Building a Non-Euclidean Roadmap from a Small Set of Images

Project By:

Alexandra Levinsky

Majd Srour

Supervised By:

Prof. Ehud Rivlin

Table of Contents

Introduction

In many tasks a mobile robot is required to navigate from any initial position to a desired position in its environment. This kind of task will require a global localization technique that allows the robot to operate without prior knowledge about its position and recover from odometry errors. Thus, the representation of the target position has to provide enough information for global localization. Such a representation should be robust with respect to minor changes in the environment.
Prevalent approaches to the navigation problem are model-based navigation and appearance-based navigation. Model-based approaches rely on knowledge of a 3D model of the environment. The localization is then performed by matching the global model with a local model deduced from sensor data .The model-based approaches require that either a map of the environment be provided, or that it be built by the robot. The latter is usually termed SLAM (Simultaneous Localization and Map Building).

The appearance-based approach does not require a 3D model of the environment. It has the advantage of working directly in the sensor space.

Our work is in the domain of the appearance-based approach. The goal is to implement [1] for an autonomous robot given a small set of images: these are taken in the preparation stage, with the robot's camera, by the user in order to describe the environment. The positions from which the images were taken are the target locations.
Each image represents the position and orientation from which it was taken. Each image defines a region in the configuration space from which the image can be matched.
In the learning stage the robot autonomously explores the environment, locates the targets, and builds a graph of the paths between them. The exploration is driven by the coverage problem. After the learning stage the robot can navigate to any target in the environment from any position using the graph.

Project's Goal

To implement the algorithm [1] (Building a Non-Euclidean Roadmap from a Small Set of Images) on the Pioneer 3-AT robot.

The robot should receive a small set of target images, taken by the user in the room with the robot's own camera. The robot should autonomously learn the positions of the targets. Those positions should be saved in a manner allowing navigation between targets without any global positioning information. Also an option for navigation to a specific target from unknown position is needed.

The algorithm should run on the Pioneer 3-AT robot. The robot shown in Fig 1, is equipped with a single stationary camera. The robot moves only on the plane (X-Z plane).

Figure 1: The robot

Camera

We used an IP network camera - Axis 270, which is a normal mono-vision camera. In order to fix the distortion in the image and to derive the Essential Matrix, we needed to know the intrinsic and distortion parameters of the camera, in order to do so we used the Camera Calibration Toolbox in Matlab.

We took 20 images of a chessboard *(Fig. 2)* with different orientations, and then we used the toolbox to extract the required parameters. The process is simple, for each image we select the 4 corners as shown in *Fig 3a*, then the toolbox find all the corners of each square as shown in *Fig 3b*.

The chessboard is used for calibration as it has certain properties that make it good for calibrating purposes:

- It is a plane. We don't need to deal with depth.
- There exist points on this plane that you can identify uniquely (such as corners).
- You can extract these points very easily.
- There points physically lie in straight lines, no matter how they appear on the camera.

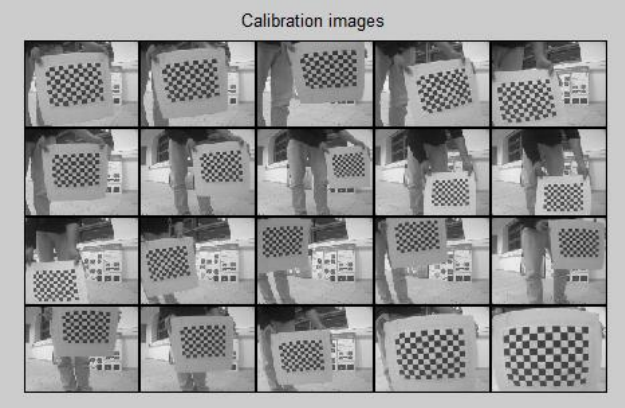

Figure 2: 20 images of the chessboard with different orientation

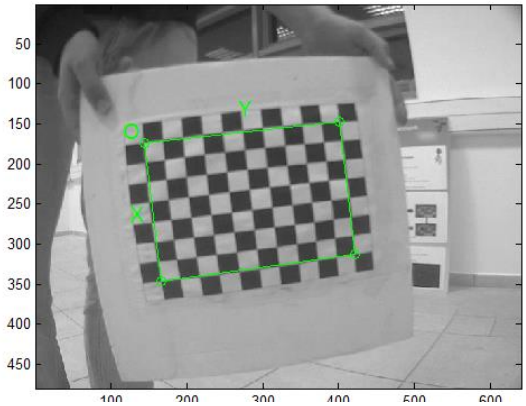

Figure 3: (a) Selecting the 4 corners of the chessboard – marked with circles

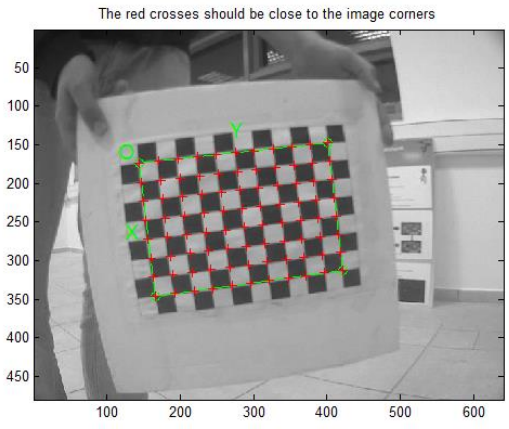

Figure 3: (b) Automatic detection of all the corners in the chessboard

After calibrating the camera and getting the intrinsic and distortion parameters, we use OpenCV remap function to ‘undistort’ the images taken from the camera (Fig. 4).

Figure 4: to the left is the original image, and to the right is the undistorted one.

Algorithm

Overview

The algorithm receives a small set of target images, taken by the user in the room with the robot’s camera. The algorithm assumes that the images are informative (containing enough features to represent the location). The autonomous algorithm consists of two learning stages and an application for navigation:

1. Locating the targets: The robot searches for the targets in the room while maintaining reliable positioning. The search is driven by the goal to cover the search space with a minimum number of movements and images. To solve the coverage problem, we have defined the “range” of our special “sensor.”
2. Connecting the targets in the graph: The robot traverses the targets to improve its estimate of their relative positions. The result is a graph with each directed edge containing the required motion between the two targets.
3. Navigation: The robot is placed in an unknown position in the environment and is asked to navigate to a specific target. The robot rotates on the spot, taking images and trying to match them to any of the targets. If it does not succeed, it moves in a random direction and tries again. Once a target is matched, the robot uses the homing algorithm to reach it. From the intermediate target the robot uses the graph to traverse to the desired target.

Algorithm for Building and Using a Non-Euclidean Roadmap

The autonomous algorithm consists of two learning stages and an application for navigation:

1 Finding the targets.
2 Building the graph.
3 Navigation from an unknown pose to a target.

Finding the targets

Finding the targets consists of two stages:

Room Estimation

Since the algorithm works without any prior model of the environment in which it operates, the robot builds a rough estimation of the environment. This estimation is needed in order to define the space to be covered and to estimate, for any position and orientation, the distance to the viewed scene.

The free floor space is estimated by 12 stereo image pairs. Two images with cameras heading in the same direction with knowledge of the distance between the cameras centers. The points in the images are matched by searching for a match (only in the epipolar line, which is a horizontal line in this case). Such a pair of images with the matched points is depicted in Fig 5.

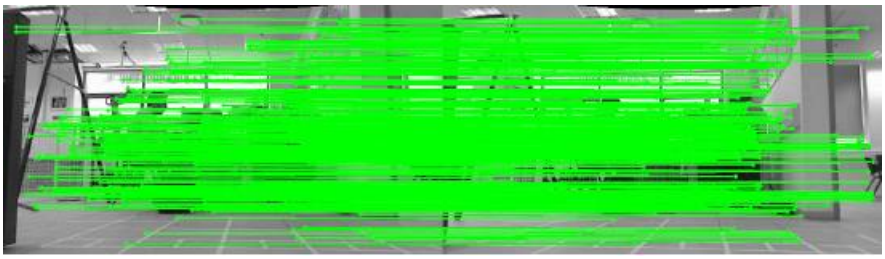

Figure 5: SIFT features between the left and right image

The robot traverses a dodecagon, taking two parallel images from the corners of each edge and then rotating by 30 degrees. The stereo pair locations are depicted in Fig 6.

The images taken during this process are also matched to the targets. The matches are remembered by the system and the robot explores them after the room estimation has been completed.

From the images taken during the room estimation process, the image with the largest number of features is chosen to be the **reference image**. The position and orientation (pose) from which this image was taken becomes the reference pose to which the robot returns to improve its positioning.

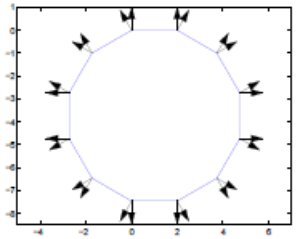

Figure 6: 12 stereo pairs

Room Coverage

Our problem is actually a coverage problem in which the robot traverses the positions which maximize the coverage of its configuration space while trying to match the targets. The coverage process continues until all the targets have been found. An optimal solution for the coverage problem would require an algorithm that would attempt all pose sequences with look-ahead and choose a sequence that will cover the space with a minimum number of poses. Such solution is not feasible due to its extremely high computational cost. To generate a practical working solution, we divide the space into grid with a spatial resolution of 10cm and an angular resolution of 5 degrees, then for each position, 12 images will be taken, one every 30 degrees. For each set of 12 images a score is computed to maximize the covered space while preventing the fragmentation of the uncovered portion of the configuration space.

Once the algorithm selects a position and first orientation, the robot traverses to that position an starts taking the 12 images. The coverage is updated after each image is taken, because not all the images will be taken if there is a match. Once a target is match the robot commences to navigate to that target. After the target is reached, the robot returns to the reference pose. Even if no target were found, the robot returns to the reference pose thus preventing accumulation of localization errors and allows the robot to better estimate the pose of the target.

Graph Building

After the target poses relative to the reference pose have been found, the robot needs to find pairs of targets between which it can travel safely, without getting lost. The robot attempts to connect every target with every other target, but it is allowed to fail in some cases. All that is required is that the resulting graph be connected. There will be an edge in the graph between targets a and b only if the robot succeeded in going from a to b and back. If the robot gets lost, it returns to the reference pose, discards the edge it attempted to create, and proceeds with the next edge.

The relative positions and orientations of a and b are calculated using the estimation from a to b and the estimation from b to a with their covariance. The estimation in each direction is obtained by using observation 1 (the angle between the first matching image and the target image). The estimations of the two directions are merged to give a better estimation with a smaller covariance. The information saved for directed edge $a \rightarrow b$ the description of the path from a to b. Starting at pose a the robot first rotates $angle1$, then drives distance $dist$, and finally rotates $angle2$ to reach pose b.

Navigation to target

After the graph has been build, the robot can use it to navigate. The robot is placed in an unknown pose and commanded to navigate to a certain target. The navigation algorithm is as follows:

1. Rotate on the spot, trying to match any images in the graph. If no match is found, move in a random direction and go to 1.
2. Navigate to the matched image.
3. Find a path on the graph from the current image to the target image. Choose the path with the lowest cost, where the cost of an edge is $\sigma_x + \sigma_y$.
4. Navigate to the images/set-points on the path until the target is reached.

The algorithm is actually a vision-based roadmap algorithm without the assumption that the robot possesses perfect positioning.

Navigation to a Single Target

One of the main building blocks of the system is a process for reaching a pose from which a target image was taken. Assuming the robot is at pose in which there is an overlap between the target Image I and the current image I', the algorithm has to calculate the epipolar geometry connecting the two images. A full-perspective projection model and knowledge of intrinsic camera calibration parameters are assumed. We therefore wish to estimate the essential matrix. We will first describe the feature matching process, then how the matched features are used to robustly recover the direction to the target and the difference between the orientations. It is impossible to find the distance to the target from two images and therefore a third image is required.

Feature Matching

In the implementation the SIFT feature detector is used. The SIFT feature descriptor is invariant to scale and orientation changes. We have canceled its orientation assignment step, and the descriptor is calculated with a fixed orientation. The orientation assignment is useful when the orientation of the features can change substantially between images. In planar motion the feature orientation changes are very small. Thus, assigning a fixed orientation is more accurate than relying on the orientation calculated by the SIFT algorithm. Another benefit is that no additional key points are generated at locations with multiple dominant orientations. Thus reduces the matching time.

The SIFT matching algorithm gives for each match the ratio between the distance of the best match and the second-best match. The lower the ratio, the higher is the probability that the match is correct. In our implementation we only use matches that have a ratio below 0.65

Finding the Direction to the Target

In planar motion there are only three degrees of freedom. Between the positions and orientation in which images I and I' were taken, there is a rotation about the Y-axis theta and a translation on the X-Z plane $(t_x, 0, t_z)^T$. It was shown in [2] that in planar motion the essential matrix is

$$E = \begin{pmatrix} 0 & -\cos(\theta) + \sin(\theta) t_x / t_z & 0 \\ 1 & 0 & -t_x / t_z \\ 0 & \sin(\theta) + cos(\theta) t_x / t_z & 0 \end{pmatrix}$$

This essential matrix will hold for each match between points (x_i, x_i') :

$$x_i' E x_i = 0$$

In order to find E from the equation above linearly, at least three correspondences are needed. We use the linear solution as an initial guess for θ and t_x / t_z . We improve the initial guess by running Sampson distance on all correspondence.

Since we can only recover θ and t_x / t_z , we do not know from the essential matrix in which direction on the line defined by t_x / t_z the target position is. We get the direction on that line by

first transforming the points on image I' by rotating them about the $Y-axis$ by θ. Afterwards we calculate the shifts in the location of the points between the points in I and the rotated points in I'. From each shift we get if the robot needs to go forward or backward on t_x / t_z line. We choose to go forward or backward by the majority vote.

Finding the Distance to the Target

It was shown in [2] that the distance to the target can be found by taking a step in the t_x / t_z direction and capturing a third image I'' (the first image is denoted by I), the author showed that if the robot's step had length λ, then from each correspondence in the three images (x_i, x_i', x_i'') the distance to the target can be calculated. In the following equations α represents what percentage of the distance to the target is λ, and d_i is the estimate of the distance to the target obtained from triplet i.

$$d_i = \frac{\lambda}{\alpha} = \frac{\lambda(x_i' - x_i)(x_i'' - t_x / t_z)}{(x_i'' - x_i')(x_i - t_x / t_z)}$$

Results

During the testing, we gave the robot 3 targets, the left images (Fig. 7 a). The right images (Fig. 7 b) are the pose the robot reached; it is clearly the robot almost hit the same post as the target. We can state the algorithm works almost perfectly, due to the accurate robot odometers.

Figure 7: (a) target images

Figure 7: (b) reached goals

Challenges

1. **Learning to communicate with the robot:**
   We needed to learn how to communicate with the robot, and learn how to use the Aria library.
2. **Accessing the Network Camera:**
   Communicating with the camera wasn't easy, as it doesn't provide an interface for C language for capturing images, what we had to do is to use an external library (poco) which gives us the ability to download images from an URL. So, what we did, we program the camera to generate images in a fixed URL, and then, using the library to download the image to a memory buffer, then reinterpreting the memory as an IplImage (OpenCV Image format).
3. **Understanding the first version of the code:**
   One of the main challenges was trying to understand the first version of the code which is divided into tens of Matlab .m files and C++, the code was messy and isn't documented. We had to read the thesis few times in order to understand the code. After doing so, we had to change many things to tune the code to work with our robot and camera, since it was tuned to a different robot platform.
4. **Run time:**
   Because most of the computations are done on Matlab, and what we did, is opening a Matlab engine from C and then running the Matlab scripts, and the communication to the camera is done via "URL" so downloading the image takes time; the code took a lot of time to run, about an hour!. So each time we needed to fix something (debug) we had to start all over, and after ~1.5 hours we had to recharge the robot, thus taking from us a lot of wasted time.

Future Work

1. **Improve the overall runtime:**
   The runtime can be improved by porting the code into C, and getting a better camera (faster access), we believe this will improve the code by 40% at least, as most of the runtime is wasted on accessing the camera and opening the Matlab engine.
2. **Multi-Robots:**
   Instead of one robot doing all the Room Building and graphs, the algorithm can better be improved to use multi-robots, each robot will be responsible on some of the targets, thus improving the runtime and giving more reliability, if one of the robots is dead, then the other one will take care of the remaining targets.
3. **Porting the code into ROS:**
   ROS is Robotic Operating System, which is the standard system/simulator for robotics nowadays, the code can be imported to work with ROS and Gazebo (Graphical/Physical simulator), which enables us to run the algorithm on the simulator and the real robot in the same way, also, allowing us to run the code in a distributed system, which, makes the performance even better by running it on multiple computers, and easing the process to improve the algorithm with multi-robots, as, each robot will have his own computer and ROS can communicate with each one as a distributed system.

Acknowledgment

We'd like to express our deep gratitude to the *CIS* lab for providing us with the tools and equipment to make this project possible.

Bibliography

1. *Ilan Shimshoni, Ehud Rivlin, Sylvina Rybnikov : "**Building a Non-Euclidean Roadmap from a Small Set of Images"**, Technion 2009*
2. *Ehud Rivlin, Ilan Shimshoni, Evgeny Smolyar : **"Image-Based Robot Navigation in Unknown Indoor Environments"**, Dept. of Computer Science, Dept. of Ind. Eng. And Mgmt. Technion – Israel Institute of Technology, IEEE/RSJ October 2003*