been found to be very consistent, a single dot on the result image at the exact location of the landmark. In case that due to movement or due to other blinking lights more than one landmark is detected, the robot

can repeat the measurement with a slight different timing to avoid stumbling on the same periodic blinking lights.

The imprecision in identifying the landmark as pixel in the image is 1 centimeter at a distance of 2.7 meters from the robot, this corresponds with an imprecision angle of 0.21 degree. However the result of measured angle is subject to camera calibration, and the 0.125 imprecision in pixel size, therefore the final result may have a higher imprecision varying from measurement to measurement.

The reliability of the active landmark detection is a very nice result, validating the whole concept of using an integrated domotic system where the robot is just one component of the system. The ability of the robot to control the equipment from the house as needed truly gave it a whole new set of capabilities.

It is also important to notice that the processing described above uses only basic computer vision operations which are very efficient. As opposed to the Lucas-Kanade which had to be executed on the base station due to high CPU requirements, this set of operations has been executed on the embedded computer board of the robot using a single core of the 4 that are available on RPI2 and performed at a frame rate of 1.5 frames/second. It is expected that with future optimization of using all 4 cores of the CPU this algorithm can perform in real-time on RPI2.

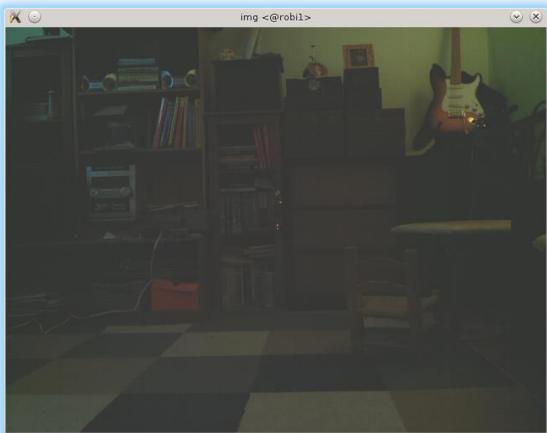


Figure 57 Original Image

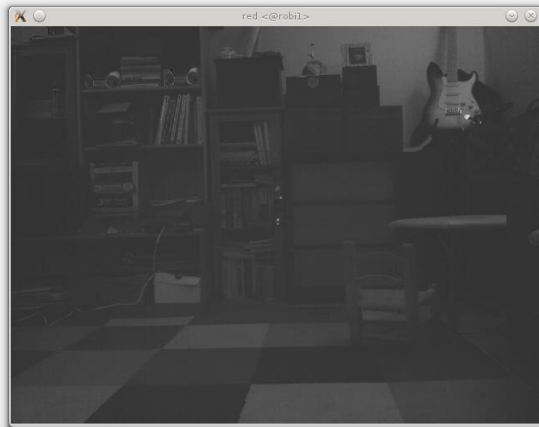


Figure 58 Red Channel

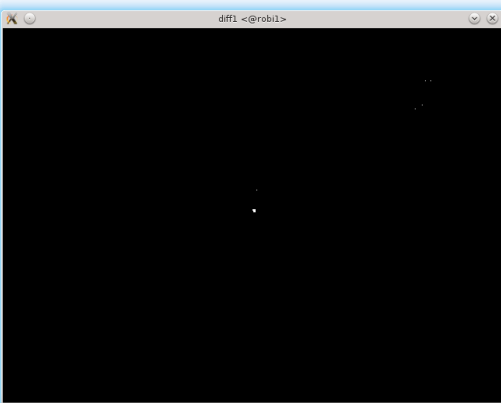


Figure 59 Diff1

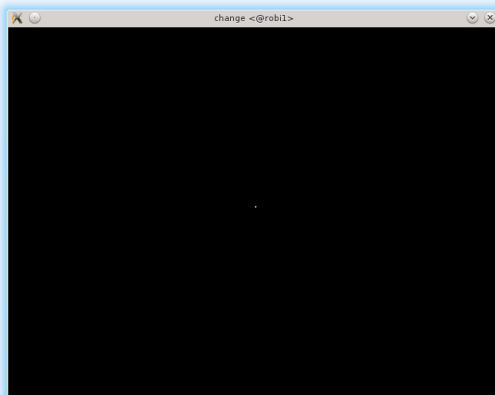


Figure 60 Result

11.7 Static Landmark Localization Experiment

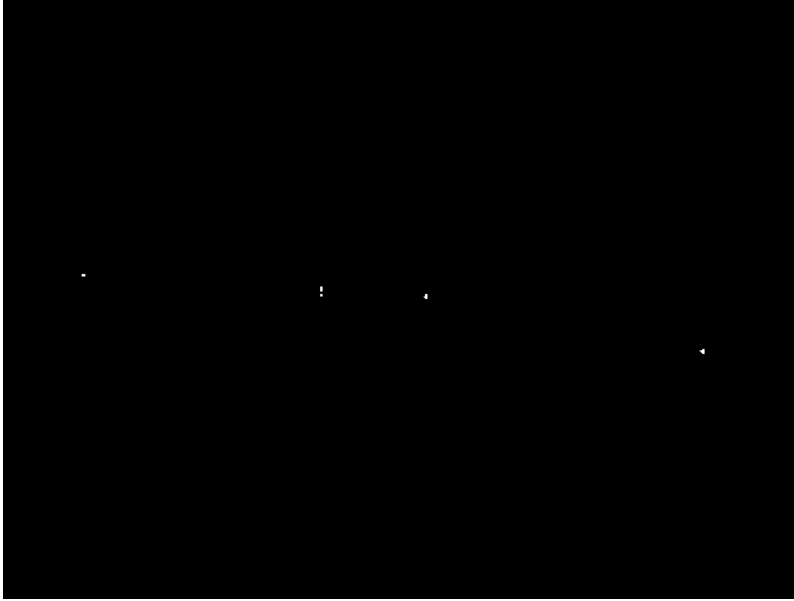


Figure 61 Four Active Landmarks Detected

The focus of this experiment has been to use the active landmarks developed above to perform localization of the robot inside the room by processing a single image. To allow room for error we used four active landmarks that were detected very clearly as seen from figure 61 (right).

The landmarks were detected at the horizontal pixel numbers of 339, 255, 64 and 560 corresponding in order to landmarks id as L1, L2, L3 and L4.

The landmark numbering is based on the order the robot did the reading and has no relationships with their physical localization of them. The position information about the landmarks are presented in the table below. The distances are in centimeters and the corner (0, 0) has been selected to be a right hand Cartesian system. For the scope of this

static localization test, the Robot has been located at the fixed position of (59.5, 116.0) centimeters from the room corner selected as the origin of the Cartesian axes.

Table 5 Results of First 3 Landmark Localization Experiment

| Landmark ID | Position (x,y) centimeters | Horizontal Pixel |
|--------------------|-----------------------------------|-------------------------|
| L1 | (319.0, 128.0) | 339 |
| L2 | (319.0,162.0) | 255 |
| L3 | (323.0,243.0) | 64 |
| L4 | (251.0, 60.0) | 560 |

Because for landmark based localization accurate measurements of the angles between landmarks are necessary, the first step has been the horizontal calibration of the camera. The purpose of the calibration was to be able to provide a mapping that translates a pixel number into an angle value in respect to the camera axis.

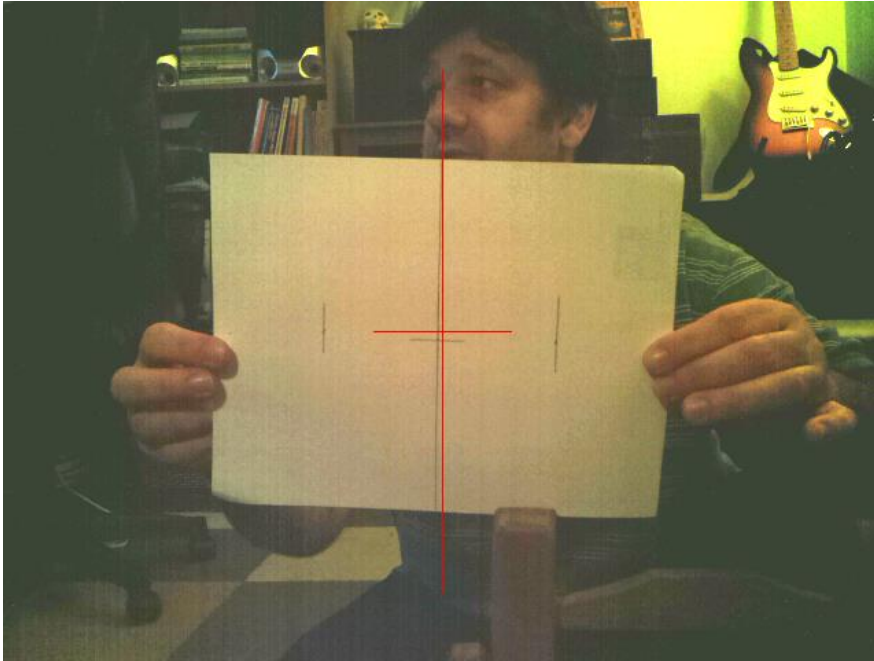


Figure 62 First Camera Calibration

The calibration has been performed holding a piece of paper 28 centimeters long with 2 marks at 7 cm from the center at a distance of 50 centimeters from the camera and taking a picture of it. Using the **arctangent** function the angles of the marks and end of the paper has been calculated.

Having identified the horizontal pixel locations of the marks and that of the end of the paper, a mapping function between the pixel values and the angle of sight has been implemented.

After taking a set of measurements the eight potential points were calculated and those that resolved for all the criteria were checked against the room geometry, and the points that were not located inside the room were also discarded. Each pair of marks had

associated a reflecting index 0 being the point calculated originally with the formulae derived, while 1 denoted the calculated reflection of that point in the respect to the segment.

For the first measurement the Landmarks L1, L3 and L4 had been chosen for localization and the calculation of angles from the pixel locations gave the values: $\theta = 26.182$ and $\psi = 20.061$.

After running the localization algorithm from the 8 possible equations 5 of them resolved with valid solutions but one was eliminated as being out of the room area.

The values obtained from the first test were:

Table 6 Results of Second 3 Landmark Localization Experiment

| Reflection ID | Calculated Location | Error |
|---------------|---------------------|---------|
| 0 0 1 | (320.801 127.402) | 261.999 |
| 0 1 0 | (321.680, 244.297) | 292.292 |
| 1 0 0 | (245.095, 59.822) | 194.342 |
| 1 1 1 | (285.957, 291.032) | 286.571 |

Which was nowhere near the actual location of the robot, rendering the results totally useless.

Running the test with L2, L3 and L4 provided the same type of results way off, even if apparently marginally better. In this test 3 of the 8 possible solutions resolved but

one has been automatically discarded by the program because its answer was located outside of the room boundaries. The calculated angles for this test were: $\theta = 18.250$ $\psi = 27.993$ and the two passing solutions:

Table 7 Results of Third 3 Landmark Localization Experiment

| Reflection ID | Calculated Location | Error |
|---------------|---------------------|---------|
| 0 0 1 | (326.604, 158.799) | 270.956 |
| 1 1 1 | (70.405, 229.509) | 114.076 |

The first suspect in this embarrassing result was the possibility that I've made some math mistakes while deriving the localization equations. After reviewing them and not finding anything wrong a simulation test has been attempted in an effort to prove or disprove the validity of equations.

The first tuple landmarks (L1, L3, L4) used in the previous experiment and the actual position of the robot has been plotted in GeoGebra which calculated the expected angles as $\theta = 23.12$ $\psi = 18.99$.

Plugging in these values in the localization algorithm in lieu of the values detected by the computer vision algorithm, the calculated position for the index **1 1 1** was:

(60.047, 115.857) located at a distance of barely 0.56 cm from the real location.

The same result has been replicated using the second set of simulated landmarks (L2, L3, L4) for which the GeoGebra calculated values were $\theta = 15.70$ $\psi = 26.41$ resulting in localization with only 1.1 cm error in distance.

The localization within millimeters into the simulation program showed that the problem is not with the correctness of the localization equations which provided practically perfect localization within the error generated by the truncation of the angles with 2 decimal digits in GeoGebra display.

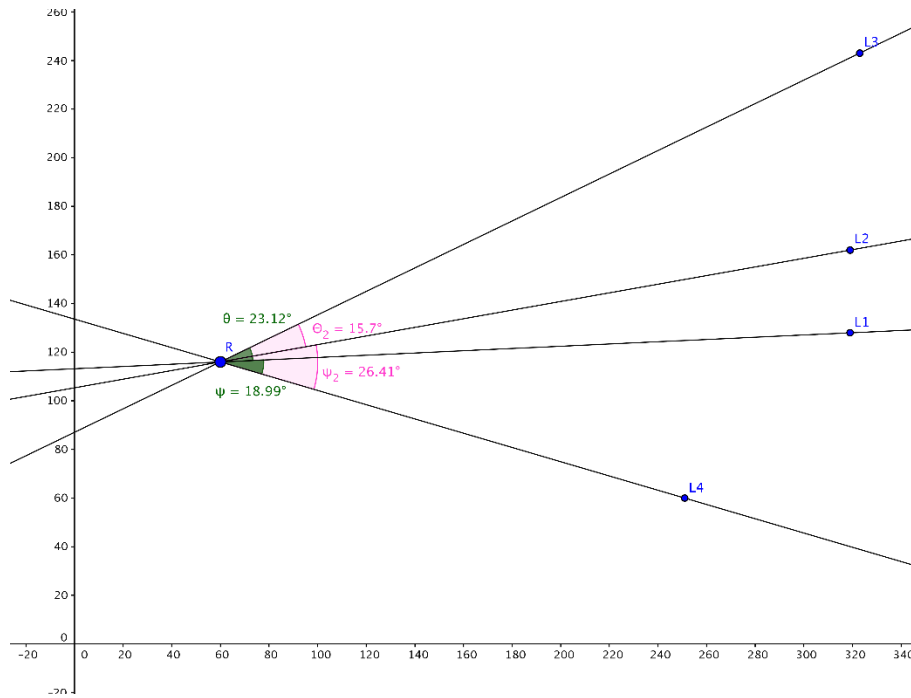


Figure 63 GeoGebra Simulation of Localization Experiments

The next focus point in the quest for an explanation for these results was the calibration process. It appeared that calibrating the camera while holding a sheet of paper in hands may not have been accurate enough. There are some differences (3.06 degrees in theta and 1.07 degrees in psi) in the calculated angles and the measured ones.

However the fact that these differences are small also means that besides improving calibration we have to take a look at the stability of the localization equations under noise generated by real-world measurements.

A second calibration task has been attempted by placing 2 chairs at 174 cm in from on the camera, one exactly on the median pixel and another at 87 cm distance parallel with the camera sensor plane, then identifying the pixels on the image. Special care was taken to distance measurement, placement of the objects and pixel identification. To avoid any vibrations the servo-motors of the robot were turned off and the orientation of the camera set by hand.

After the higher precision calibration, the set of landmarks L2, L3, L4 showed a discernible improvement for index **1 1 1**, however nowhere near as being usable for real-life robot navigation in domestic environments.

Table 8 Results of Fourth 3 Landmark Localization Experiment

| Reflection ID | Calculated Location | Error |
|---------------|---------------------|---------|
| 0 0 1 | (324.508, 159.497) | 268.998 |
| 1 1 1 | (41.925, 181.477) | 67.680 |

Unfortunately, no discernible improvements have been seen in localization with the set of landmarks L1, L3, L4 where the best improvement has been a drop in **1 1 1** error from 286 cm to 208 cm, still ridiculously off.

The failure to see any substantial improvement with the best calibration I was able to achieve prompted an investigation into the stability of the localization equations under noise generated by the errors in real life angle measurement.

Analysis of the stability of the localization equations

Looking at the lack of improvement in the localization accuracy after the second calibration it became clear that an analysis of the stability of the localization equations is required. The calculated angles after the second calibration are very close to the theoretical values that provided perfect spot-on localization within the simulated environment.

For the set (L1, L3, L4) the real measurements were $\theta = 23.565^\circ$ $\psi = 18.938^\circ$ and the theoretical values of 23.12° $\psi = 18.99^\circ$ differs by a totally insignificant 0.44 and 0.05 degrees respectively while the calculated localization is completely out of any acceptable ballpark.

As far as the angle measurement is concerned, these are the best measurements that are expected to ever get in a real life situation with a 640x480 pixel camera mounted on top of a Pan-Tilt mechanism powered by two hobby-grade servo motors. Due to vibration of the motors a 0.5 to up to 1 degree error in measurement is not an outrageous occurrence so the stability of the equations to measurement noise is essential for practical purpose.

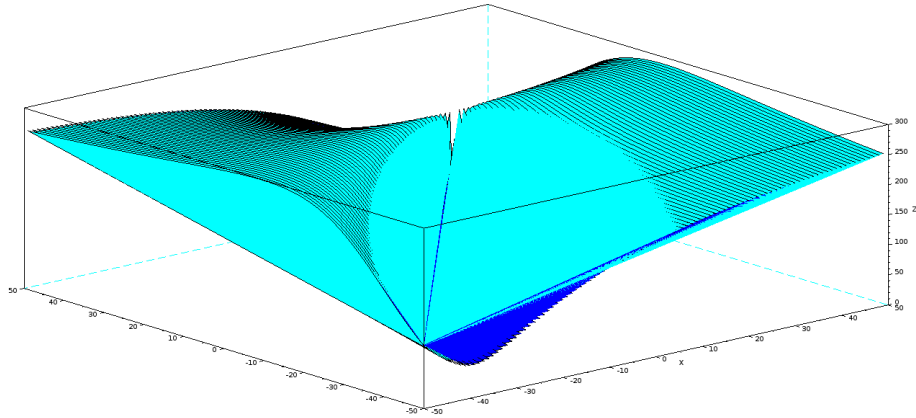


Figure 64 3D Stability Plot of the Static Landmark Based Equations in Respect to the Landmark Set (L1, L3, L4). The axis for theta and psi are in 1/10 of a degree.

To plot the graph of the stability of the equation to the measurement noise we've started with the position of the first set of landmarks (L1, L3, L4) and the theoretical values that achieve perfect localization. We wrote a simulation software that injects small changes from -5 to +5 degrees in both theta and psi with an increment of a 1/100 degrees and plotted in 3D the distance between the actual location and the calculated one. The large error is very easy to be seen in this plot while the sudden cliff in the middle represents the sudden drop to zero error for the exact values. Unfortunately the strange surface makes things very hard to see unless you can rotate the plot in real time.

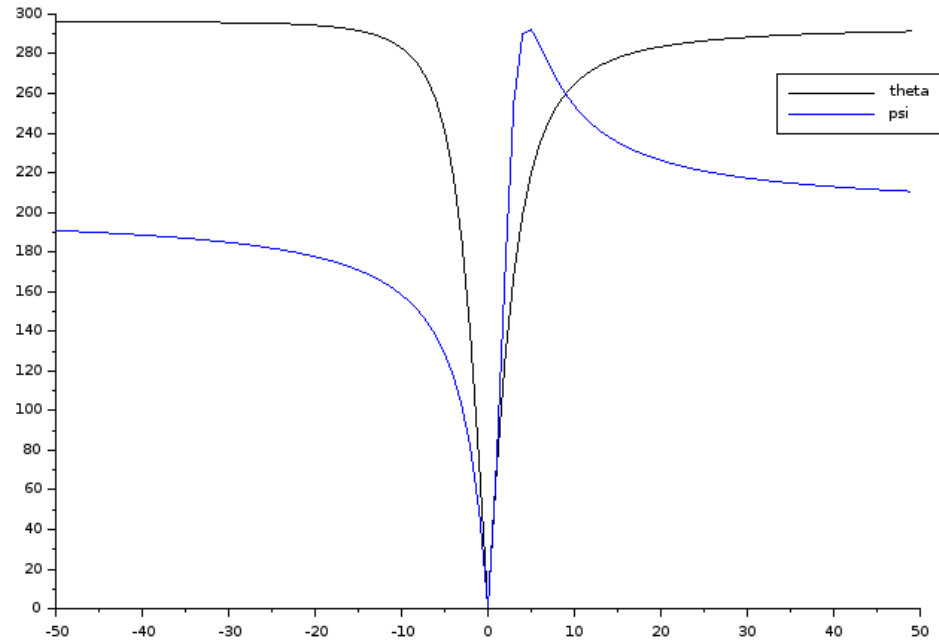


Figure 65 Graph of the partial derivatives of the 3D stability plot at the center. The X axis is in 1/10 of a degree while Y axis represent the error in localization in centimeters.

A more easily read plot can be obtained by looking the variation of each of the angle individually while keeping the other one fixed at the theoretical value resulted from simulation.

In this plot the values for theta and respective psi has been represented in 1/10 degree increments around the theoretical value while keeping the other angle fixed at the theoretical value. The variation to be studied went from -5 to +5 degrees around the base values (-50 to +50 tenths of a degree in the plot).

It is easy to see the very steep increase in the error even for the very small angles. For example an almost insignificant +0.5 degree in psi over the theoretical value will

make the error to reach the top (3 meters in error) even if theta stays at the exact theoretical value for perfect localization. A combination of small variation in theta and psi simultaneously will very easily throw the results outside any useful zone.

In conclusion, the derived analytical equations for 3 landmark based static localization while they are a very elegant theoretical construct have no practical engineering application due to their extremely high sensitivity to measurement errors. They can still be used as a teaching tool to show a nice derivation, however unless we find ways to improve their stability they have no practical application in robotics as they stand today.

11.8 Running Fix Localization Experiment

To test the equations derived for the Running Fix two experiments were performed. On both of them the robot started at the location of (58 cm, 115.5 cm) in room coordinates and after first landmark detection moved 50 cm along the X axis of the room to the second position where another detection of the target took place. From the target horizontal pixel coordinate an angle measurement was taken.

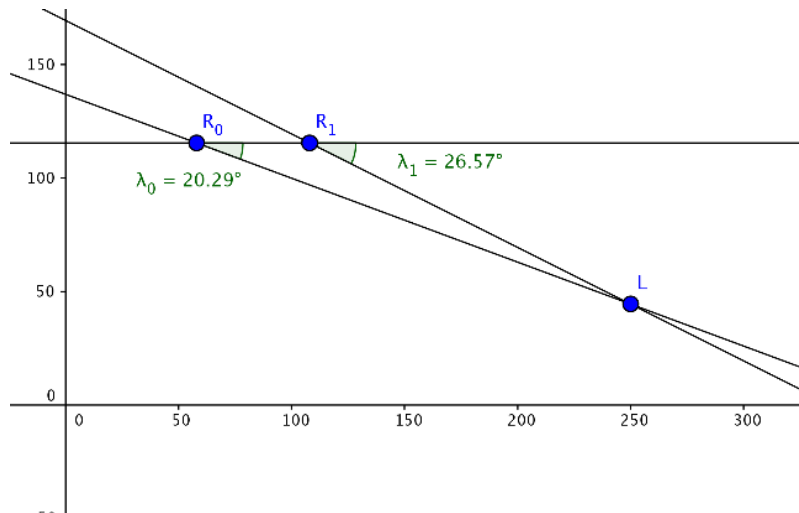


Figure 66 Simulation of first Running Fix test

In the first experiment an active landmark was placed at (250 cm, 44.5 cm) in room coordinates and the robot camera was pointing forward along the X. A graphic description of the experiment can be seen in the drawing aside.

The landmark has been detected at pixel horizontal position of 550 resulting on angle $\lambda_0 = -19.71$ degrees.

The robot moved 50 centimeters along X axis and then the landmark has been detected again at pixel 620 corresponding to $\lambda_1 = -25.70$ degrees.

Applying the Running fix equations the calculated position of the robot resulted at (54.612, 114.495) therefore the error in localization from the real position was 3.53 centimeters.

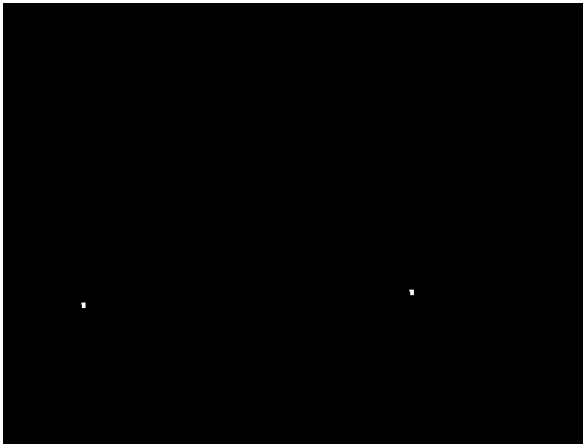


Figure 67 Landmarks Detected in Side View Experiment

For the second experiment, we rotated the robot camera all the way to the right (at -90 degrees) and the active landmark has been placed in the camera view at (91, 27.5) centimeters in room coordinates. Starting at the same position (58, 115.5) and advancing 50 cm the robot detected the landmarks at the horizontal 88 and respectively 446 corresponding

to angles 19.880 and -10.797 from the camera axis. These two measurements translated in $\lambda_0 = -70.119$ and $\lambda_1 = -100.797$ from the robot axis. The results from Running Fix algorithm calculations determined the robot to be located at (58.264, 118.025) a localization error of 2.54 cm from the real position. The fact that the landmark was closer resulted in a much wider angle in the test with the camera oriented sideways and this had a direct effect on improving the accuracy.

The two experiments not only validated that our derivation of the equations for the localization by the Running Fix method are correct but also showed them to be directly usable for practical applications. The stability analysis shows clearly way.

The study of the stability of the solution to the localization problem by Running Fix method started with sketch for the first test, where the camera was pointing forward. For the angles, the effect of the error in measurement has been studied for ± 5 degrees

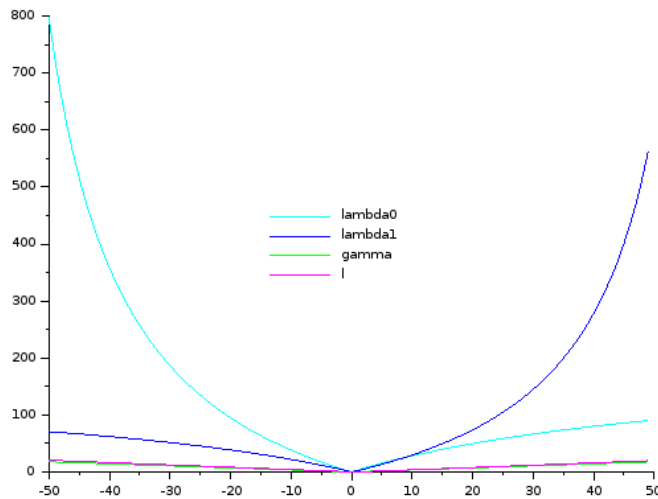


Figure 68 Partial Derivative Stability Plot for Running Fix equations

and for the length of movement a 10% of the displacement (i.e. 5 cm error) has been scaled to cover the X axis.

In the plot λ_0 and λ_1 are the angles under which the robot saw the landmark at original and respective second location, while γ is the direction of movement of the robot and l is the length of the movement. The values for angles has been plotted from ± 5 degrees in error while for the length the -50 to +50 of the X axis in plot is scaled to cover an error in measurement of $\pm 10\%$ of the motion length.

As can be seen from the stability plot, the error for the measurement for the two λ angles is relatively wide, that means small errors in measurement being tolerable. Both errors in the direction of movement (γ) and measurement of distance (l) are even better tolerated by the algorithm, as their slope is very small.

However, there is also an unexpected surprise on this experiment too, albeit a very pleasant one. Based on the partial derivative stability plot we would expect the error for the first localization experiment to be in excess of 30 cm instead of the smaller 3.53 cm we obtained in the real life experiment. We expected this because of the difference in angles between the simulated experiment calculated at $(-20.29, -26.57)$ and the real life measurements taken as $(-19.71, -25.70)$ is quite significant: $(0.58, 0.87)$ degrees. This discrepancy of an order of magnitude between the expected result and the almost perfect localization in real-life prompted a deeper investigation in the stability of the Running Fix equations.

After countless tests in SciLab, the mystery has been solved once plotted the absolute values of the error for the components L_u and L_v function of the error of measurement for λ_0 (labeled x) and error in λ_1 (labeled y in plot). The explanation

consists on the nature of the measurement errors. A measurement error (E) can be described as being composed by two components: Random (ρ) error Gaussian with zero mean and a Bias error (β).

$$E = \rho + \beta$$

The Bias error is a component of the error that is constant or correlated with the measured value, while the Random error has a normal distribution with a mean value of 0. In practice, is usually difficult to eliminate the Bias component unless you have some prior knowledge about it, however the formulae for Lv and especially Lu are special, in the sense that they self-correct most of the Bias error.

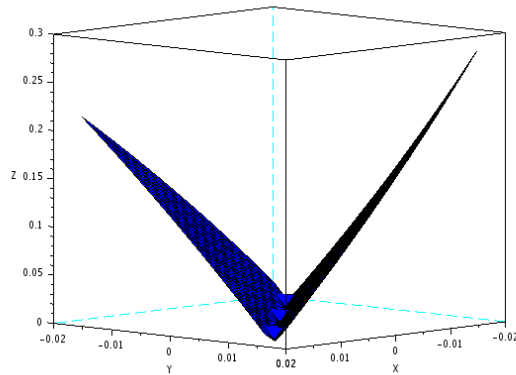


Figure 69 Abs(Lu(l0+x,l1+y)-Lu(l0,l1))

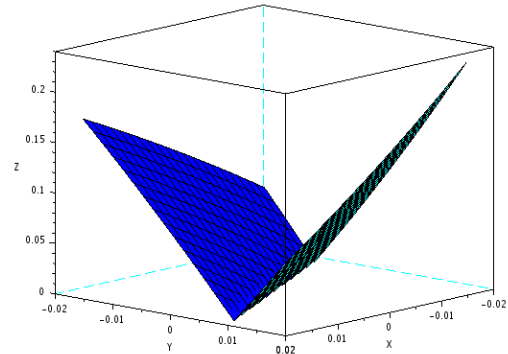


Figure 70 Abs(Lv(l0+x,l1+y)-Lv(l0,l1))

In the 3D plots above is displayed the error generated by the noise (x,y) added to the values lambda0 and respective lambda1 for both Lu and Lv formulae which are the precursors of the Running Fix localization. As can be seen from the plot, the error almost

collapses to zero for values that are very close to the diagonal, i.e. if the error in measurement for λ_0 and λ_1 are dominated by a constant Bias component (therefore for values of $x \approx y$) the error in L_u and L_v is practically almost zero.

In our case it is most likely that the Bias error in angle measurement has been introduced by our less than perfect camera calibration, however the calibration error being a constant it has the same in readings of λ_0 and λ_1 therefore it was almost canceled by the stability of the equations. We expected the error to be over 30 cm by looking at the partial derivative plot, but in that plot the value of one angle error has been kept at 0 and plotted for the error in measurement of the other angle. Because of the nature of it, the self-correcting error for the same values for both x and y is impossible to be seen in partial derivative plot. Only when looking at the 3D plot, the self-correcting nature of these equations became obvious.

As a direct result of self-correcting for Bias errors, with Running Fix method the most damaging errors are the Random errors, however the impact of those errors can be reduced by multiple sampling and averaging or even better with the use of a Kalman filter which is the optimal Gaussian estimator.

The results of this experiment along with the stability plots validates the Running Fix equations as being an outstanding localization method with direct practical applications in robot (and not only robot) navigation.

11.9 Localization Experiment with Fixed Camera

In this experiment participated by all the components already developed for the system.

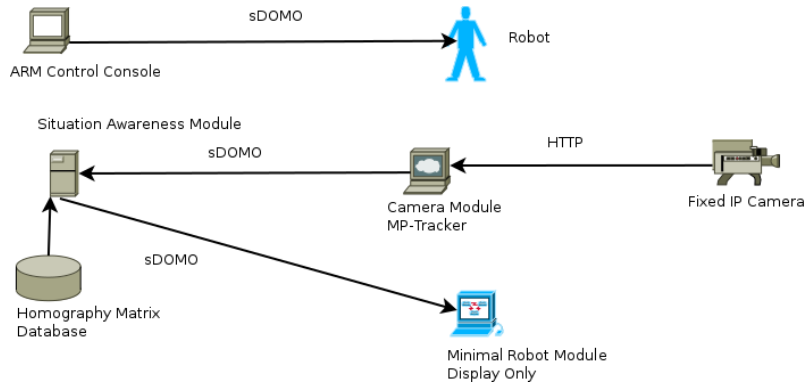


Figure 71 Experiment Design for Localization with External Camera

The embedded robot software Autonomous Robot Module (ARM) was disconnected from the fixed Robot Module (RM) and controlled by the ARM Console, practically transforming the robot into a remote controlled vehicle. The Robot Module has been used only as a print-out on the debug window the location of the robot. The Camera Module (CM) was acquiring images from one of the fixed IP camera using HTTP and run MP-Tracker algorithm on the sequence of the images. The information about tracked targets were broadcasted over sDOMO protocol.

Situation Awareness Module (SAM) received the Tracking Information from CM, loaded the Homography matrix for this particular CM and translated the tracking information from image coordinates in Floor Coordinates. For display purposes, we selected the coordinate system for the floor to have the same structure as the one used by display and allocating each pixel to half an inch. This made it an easy task to display the results

of the images transformed by homography, however we had to do an extra coordinate translation in the Robot Module to display the results in the same system as for previous experiments. For the purpose of this experiment the Robot Module (RM) has been used exclusively to print out the location as a debugging statement.

The Robot Control Console has been used exclusively to give to the robot a minuscule move, in order to get in the “attention” of the MP-Tracker algorithm that runs on CM. This small move kick-started the whole process of localization via MPTracker.

Prior to the experiment we had to calculate the Homography matrix for this camera. To achieve this we selected 4 points in the image: one being the foot of the shelf in the left, two corners of the middle carpet and the corner of the little table seen at the bottom of the image.

Table 9 Data Input for Calculation of Homography Matrix

| Point | Image Pixel | Floor Coordinates | Cartesian (cm) |
|-------|-------------|-------------------|-------------------|
| 1 | (262,208) | (32, 151.0) | (191.77, 40.64) |
| 2 | (353, 237) | (111.0, 180) | (228.6, 140.97) |
| 3 | (439, 389) | (206, 326.4) | (414.528, 261.62) |
| 4 | (99, 441) | (146, 458) | (581.66, 185.42) |

From the calibration quad above the homography matrix has been calculated as:

$$A = \begin{pmatrix} -1.820696309215 & -4.454758771421 & 1314.877616567 \\ 0.616170273279 & -8.945472660604 & 1280.50511505 \\ -0.0028807012106 & -0.0145106433222 & 1 \end{pmatrix}$$

Having the robot take a small move, ending up at (534.5, 170) centimeters in Cartesian coordinates triggered a detection at (544,182) cm therefore an error of 15.30 cm in localization.

An error in calibration was noticed in this experiment, we selected the corner of the table as a point corner which is not on the floor but at a height of 43 cm. Another point was selected on the floor at (520.7, 191.77) cm corresponding to pixel (206, 456) and after recalculating the new homography matrix the robot located at (533, 170.5) cm has been detected at (524.5, 177.5) cm, resulting a distance error of 11 cm. It is also noticeable from the images after the projection that the result of second tests looks a little bit less distorted than the first projection with the incorrect matrix.

The fact that we still have an error of 11 centimeters can be explained due to two factors. The first it is the fragmentation of motion blobs if the color of a certain part of the background is the same as the foreground image. If this situation happens, the SAM is sending to the RM all the fragmented targets detected and RM will eventually choose the center of the largest one as the robot position. In reality the center of the robot in the image is a bit different, shifted toward the other blobs.

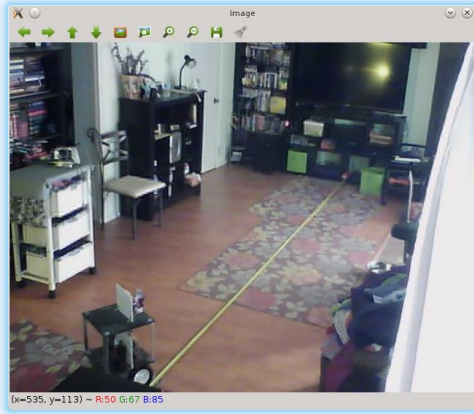


Figure 72 Test1

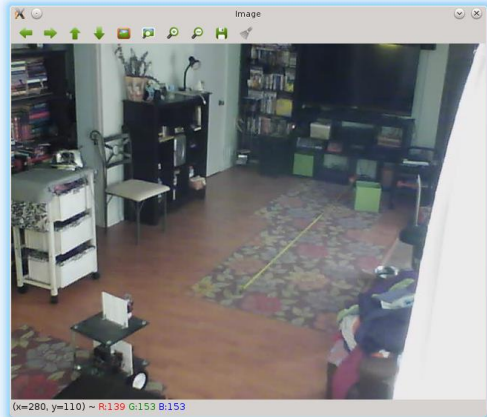


Figure 73 Test 2 with corrected calibration



Figure 74 Test 1 Homographic projection

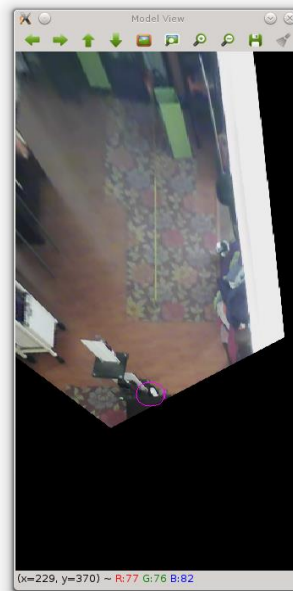


Figure 75 Test 2 Homographic projection

The second has to do with the perspective projection error. The center of the object (C) and the center of the observed image (O) does not coincide due to the perspective projection. There will always be an error between them.

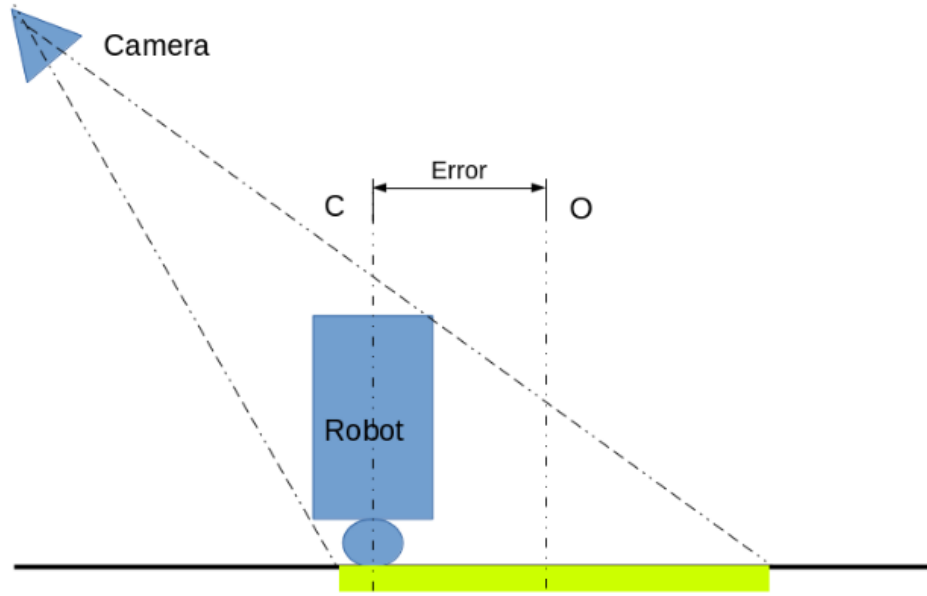


Figure 76 Center of the robot versus center of the perceived image

An idea to correct for this error, the solution is a full 3D modeling on the SAM. The RM then can inform SAM that he is at this target and upload in SAM a 3D model of itself. Having this information SAM can back-project the model to match the observed image, then find the exact localization of the object. However, this complex 3D modeling has not been implemented in the software written for this dissertation.

11.10 Optical Flow Odometry Experiment

In this test, we attempted to use the optical flow captured from the robot main camera to provide odometry calculations. As presented earlier in the paper in Chapter 9.3,

to achieve odometry calculations we captured images with a camera looking at the floor, calculated Lucas-Kanade sparse optical flow and used the detected matches in our vector classifier algorithm which will eliminate outliers. The resulting filtered vectors will undergo a homography transformation to translate the in floor coordinates then we average the transformed vectors. The estimated speed and direction is then calculated as the magnitude and orientation of the average vector.

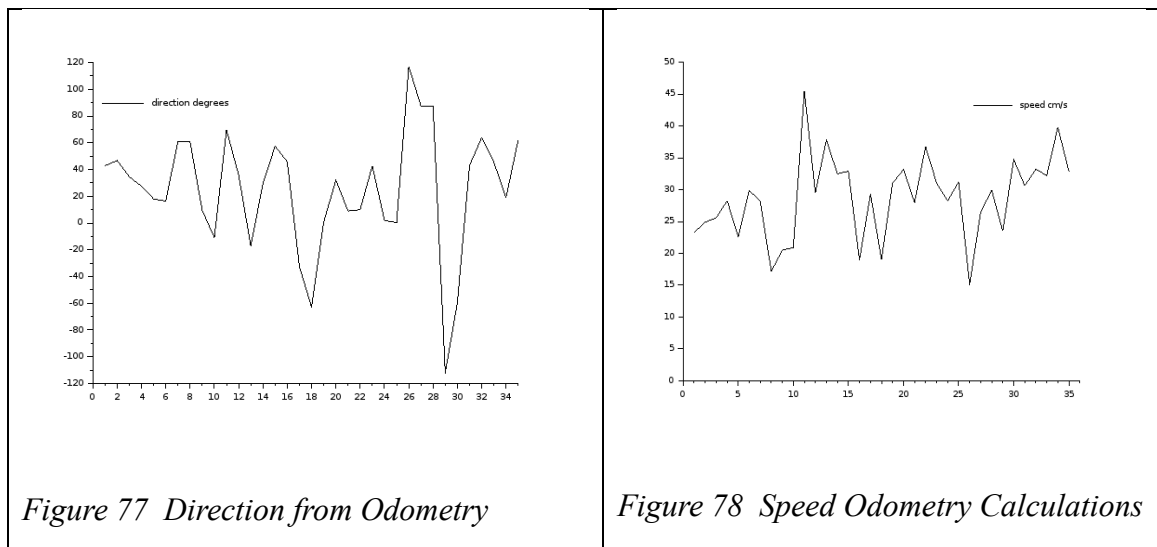
The first step in our experiment has been to calculate the homography matrix which describes the correspondence between the points on the floor in robot coordinates and the captured image. To aid the process, a dedicated dialog has been added to the ARM Console GUI program. The robot has been placed into a known position in respect to a certain pattern on the floor and four points of interest are identified in the image, their position as pixels in the image has been captured with a click and their position in centimeters in respect to the robot camera has been measured. The new dialog allowed to provide mappings between the pixels in the image and the measured coordinates.

Once all four points have been identified and mapped a special RobotCameraCalibration message is send via sDOMO from the ARM Console to ARM. Using the correspondence from this message ARM software running on the robot is calculating the matrix and saves it into the homography database which is indexed based on the angles of the camera for each calibration. When needed for calculations, the camera is moved at one of the pre-determined positions and the corresponding homography matrix is loaded.

For the purpose of this experiment the robot has been commanded to go straight for 21 seconds and it moved 163 inches i.e. 414 centimeters therefore resulting a speed of

19.71 cm/s. The odometry algorithm was added to ARM software and it was running in Real-Time on the embedded computer board. The full video processing on the ARM performed calculation with an average of 513 ms/frame therefore just a little shy of 2 frames per second. After eliminating the few records at the beginning and end of robot motion when the shocks applied to the camera make it move out of control, the data has been collected and plotted in the graphs below.

While the mean value for speed 28.69 cm/s as estimated by the camera is not very



far away from the real 19.71 cm/s value, the dispersion of measurements from frame to frame is not very consistent as can be seen from the first plot. The situation is even worse for the calculated direction of movement where the variation are totally wild.

The vector classifier algorithm has been unit tested and found to work correctly, the math for deriving the odometry from the vectors is straight forward and has been tested in simulation.

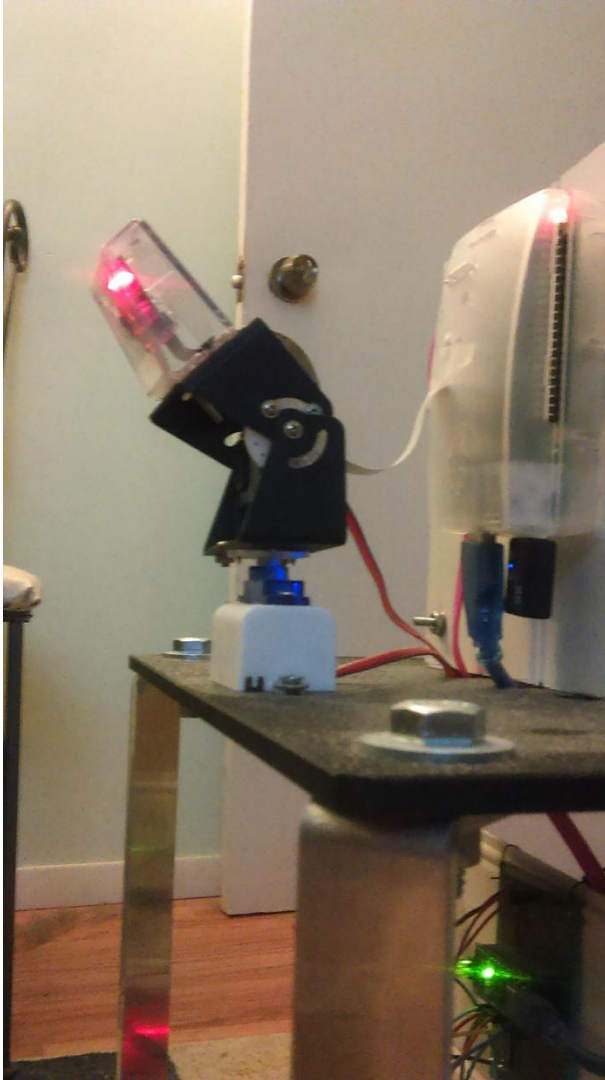


Figure 79 Pan-Tilt mechanism puts all the weight on one servo's axis.

This moved the focus to one of the pieces of equipment that already created a lot of headaches: The cheap Pan-Tilt bought for less than \$15 on Amazon.com. As can be seen from the picture, the whole weight of the camera, mechanism hardware and the servo-motor for looking up-and-down is supported by the axis of the servo-motor for left-right movement. When the robot starts or stops its motion the oscillation of the camera makes all the data capture unusable, this is the reason we eliminated from the beginning the records

when the robot started and stopped the movement. It turned out that even the data captured while the robot was in motion at a relative constant pace is not reliable enough. The camera still oscillates under the movement, even if not so obvious as when it starts and stop, and the magnitude of the noise is enough to affect the measurements.

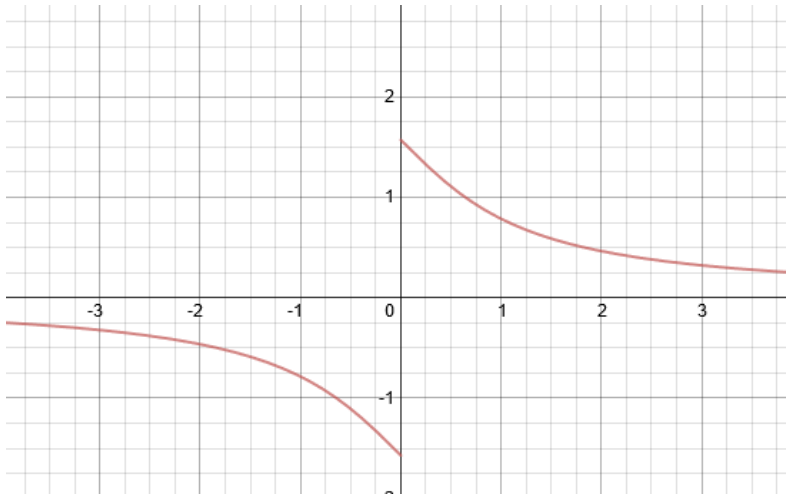


Figure 80 Example of discontinuity for $\arctan(1/x)$

An interesting question however, is why the noise impacts the measurement of angle so much more than that of the speed. The answer is again the stability of the functions. In calculations of the direction we rely on the $\arctan(y/x)$ to determine the direction of the average vector. When the vector is (almost) vertical like in the case when the robot is moving straight ahead the values for x are measured to be very close around 0. And in this area the function will present discontinuity. By contrast the speed calculations just gets the length of the average vector which does not present any discontinuity. It has been suggested that using quaternion mathematics this kind of discontinuity can be alleviated, but this approach has not been attempted in the scope of this dissertation.

An aggravating factor for our measurements has been the low frame rate we were achieving in optical flow processing, a little less than 2Hz. We were achieving this low frame rate because all the visual processing in ARM is taking place in the context of a single thread. It is expected that by rewriting the software to use two or three cores of the RPI2 the frame rate can be doubled and this may have a positive impact in accuracy.

The results above raises the question, as to what it takes to achieve quality visual odometry using optical flow processing. As everybody is accustomed the optical mouse is a widely used device performing visual odometry. I took a look at the specification for the integrated circuit ADNS-2051 which is common on many models of optical mice. The circuit has a very low resolution; only 16x16 pixels however, it is sampling the motion at a frequency of 1500 to 2300 frames per second and on top of that it is also controlling its own lightning. Even at its slowest rate, the ADSN is processing over 750 times more frames per second than our robot. Of course, an optical mouse is not doing any homography transforms nor any other advanced video processing since the sensor is expected to be always parallel and at a relatively known distance from the table. This allows the calculations to be very simple and performed at high speed by dedicated hardware DSP (digital signal processor) embedded inside the sensor.

While our data-capture at 640x480 pixels is way more accurate than to 16x16 on the mouse sensor, for the stability to noise it is the raw sampling frequency that matters. Sampling and averaging data at a frequency of 1500 Hz can eliminate a lot of the Gaussian errors due to vibrations which are impossible to eliminate at barely 2Hz sampling rate as performed by our software.

Therefore, as practical recommendation for follow-up projects it is definitely cheaper and better to add to the robot specialized odometry units like ADNS-2051, mounted under the belly of the robot and retrieve the odometry information that way. The alternatives will be to either have a gyroscopic stabilized gimbal for the camera mount or to have on the camera an Inertial Measurement Unit (IMU) consisting on an MEMS accelerometer and gyroscope and have a high speed DSP to correct the image due to detected vibrations. Both of the options will be more expensive than and probably not as accurate as just sticking a pair of mice sensors under the belly of the robot and reading them via USB.

11.11 Navigation Experiment with Fixed Camera Alone

The overall functionality of the system was subject to a series of tests in which the robot was attempted to be driven on a pre-determined trajectory with correction from the MP-Tracker algorithm. The prescribed trajectory had 6 Way-Points the last one being the target. The shape of a trajectory was roughly an ellipsis arch with about 40 degrees' difference from the starting point till the goal point. The length of the trajectory was spanning the field of view of a single camera.

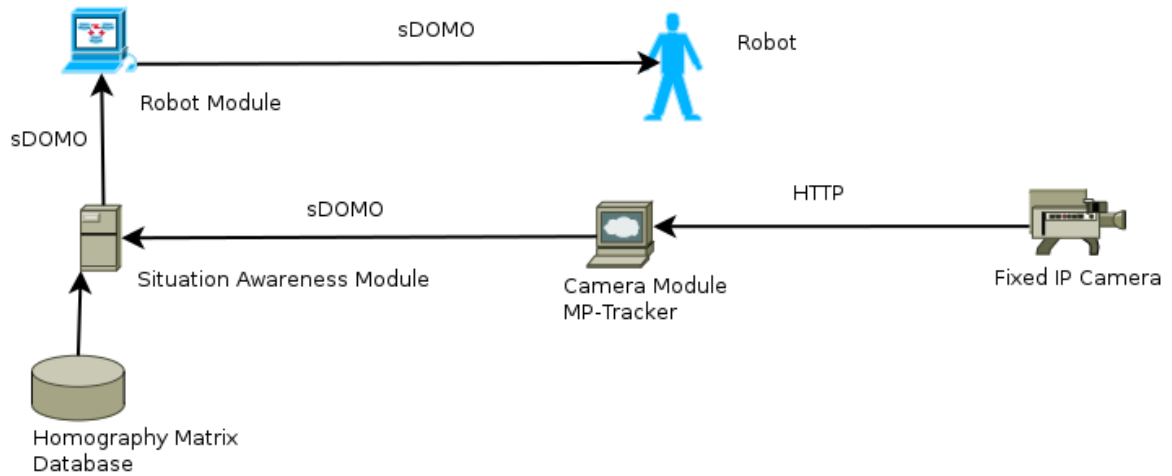


Figure 81 Experiment Design for Navigation with External Tracking

The motion of the robot triggered the CM to start sending tracking information at about 2 Hz (2 tracking packets per second). The Tracking info were translated by SAM in absolute floor coordinates and passed to RM which attempted to minimize the error between tracked position and the next Way-Point.

From 20 experiments in one of them the robot was able to complete successfully the full trajectory and it failed in the remaining 19 most of the time at the sharpest turn where the inaccuracy in localization proved too big and the robot entered in oscillation by constantly overshooting it target point.

The analysis of the failures highlighted the inaccuracy of the tracker ability to properly assess the center of the robot. Small changes in the direction of the robot appeared to the tracker as movements into a random direction and the RM was therefore attempting to compensate for the falsely perceived movement. The result was that the movement of the robot became chaotic and once it entered into such a loop, it was unable

to break free, overshooting the way-point and eventually ending up bumping into the furniture.

11.12 Conclusion and Future Work

The amount of work to fully implement the envisioned system is way over the scope of a single dissertation and follow up work is needed to fully develop the capability of the system. However, from the preliminary results we have demonstrated that in principle the architecture of an intelligent house where the robot is just a mobile component of the smart environment is a potentially viable solution. The experiment with active artificial landmarks, for example, showed the power of an integrated domotic system, and the multiplicative capability that can be obtained from the ability of the robot to interact directly with remote intelligent objects in the domestic environment.

This project spawned a multitude of fields, taking us from deriving geometric locus equations to soldering electronic circuits, drilling holes and everything in between. The literature reviews about the aging populations and the privacy implications of the Internet of Things were true eyes openers and will provide good material for rumination for quite a while.

So far, for the purpose of this work we have written from scratch in excess of 32'000 lines of code in over 300 source code files. The large majority of the code was written in C++, yet with all this amount of work we barely scratched the surface of the field of distributed processing for vision based domestic robotics. The sDOMO backbone was proven rock-solid with the Hub running for months without restarting and allowing 5 computing systems implementing various components of the system to interoperate in

real-time by exchanging messages reliably. The effort resulted in four published papers and a fifth one already accepted to a conference.

We demonstrated that it is possible to utilize cheap network and camera infrastructure that already exists in most of the houses in order to create an intelligent environment where a robot can operate. And we have shown that a distributed system for processing of the computer vision for domestic robots it is feasible.

In the process we developed a new communication protocol for Home Automation and Robotic Systems that addresses the major concerns about privacy and security of the residents and which we hope to be able to promote it to eventually become used by the industry and community. The work with sDOMO is taking a life on its own and beside two published papers it resulted on a companion website where this protocol is being actively developed and offered to the community as open source. Since so far no other existing communication protocol for Home Automation have anywhere the assortment of capabilities of sDOMO we have high expectations for this line of work.

Another important contribution is the Multi-Threaded Message Dispatcher design which is being presented as a separate research paper (already accepted at IEEE SERA 2016) and has good potential to help the community of Mission and Safety Critical software developers. The new design pattern takes away from the implementer most of the effort required to insure the correctness of critical sections allowing them to focus on the task at hand. Beside the usage of this pattern in sDOMO protocol, direct applications in industry are underway to use the MTM-Dispatcher as the core design pattern for the next generation of the Vehicle Specific Module for the Shadow 200 Unmanned Aircraft Systems developed by Textron Systems.

We designed, successfully implemented and tested an affordable platform for robotic experiments for way under \$500 which can be very easily replicated by future researchers and hobbyists. The robot architecture proved that the decision to distribute the “hard real-time” motor and servo control functions to a microcontroller while keeping in the main computer only “soft real-time” functionality is winning design decision. The experiments with the platform also highlighted the need to include supplementary hardware like specialized odometry units or inertial units and higher computing power potentially in the form of hardware accelerated DSP.

Despite the large amount of effort invested in MP-Tracker algorithm we were not able to achieve a flawless localization of the moving objects and this imprecision had significant setbacks in the navigation experiments conducted with the tracking from the external camera. While the original experiments done with MP-Tracker showed a lot of promise, despite the CPU spike on the motion of large objects, when the accuracy of the blob center detection was essential for precision navigation the hidden flaw became a problem. There is room for improvement on MP-Tracker algorithm and this is one of the follow-up works required. The follow-up work has to either improve the precision of localization or to prove that the required precision for navigation is unachievable by motion-blob tracking alone and therefore it cannot be relied upon by itself, therefore making full 3D modelling of the environment a hard requirement for precision navigation.

On the Situation Awareness Module, the bulk of the remaining work consists in implementation of 3D model and the algorithms to detect the obstacles from multiple views of the same place and update the model accordingly. Due to the huge scope of this

work it was not pursued on this dissertation. Once a full 3D model is in place, SAM can be used to improve the tracking accuracy by eliminating the perspective errors.

The Robot Module needs to be updated with more intelligence to be able to get access to the 3D model from SAM and calculate trajectories that avoid the obstacles. It also needs to implement the landmark based localization algorithms which however require full ability to recognize the landmarks. Since object recognition is a huge task in Computer Vision this work has not been pursued further. Due to high complexity that the RM is going to have to implement all these requirements, the existing experimental code base needs to be rewritten most likely using the MTM-Dispatcher in order to be flexible and easily extensible.

On the robot itself, Autonomous Robot Module needs to be updated with the ability to detect obstacles by variation in Optical Flow and inform SAM about unknown obstacles detected along the way. The existing algorithms for navigations needs also quite some rework. The video processing software needs to be rewritten to take advantage of multiple cores of the CPU to improve the frame rate and new sensors for odometry need to be incorporated in the robot to allow accurate motion estimation.

REFERENCES:

- [1] R. Madhavan, K. Fregene, and L. E. Parker, “Distributed Cooperative Outdoor Multirobot Localization and Mapping,” *Autonomous Robots*, Vol. 17 Issue 1, pp. 23-39, July 2004.
- [2] J. Cobos, L. Pacheco, X. Cuffi, and D. Caballero, “Integrating Visual Odometry and Dead-Reckoning for Robot Localization and Obstacle Detection”, *IEEE International Conference on Automation Quality and Testing Robotics (AQTR)*, Cluj-Napoca, May 2010.
- [3] A. L. Majdik, I. Szoke, L. Tamas, M. Popa, and G. Lazea: “Laser and Vision based map building techniques for Mobile robot navigation”, *IEEE International Conference on Testing Robotics (AQTR)*, Cluj-Napoca, May, 2010.
- [4] P. Biber, S. Fleck, and T. Duckett, “3D Modeling of Indoor Environments for a Robotic Security Guard”, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshop*, San Diego, June, 2005.
- [5] D. Pizarro et al., “Robot and Obstacle Localization and Tracking with an External Camera Rig”, *IEEE International Conference on Robotics and Automation*, Pasadena, pp. 516-521, May, 2008.
- [6] P. Chakravarty, and R. Jarvis: “External Cameras & A Mobile Robot: A cooperative surveillance system”, *Australian Conference on Robotics and Automation (ACRA)*, December, 2009.

- [7] K. Huang and Y. Ma, "A Survey of Geometric Vision", Robotics and Automation Handbook, CRC Press, 2005.
- [8] P. Kaewtrakulpong, and R. Bowden, "An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection", 2nd. European Workshop on Advanced Video Based Surveillance Systems, pp 135-144, 2001.
- [9] M. Sonka, V. Hlavac, and R. Boyle, Image Processing, Analysis, and Machine Vision, PWS Publishing, 1998.
- [10] M. Ulrich, C. Wiedermann, and C. Steger, "Combining Scale-Space and Similarity-Based Aspect Graphs for Fast 3D Object Recognition", IEEE Transactions on Patters Analysis and Machine Intelligence 34(10), October, 2012.
- [11] M. Marszalek, and C. Schmid, "Accurate Object Recognition with Shape Masks", Int J Comput Vis., Volume 97, pp 191-209, April, 2012.
- [12] D. L. Recio, E. M. Segura, L. M. Segura, and A. Waern, "The NAO Models for the Elderly", 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI), Tokio, pp. 187-188, March, 2013.
- [13] S. A. Mehdi, C. Armbrust, J. Koch, and K. Berns., "Methodology for Robot Mapping and Navigation in Assisted Living Environments", The 2nd International Conference on Pervasive Technologies, Corfu Greece, June, 2009.
- [14] Y. H. Wu, C. Fassert, and A. S. Riguid, "Designing robots for the elderly: Appearance issue and beyond", Archives of Gerontology and Geriatrics, Vol. 54 Pp. 121-126, January-February, 2012.

- [15] P. E. Lopez-del-Teruel, A. Ruiz, and L. Fernandez, "GeoBot: A High Level Visual Perception architecture for Autonomous Robots", IEEE International Conference on Computer Vision Systems, pp. 19, January, 2006.
- [16] A. A. S. Souza, and L. M. G. Goncalves, "2.5-Dimensional Grid Mapping from Stereo Vision for Robotic Navigation", Brazilian Robotic Symposium and Latin American Robotics Symposium, pp. 39-44, October, 2012.
- [17] A. J. Davison, "Real-Time Simultaneous Localization and Mapping with a Single Camera", Proceedings of the Ninth IEEE International Conference on Computer Vision, Vol. 2, pp. 1403, 2003.
- [18] C. S. Fernandes, M. F. M. Campos, and L. Chaimowics, "A low-cost localization system based on Artificial Landmarks", Brazilian Robotic Symposium and Latin American Robotics Symposium, pp.109-114, October, 2012.
- [19] R.I. Hartley, and A. Zisserman, "Multiple View Geometry in Computer Vision", Cambridge University Press, March, 2004.
- [20] R. Khosla, and Q. Zhang, "A computational account of dreaming: Learning and memory consolidation", Cognitive Systems Research Vol. 10, pp. 91-101, June, 2008.
- [21] M. C. Martin, "Evolving Visual Sonar: Depth from monocular images", Evolutionary computer vision and Pattern Recognition Letters, Vol. 27, Issue 11, pp.1174-1180, August, 2006.
- [22] UN Department of Economic and Social Affairs Population Division, "World Population Ageing: 1950-2050". Online:
<http://www.un.org/esa/population/publications/worldageing19502050/>

- [23] G. A. Cuckler et al., "National Health Expenditure Projections, 2012–22: Slow Growth Until Coverage Expands And Economy Improves". Health Aff October 2013 vol. 32 no. 10 1820-1831.
- [24] M.U. Farooq et al. "A Critical Analysis on the Security Concerns of Internet of Things (IoT)". International Journal of Computer Applications (0975 8887), Volume 111 - No. 7, February 2015.
- [25] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things". IEEE Computer, vol. 44, no. 9, pp. 51-58, September 2011.
- [26] M. Ball and V. Callaghan, "Who is in Control of Intelligent Environments? A Question of Autonomy". 10th International Conference on Pervasive Computing, June 18, 2012.
- [27] S. Solaimani et al. "Critical design issues for the development of Smart Home technologies". J. Design Research, Vol. 11, No. 1, 2013.
- [28] Hewlett-Packard, "Internet of Things Research Study, 2014 Report". Online: <http://h20195.www2.hp.com/V2/GetDocument.aspx?docname=4AA5-4759ENW>
- [29] B. Zhang et al., "The Cloud is Not Enough: Saving IoT from the Cloud". 7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15), 2015.
- [30] I. D. Addo et al., "Reference Architectures for Privacy Preservation in Cloud-based IoT Applications ", International Journal of Services Computing (IJSC), 2(4), 2014, pp.65-78.
- [31] D. B. Reid, "An algorithm for tracking multiple targets". IEEE Transactions on Automatic Control 24:843–854, 1979.

- [32] D. Antunes, D. M. de Matos, Jose Gaspar "Multiple Hypothesis Group Tracking in Video Sequences", In 16th Portuguese Conference on Pattern Recognition Vila Real, Portugal, October 2010.
- [33] M. Chovanec, "Computer Vision Vehicle Tracking Using Background Subtraction", Journal of Information, Control and Management Systems, Vol. 1, (2005), No.1 7.
- [34] G. Jun, J. K. Aggarwal, M. Gokmen "Tracking and Segmentation of Highway Vehicles in Cluttered and Crowded Scenes", IEEE 2008 Workshops on Applications of Computer Vision Copper, Colorado, Jan. 2008.
- [35] S. Saravanakumar, A. Vadivel, C. G. Saneem Ahmed "Multiple object tracking using HSV color space" Proceedings of the 2011 International Conference on Communication, Computing & Security, ICCCS 2011, Odisha, India, February 12-14, 2011.
- [36] M. Byeon, H. J. Chang, J. Y. Choi, "Hierarchical Feature Grouping for Multiple Object Segmentation and Tracking", IVCNZ'12 Dunedin, New Zealand, November 26-28 2012.
- [37] A. Bissacco and S. Ghiasi "Fast Visual Feature Selection and Tracking in a Hybrid Reconfigurable Architecture" Proceedings of the Workshop on Applications of Computer Vision 2006.
- [38] B Benfold and I D Reid, "Stable Multi-Target Tracking in Real-Time Surveillance Video" Proceedings of Computer Vision and Pattern Recognition, Colorado Springs, June 2011.

- [39] T. S. Hjort, R. Torbensen, "Trusted Domain: A security platform for home automation", Elsevier, Computers & Security 31 (2012) 940-955.
- [40] T. Perumal, A. R. Ramil, C. Y. Leong, "Interoperability Framework for Smart Home Systems", IEEE Transactions on Consumer Electronics (Volume: 57, Issue: 4) November 2011.
- [41] Y. W. Kao, S. M. Yuan, "User-configurable semantic home automation", Computer Standards & Interfaces 34 (2012) 171-188.
- [42] P. Sanchez et al., "A framework for developing home automation systems: From requirements to code", The Journal of Systems and Software 84 (2011) 1008-1021.
- [43] P. E. Røvsing et al., "A Reality Check on Home Automation Technologies", Journal of Green Engineering 303-327 2011.
- [44] V. Miori et al., "An Open Standard Solution for Domotic Interoperability", IEEE Transactions on Consumer Electronics, Vol. 52, No. 1, FEBRUARY 2006.
- [45] J. H. Su et al., "The Design and Implementation of a Low-cost Programmable Home Automation Module", *Consumer Electronics*, vol. 52, Issue 4, Nov. 2006, pp. 1239-1244.
- [46] A. Z. Alkar et al., "IP Based Home Automation Systems", Consumer Electronics, IEEE Transactions on, 2010.
- [47] Internet Engineering Task Force, RFC-6120, RFC-6121. P. Saint-Andre, March 2011. ISSN: 2070-1721.
- [48] Nest Labs Inc., "About the Nest API"; Online: <https://developer.nest.com/documentation/cloud/about/>

- [49] Internet Engineering Task Force, RFC-7252, Z.. Shelby et al, June 2014. ISSN: 2070-1721.
- [50] M, Castro et al., “Enabling end-to-end CoAP-based communications for the Web of Things”. Journal of Network and Computer Applications (2014), Online:
<http://dx.doi.org/10.1016/j.jnca.2014.09.019>
- [51] S. M. Chun et al., “CoAP-Based Mobility Management for the Internet of Things”. Sensors 2015, 15, 16060-16082; doi:10.3390/s150716060.
- [52] R. Piyare and S. R. Lee, "Towards Internet of Things (IoTs): Integration of Wireless Sensors Network to Cloud Services for Data Collection and Sharing".International Journal of Computer Networks & Communications (IJCNC) Vol.5, No.5, September 2013.
- [53] J, Singh et al., "Twenty Cloud Security Considerations for Supporting the Internet of Things" in Internet of Things Journal, IEEE , vol.PP, no.99, pp.1-1, July 2015.
- [54] Á. Asensio et al., "Managing Emergency Situations in the Smart City: The Smart Signal".Sensors 2015, 15, 14370-14396; doi:10.3390/s150614370.
- [55] R. Nandyala, “Intelligent Internet of Things (IIoT): System Architectures and Communications”. Online:
https://www.academia.edu/14076518/Intelligent_Internet_of_Things_IIoT_System_Architectures_and_Communications
- [56] D. C. Schmidt, “Reactor – An Object Behavioral Pattern for Demultiplexing and Dispatching Handles for Synchronous Events”, 1995.[57] R. Greg Lavender and Douglas C. Schmidt, “Active Object – An Object Behavioral Pattern for Concurrent Programming”. 1996.
- [58] R. G. Lavender and D. C. Schmidt, “Monitor Object - An Object Behavioral Pattern

for Concurrent Programming”. 1996.

[59] I. Pyrali, C. O’Ryan, and D. C. Schmidt, Patterns for Efficient, Predictable, Scalable, and Flexible Dispatching Components. In *7th Pattern Languages of Programs Conference (PLoP)*, 2000.

[60] M.T. Marginean and C. Lu, “A Distributed Processing Architecture for Vision Based Domestic Robot Navigation”, International Conference on Computers, Communications and Systems; Nov 1, 2013.

[61] M. T. Marginean and C. Lu, “A Multi-Paradigm Object Tracker for Robot Navigation Assisted by External Computer Vision”. ACM RACS, 2014.

[62] M. T. Marginean and C. Lu, “sDOMO – A Simple Communication Protocol for Home Automation and Robotic Systems”, IEEE International Conference on Technologies for Practical Robot Applications; May 11 – 12, 2015.

[63] M. T. Marginean and C. Lu, “sDOMO Communication Protocol for Home Robotic Systems in the Context of Internet of Things”, International Conference on Computer Science, Technology and Application March 18, 2016.

[64] Mezonix Technology: “sDOMO - A protocol for fully integrated Home Automation and Robotic Systems”, Online: <http://www.mezonix.com/sDOMO.html>

[65] C. Stauffer, W. E. L. Grimson “Adaptive background mixture models for real-time tracking”. IEEE Computer Society Conference on Computer Vision and Pattern Recognition **2**. pp. 246–252. doi:10.1109/CVPR.1999.784637.

[66] H. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing

Robot Rover". Tech Report CMU-RI-TR-3 Carnegie-Mellon University, Robotics Institute, 1980.

[67] C. Harris and M. Stephens. "A combined corner and edge detector". Proceedings of the 4th Alvey Vision Conference 1988. pp. 147–151.

[68] B. D. Lucas, and T. Kanade, T. , "An iterative image registration technique with an application to stereo vision". Proceedings of Imaging Understanding Workshop, pages 121–130, 1981.

[69] J. Y. Bouguet, "Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm". *Intel Corporation*, 5(1-10), p.4, 2001.

Marcel-Titus MARGINEAN

Highly experienced engineer and scientist with extensive knowledge in real-time software for unmanned aerial vehicles, networking, embedded devices, automation, advanced algorithms, distributed systems, computer vision, security, cryptography and robotics. Vast interdisciplinary knowledge, strong background in mathematics, ardent passion for technology, strongly driven by active curiosity to explore new fields and push upwards the boundaries of possible.

HISTORY

2008-Current: Textron Systems – Principal Software Engineer

- Company representative at Industry-Government Consortia: *Future Airborne Capability Environment* and *Sensors Open Systems Architecture*
- Design Architectures for Airborne Vehicle Specific Modules for UAVs
- Real-Time Ground Support Software for UAVs
- Algorithm for AV position estimation in GPS denied environments
- Software for UAV takeoff/landing using RTK-GPS

2004-2008: LRW Digital – Software Engineer

- Cryptographic Protocol for PIM synchronization between Mobile Device and Enterprise Infrastructure
- Encrypted Storage for Mobile Device
- Encrypted protocol for remote management of mobile devices

2003-2004: Smartpants Media – Software Engineer

- Embedded Linux Video Encoder and Broadcaster
- Infrastructure for Web access to Surveillance Video

1999-2003: Mastech – Software Consultant for Corning

- Mathematic Algorithm for optimal cable design
- Distributed System for Inventory Management

1993-1999: Romania & Sweden – Software Engineer and IT Consultant

- First commercial ISP in Alba County, Romania
- Replicator Algorithm for Distributed Database System
- Custom Software for Networking & Data Management

CONTACTS

web: <http://www.mezonix.com>

EDUCATION

Doctor of Science

Information Technology
2016 Towson University, USA
Dissertation: Distributed System for Domestic Robot Operation Using Computer Vision

Electrical Engineer – 5 Years advanced degree

1993 Polytechnic University of Timisoara, Romania

Major: Information Systems for Process Control

Thesis: Optimization of Flexible Manufacturing Lines Attended by Industrial Robots

Electro-Mechanic Technician

1987 Industrial School No. 1
Alba-Iulia, Romania

PUBLICATIONS

- **A Distributed Processing Architecture for Vision Based Domestic Robot Navigation.** Marcel-Titus Marginean and Chao Lu. International Conference on Computers, Communications and Systems; Nov 1 2013 Daegu University, Korea
- **A Multi-Paradigm Object Tracker for Robot Navigation Assisted by External Computer Vision.** Marcel-Titus Marginean and Chao Lu, ACM Conference on Research in Adaptive and Convergent Systems; October 6 2014 Towson, MD, USA
- **sDOMO – A Simple Communication Protocol for Home Automation and Robotic Systems.** Marcel-Titus Marginean and Chao Lu, IEEE International Conference on Technologies for Practical Robot Applications; May 12 2015 Boston, MS, USA
- **sDOMO communication protocol for home robotic systems in the context of the internet of things.** Marcel-Titus Marginean and Chao Lu, International Conference on Computer Science, Technology and Application; March 18 2016 Changsha, Hunan, China
- **Multi-Threaded Message Dispatcher Framework for Mission Critical Applications.** Marcel-Titus Marginean and Chao Lu, IEEE/ACIS International Conference in Software Engineering, Research , Management and Applications; June 9 2016 Towson, MD, USA

ACTIVE RESEARCH INTERESTS

- Integrated Domotic Systems
- Privacy and Security of Home Automation Systems
- Design Patterns for Critical, Real-Time Applications
- Computer Vision for Robots and Unmanned Vehicles

OPEN-SOURCE PROJECTS

- **sDOMO:** A protocol for fully integrated Home Automation and Robotic Systems, putting emphasis on security and privacy of the residents
 - <http://mezonix.com/sDOMO.html>
- **MTM-Dispatcher:** Multi-Threaded Message Dispatcher - Minimalistic Framework and Design Pattern to aid development of Mission/Safety Critical Message Processing Applications
 - <http://mezonix.com/MTMDispatcher.html>

