Robotic Home Sentinel

David Pereira, Ricardo Ferreira, José Gaspar
Instituto Superior Técnico, Lisbon, Portugal
davidpereira@ist.utl.pt, ricardo@isr.ist.utl.pt, jag@isr.ist.utl.pt

***Abstract*—This dissertation tackles the problem of mapping and navigating a mobile robot in indoor environments using artificial vision. A color-depth camera was first chosen as a sensor. Using the depth information is possible to infer the 3D location of selected points in the image (visual features) and estimate camera motion. Depth images also allow to calculate the free space in front of the robot. From several readings of the free space an occupancy grid can be created that is then used to find paths for navigation. An RGB camera was also used and a visual SLAM approach was adopted. The results obtained for both methodologies are presented at the end.**

Keywords: **Camera Model, EKF, SLAM, Path Planner, Rigid Transformation.**

I. Introduction

Recent vision sensors, namely the Microsoft Kinect allow effectively measuring the empty space in front of a robot.The Kinect depth sensor is a structured light 3D scanner. It captures both depth and color images simultaneously at a frame rate of up to 30 fps. This paper is therefore concerned with using Kinect like sensors for freely navigating in a cluttered home environment while reporting prespecified scene-views.

A. Related Work

In [4] it is presented a real-time algorithm which can recover the 3D trajectory of a monocular camera, moving rapidly through a previously unknown scene. In [3] it is presented a new parametrization for point features within monocular simultaneous localization and mapping. For camera motion estimation some important references are [6] where a kinematic model-based approach for the estimation of 3-D motion parameters from a sequence of noisy stereo images is discussed. In [1], [2], where the problem considered involves the use of a sequence of noisy monocular images of a three-dimensional moving object to estimate both its structure and kinematics. This work started as the continuation of the work developed by João Tomaz [5] in his MSc thesis that followed a SLAM approach.

B. Problem Formulation

Using the sensor available we have to be able to characterize the surroundings in every step. The ultimate goal is to map the environment and plan the path the robot takes to reach its goal using first a kinect-like camera (RGB+D) and then a normal (RGB) camera. It uses only visual odometry and has to detect and bypass obstacles that may present in its way.

C. Report Structure

Section I introduces the problem to approach in the dissertation, in particular presents a short discussion of the state of the art. Section II presents the theoretical background in camera model, perspective projection and Kalman filtering. Section III describes how the mapping and navigation using a color-depth camera is made. Section IV describes how the mapping and navigation using a monocular camera is made. Section V shows the results of some experiments done to validate the ideas presented. Section VI summarizes the work performed and highlights the main achievements in this work. Moreover, this chapter proposes further work to extend the activities described in this document.

II. Background

This section provides an introduction to the theoretical background concepts that will be used in the thesis. Two main aspects are introduced here: the filtering of noisy data based in Kalman filtering and the camera model that gives structure to the measurements equation of the filtering process.

A. Camera Model

The principles of perspective were discovered during the Italian Renaissance by da Vinci and other architects and painters. Cameras perform a projection of points in space onto the image plane described by a perspective projection

$$\mathbf{m} = \mathbf{\Pi M} \tag{1}$$

where M, m are the homogeneous coordinates of the points in space and in the image plane and Π is a 3×4 matrix denoted as camera matrix.

Cameras concentrate electromagnetic radiation (light rays) on the surface of their sensors using lenses. However this whole process can be approximated by the pin hole model which simplifies and reduces the lenses to a small hole and all the light rays arriving pass through that hole, called the optical center.

From the pin hole model the coordinates for point P are (X,Y,Z) and for its projected point p are (x,y). Using the similarities of triangles we get

$$\begin{aligned} x &= f\frac{X}{Z} \\ y &= f\frac{Y}{Z} \end{aligned} \tag{2}$$

and f is the focal length of the camera. All these variables are measured in meters.

The normalized camera model ($f = 1$) in homogeneous coordinates is written as

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3}$$

1) Extrinsic Parameters: The coordinates of a point in space might not be known relative to its optical center if this is hidden inside the camera. The coordinates of a point in another frame with arbitrary location and orientation X_0 are related to the camera coordinates by a rigid body transformation where $R_{(3x3)}$ is a rotation matrix and $T_{(3x1)}$ is a translation vector and it can be written using homogeneous coordinates:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{4}$$

These parameters (R,T) are called the extrinsics parameters of the camera since they depend on the camera pose.

2) Intrinsic Parameters: The coordinates of a point in an image are measured in pixels and the frame is not usually centered at the principal of the camera. That relation between the homogeneous coordinates of the image points in meters and in pixels is represented by

$$K = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

where f is the focal length, s_x and s_y are the conversion factors from meters to pixels in their respective axis and o_x,o_y are the coordinates of the principal point in the new frame. Also, s_θ is a diagonal scale (skew) factor. These parameters are denoted as intrinsic parameters of the camera and they are independent of the camera pose. Matrix $K_{(3x3)}$ is called the matrix of intrinsic parameters. The camera model with intrinsic and extrinsic parameters is given by $\lambda x' = \Pi X_0$ where $\Pi = K[RT]$. If Π_i^T is the i-th line of Π, we get

$$\begin{aligned} x' &= \frac{\pi_1^T X_0}{\pi_3^T X_0} \\ y' &= \frac{\pi_2^T X_0}{\pi_3^T X_0} \end{aligned} \tag{6}$$

3) Depth Image to 3D Points: Cameras project points in 3D to the image plane. Depth cameras give that additional information making it possible to locate in space the position of every pixel in its image. Cartesian X, Y, Z points are represented by the camera as Z(u,v) where u, v are image points i.e. perspective projection (projective) coordinates. The following equation 7 relates the 3D position of a point with its projection in the image plane:

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \begin{bmatrix} \alpha_u f & 0 & u_0 & 0 \\ 0 & \alpha_v f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X^c \\ Y^c \\ Z^c \\ 1 \end{bmatrix} \tag{7}$$

If we make $\lambda = d$ where $d = Z(u,v)$ or the depth value for the pixel in image coordinates $(u,v)$and if we put X,Y and Z in evidence, we arrive at

$$\begin{cases} X(u,v) = \frac{u-u_0}{\alpha_u f} Z(u,v) \\ Y(u,v) = \frac{v-v_0}{\alpha_v f} Z(u,v) \\ Z(u,v) = d(u,v) \end{cases} \tag{8}$$

where we get our 3D coordinates of the pixel at (u, v).

4) Back-projection: The 3-D ray that starts in the camera optical center $\mathbf{C}$ and pierces the image plane in the point projected eventually reaches the 3-D point. The camera center $\mathbf{C}$ is defined as the intersection of the three planes defined in the projection matrix $\mathbf{P}$ columns, where $\mathbf{PC} = 0$. If we redefine $\mathbf{P}$ as $\mathbf{P} = \left[\mathbf{p}_{(123)}|\mathbf{p}_4\right]$ where $\mathbf{p}_{(123)}$ is a 3×3 matrix representing the first three columns of $\mathbf{P}$ and $\mathbf{p}_4$ as the last column. We can compute the camera center as

$$\mathbf{P}\begin{bmatrix} \tilde{\mathbf{C}} \\ 1 \end{bmatrix} = 0 \Leftrightarrow \mathbf{p}_{(123)}\mathbf{C} + \mathbf{p}_4 = 0 \Leftrightarrow \mathbf{C} = -{\mathbf{p}_{(123)}}^{-1}\mathbf{p}_4. \tag{9}$$

$\mathbf{D}$ is the 2-D point back-projection in 3-D coordinates, defined by the ray leaving the camera center and intersecting the plane at infinity, and its representation becomes

$$\mathbf{x} = \mathbf{PD} \Leftrightarrow \mathbf{P}\begin{bmatrix} \tilde{\mathbf{D}} \\ 0 \end{bmatrix} = \mathbf{x} \Leftrightarrow \mathbf{p}_{(123)}\mathbf{D} = \mathbf{x} \Leftrightarrow \mathbf{D} = {\mathbf{p}_{(123)}}^{-1}\mathbf{x} \tag{10}$$

where x is in homogeneous coordinates. $\mathbf{C}$ and $\mathbf{D}$ determine a line in 3-D space where all the points in that line is given by $\mathbf{X} = \mathbf{C} + \alpha\mathbf{D}$ where α is a scaling factor that $\in [-\infty, +\infty]$. The following equation is the full expression for representing a point x back-projection into to $\mathbf{X}$

$$\mathbf{X} = -{\mathbf{p}_{(123)}}^{-1}\mathbf{p}_4 + \alpha{\mathbf{p}_{(123)}}^{-1}\mathbf{x}. \tag{11}$$

B. Integration of measurements

A camera provides many measurements at each time stamp. Camera data is naturally corrupted by noise, both radiometric and geometric. Kalman filtering is a methodology to integrate camera measurements along time, such that the random nature of the noise is progressively attenuated.

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) solution of the least-squares method. So the general problem of trying to estimate the state $\mathbf{x} \in \Re^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$\mathbf{x}_{k+1} = A_k\mathbf{x_k} + B\mathbf{u_k} + w_k, \tag{12}$$

with a sensor measurement $z \in \Re^m$ that is

$$\mathbf{z}_k = H_k\mathbf{x_k} + v_k. \tag{13}$$

The random variables w_k and v_k represent respectively the process and measurement noise. They are assumed to be independent of each other, white and with a normal probability distributions

$$\begin{aligned} p(w) &\sim \mathcal{N}(0, Q), \\ p(v) &\sim \mathcal{N}(0, R). \end{aligned} \tag{14}$$

The $n \times n$ matrix A in equation 12 relates the state at instance k to the state at instance $k + 1$, in the absence of either an input or process noise. The $n \times l$ matrix B relates the control input $u \in \Re^l$ to the state $\mathbf{x}$. The $m \times n$ matrix H in the measurement equation 13 associates the state to the measurement z_k.

1) Extended Kalman Filter: The Extended Kalman Filter (EKF) allows handling non-linear system functions and measurement models through linearization with a first order Taylor series expansion. Both Kalman Filter and its non-linear extension can be described in two main steps with a standard set of equations, denominated respectively as prediction and update, also known as *a priori* and *a posteriori* estimates. Let the state vector $\mathbf{s}_k$ be represented as a Gaussian distribution process, with $\mathbf{x}_k$ mean value and $\mathbf{P}_k$ as the uncertainty of the estimates covariance matrix, i.e. $\mathbf{s}_k \sim \mathcal{N}(\mathbf{x}_k, \mathbf{P}_k)$, with initial state estimate also normally distributed $\mathbf{s}_0 \sim \mathcal{N}(\mathbf{x}_0, \mathbf{P}_0)$.

EKF Prediction Step: The mean value $\mathbf{x}_k$ and the covariance matrix $\mathbf{P}_k$ predicted transition from an instant k to $k + 1$ is accomplished trough the following set of equations:

$$\bar{\mathbf{x}}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \epsilon_k) \tag{15}$$

$$\bar{\mathbf{P}}_{k+1} = \mathbf{F}_k\mathbf{P}_k\mathbf{F}_k^T + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^T \tag{16}$$

The system dynamics non-linear model function $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{n}_k)$ depends on the system input control $\mathbf{u}_k$, and input transition noise $\epsilon_k \sim \mathcal{N}(0, \mathbf{Q}_k)$ with covariance matrix $\mathbf{Q}_k$. $\mathbf{F}_k$ and $\mathbf{G}_k$ stand for the state transition Jacobian matrix and input transition noise Jacobian matrix, linearized near the given points of the estimate $\bar{\mathbf{x}}_{k+1}$

EKF Update Step: Whenever a measurement $\mathbf{z}_{k+1}$ becomes available after the prediction step, one applies the update procedure:

$$\mathbf{S}_{k+1} = \mathbf{H}_{k+1}\bar{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{J}_{k+1}\mathbf{R}_{k+1}\mathbf{J}_{k+1}^T \tag{17}$$

$$\mathbf{K}_{k+1} = \bar{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T\mathbf{S}_{k+1}^{-1} \tag{18}$$

$$\mathbf{x}_{k+1} = \bar{\mathbf{x}}_{k+1} + \mathbf{K}_{k+1}(\mathbf{z}_{k+1} - \mathbf{h}_{k+1}(\bar{\mathbf{x}}_{k+1}, \delta_{k+1})) \tag{19}$$

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\bar{\mathbf{P}}_{k+1} \quad (20)$$

where $\mathbf{h}_{k+1}(\bar{\mathbf{x}}_{k+1}, \delta_{k+1})$ is the observation model function, producer of sensor measurements estimates, with $\mathbf{H}_{k+1}$ as the observation model Jacobian matrix, linearized around the estimate $\bar{\mathbf{x}}_{k+1}$. $\mathbf{R}_{k+1}$ defines the measurement model noise covariance matrix, from the Gaussian noise $\delta_k \sim \mathcal{N}(0, \mathbf{R}_k)$. The innovation process $\nu_{k+1} = \mathbf{z}_{k+1} - \mathbf{h}_{k+1}(\bar{\mathbf{x}}_{k+1}, \delta_{k+1})$ which measures the error between measurements and predicted ones, assumes a normal distribution $\gamma_k \sim \mathcal{N}(\nu_k, \mathbf{S}_{k+1})$, where $\mathbf{S}_k$ is the innovation covariance matrix. Much alike the characteristics of $\mathbf{G}_k$, $\mathbf{J}_{k+1}$ is the measurement noise Jacobian matrix, linearized around the state predicted estimates. Finally, the Kalman Gain $\mathbf{K}_{k+1}$ weights the predicted state with the newly observed measurements.

C. Application to the paper context

In the following we describe a simple map building and navigation methodology based on occupancy grids and trajectory planing. The camera is assumed to be a color-depth camera, as the Microsoft Kinect. More details in the next section.

After developing a basic navigation methodology, we study and develop a methodology to use of a simpler camera, namely a standard color camera, hence without depth measurements. In order to compensate the lack of depth measurements we use an EKF based approach, more precisely MonoSLAM. The dynamic motion model is a 6D constant velocity model. In particular we assume an hypothesis on the scene structure to be nearly a tubular shape. This hypothesis allows constraining more the data, and therefore more effective filtering, and in addition to keep a constant state size.

III. Navigation using Color-Depth Cameras

A. Building an Occupancy Grid

The principle idea of the occupancy grid is to represent a map of the environment as an evenly spaced field of binary random variables each representing the presence or not of an obstacle at that location in the environment.

The goal of these algorithms is to estimate the posterior probability over maps given the data: $p(m|z_{(1:t)}, x_{(1:t)})$ where m is the map, $z_{(1:t)}$ are the sensor readings from the beginning until $time = t$ and $x_{(1:t)}$ are the position and orientations of the robot in the same time steps.

The posterior of a map is approximated by factoring it into $p(m|z_{(1:t)}, x_{(1:t)}) = \prod_i p(m_i|z_{(1:t)}, x_{(1:t)})$. Due to this factorization, a binary Bayes filter can be used to estimate the occupancy probability for each grid cell.

B. Occupancy Grids from Depth Images

We will use the depth information now available and the RGB images to build an occupancy grid map.

1) Motion Estimation: We need to estimate where the camera is at all steps to be able to correctly update the occupancy grid. For that we use the RGB images. We will now explain in a more detailed way how that estimation is made.

a) How the camera motion estimation between steps is made: First the SIFT features are computed in the current and previous RGB images. Then a matching algorithm returns the corresponding features in both images. RANSAC algorithm is applied to the matches using a rigid transformation model to discover the inliers in the matches. Using only those inliers we input them in the Procrustes function in Matlab that returns the rigid transformation between the two point clouds (one for each image). This way we improve a lot our motion estimation of the camera and consequently the final map.

2) Map Building: To validate this approach an experiment was conducted. Using the data from a run around a VRML world composed by the four corridors and elevator hall (simulating the 5th floor IST North Tower), we have the projection matrices and depth maps for every step. By choosing the middle line in the depth image and assuming that the camera is horizontal the whole time we can simulate a laser range finder. Then we need to use that information and update our probability grid. Each cell in the grid has a probability value for the belief that it is occupied. The free cells are updated with the following equations

$$Bel\_Occ = (1 - TrueHit) \times p(x, y) \quad (21)$$

$$Bel\_Unocc = (1 - FalseAlarm) \times (1 - p(x, y)) \quad (22)$$

$$p(x,y) = \frac{Bel\_Occ}{Bel\_Occ + Bel\_Unocc} \quad (23)$$

The values for *Bel_Occ* and *Bel_Unocc* are the belief that the cell is occupied or not, respectively. Then we normalize the probability and get $p(x, y)$. The locations where obstacles were detected are also updated with

$$Bel\_Occ = TrueHit \times p(x,y) \quad (24)$$

$$Bel\_Unocc = FalseAlarm \times (1 - p(x,y)) \quad (25)$$

$$p(x,y) = \frac{Bel\_Occ}{Bel\_Occ + Bel\_Unocc} \quad (26)$$

The variables *TrueHit* and *FalseAlarm* represent the probability that an obstacle is detected by the sensor when it is actually there and when the sensor detects something that is not there, respectively. We do this for all the steps and in the end we have an occupancy grid map of the environment.

3) Loop Closure: Loop closure is the process of correcting the map representation based on the estimated position of the robot. Knowing the error present we can correct that estimated position and build a better map.

The adopted solution was: once the loop closure is detected (i.e. a past location is being visited again) and using the projection matrices in every step it is possible to make a profile of the movement of the camera in x, y, z axis for the translation and α,β,γ (the euler angles) for the orientation. Based on this evolution we need to divide our final error through all the steps. If we compute new camera projection matrices with the error correction added and rebuild the occupancy grid, the final error should be smaller. The error is just the difference between your desired and actual end pose. To find the weight of the error to add in each step we used the formula: (this is the formula for x, but it is the same for the others)

$$\begin{aligned} &e_x(t) = e_{x_{max}} \left(-\tfrac{1}{2} + \tfrac{1}{M_x} \int_0^t |v_x(t)| dt \right), \ t \in [0, t_{max}] \\ &M_x \equiv \int_0^{t_{max}} |v_x(t)| dt \end{aligned} \quad (27)$$

where M_x is a normalization constant and v_x is the velocity at that instant.

C. Motion Planning

Motion planning is a term used in robotics for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement.

1) Potential Fields: Potential Fields is a motion planning technique where the robot is treated as a particle placed on an influencing potential field. The topology of the field is created by the designer: it first creates multiple behaviors, each assigned a particular task or function, then represents each of these behaviors as a potential field and finally combines all of the behaviors to produce the robot motion by combining the referred fields.

The action vector, the speed and orientation of the robot, is produced by each behavior based on its purpose. The behavior of the goal state is to attract the agent and of an obstacle is to repel it. The collection of these vectors for all possible locations of the agent is the Potential Field.

Based on this Potential Field the robot knows how to move around. It moves at the speed and orientation of the vector at that location.

2) Randomized Methods for Path Planning: Randomized path planning algorithms were designed for one of two contexts: single-query or multiple-query planning. For single-query planning, it is assumed that a single path planning problem must be solved quickly, without any preprocessing. For multiple-query, many path planning problems will be solved for the same environment and is worthwhile to preprocess information and store it in a data structure that allows fast path planning queries.

3) Probabilistic Roadmaps: The Probabilistic Roadmaps algorithm work in two steps, the first called *learning stage* and the second called *query stage*. Given an instance of the RMP (Robot Mapping Problem) in the *learning stage* the algorithm samples the *configuration space* and builds an undirected graph *G = (V,E)* which captures the information gathered. The graph is called *Probabilistic Roadmap*. In the *query stage*, the PRM is used to solve specific RMP problem instances which are then reduced to a graph search.

The learning stage starts by generating a random configuration $c \in C_{free}$. C_{free} is the set of configurations for which the robot does not collide with

any static obstacle. It checks for nearby vertices where its distance is lower than a certain threshold. Then it connects them by the increasing order of their distance and the planner verifies that the path between them is free.

In the query stage two configurations are given (*start* and *goal*). They both are connected to the graph by the same ascending distance method. They are checked if both belong to the same connected graph component. If so, the path is returned. Possibly, path smoothing is later applied for a better solution.

4) Rapidly Exploring Random Trees:

a) The RRT Method: The RRT is an efficient data structure and sampling scheme to quickly search high-dimensional spaces that have both algebraic and differential constraints. The basic idea is to direct the search in the configuration space for a solution.

It starts by building a tree rooted at the starting point q_{init}. A random configuration q_{rand} is chosen in C_{free}. The current tree is extended from the closest vertex by a fixed step size in the direction of that configuration. Everytime the tree is extended a function checks for collisions between the closest vertex and the new vertex . There are three possible outcomes for this tree extension: the path is clear and the tree is extended in that direction with the new vertex; the path is obstructed by an obstacle and the new vertex is rejected; the random configuration is reached and added to the tree because it is at a smaller distance than the step fixed size. This process is repeated until the goal configuration is reached. Some things to keep in mind is that for this method an efficient neighbor search is important. The metric chosen is also fundamental.

b) RRT-Connect Path Planner: A similar variant of the RRT method is the RRT-Connect. The RRT-Connect planner is designed specifically for path planning problems that involve no differential constraints. The method is based on two ideas: the Connect heuristic that attempts to move over a longer distance, and the growth of two RRTs from both q_{init} and q_{goal}.

Instead of attempting to extend an RRT by a single step size, the Connect heuristic iterates the extend step until *q* or an obstacle is reached. With this method, the greedy heuristic is combined with the rapid and uniform exploration properties of RRTs, which seems to avoid the local minima.

The idea is: two trees are maintained at all times until they become connected and a solution is found. In each iteration, one tree is extended and an attempt is made to connect the nearest vertex of the other tree to the new vertex. Then, the roles are reversed by swapping the two trees. This causes both trees to explore the "free"space around them, while trying to establish a connection between them.

The key result is that the distribution of the RRT vertices converges toward the sampling distribution, which is uniform.

IV. Navigation based in Monocular Vision

This chapter addresses a *Simultaneous Mapping and Localization* SLAM methodology for a system capable of performing visual inspections in an unknown environment domain assumed to be a tubular shaped structure (*TSS*) and a rectangular section tube using a monocular camera.

Estimating the motion of a camera moving inside a shaped structure involves considering various aspects. Two important aspects are the number of degrees of freedom of the camera motion and the shape of the environment structure. A *TSS* includes straight and curved sections. The camera can move in 6D (3D in translation and 3D in rotation). Motion can be estimated by registering the texture, retrieving distinctive visual features, and dewarping into a mosaic. We approached the problem by first reconstructing points of the scene and then focus on fitting a simple 3D cylindrical model to the various tube sections, which makes simple the dewarping step.

A. Depth Information Not Available

The big difference from this method to the one presented in section III is the sensor used. In this method, a simple monocular camera does not give the depth information from its pixels. Because of this a new way of estimating the motion of the camera is necessary. The SLAM methodology is used with some assumptions made *a priori*. A structural model for the environment is defined and an EKF with a motion model and observation model work together to estimate the pose of the camera among other things.

B. Iterative Estimation of Tubular Structure and Camera Motion

Assuming the world as a *TSS* domain navigated by a forward hand-held like monocular camera, smoothly moving with a constant speed inside a tube without revisiting previous positions, one can determine the features 3D locations, which are strapped to a cylindrical section wall with radius ρ. By thoroughly defining a state vector $\mathbf{s}$ composed of both cameras and the *TSS* geometrical parameters estimates, and incorporating a geometrical observation model reliable enough to produce relevant estimations of the cylindrical section geometrical parameters, the simultaneous camera localization and mapping can be achieved.

1) State Vector: The state vector $\mathbf{s}$ is a joint composition of camera pose $\mathbf{x}$ and cylinders geometrical parameters $\mathbf{y}$:

$$\mathbf{s}_k = [\mathbf{x}_k \ ; \ \mathbf{y}_k] \tag{28}$$

where the semicolon denotes vertical stacking of vectors.

Camera motion behavior is modeled with a constant velocity dynamic model. The state vector $\mathbf{x}_k$ provides the position $\mathbf{r}_k^{WC}$, orientation $\mathbf{q}_k^{WC}$ and both linear $\mathbf{v}_k^W$ and angular ω_k^C velocities at every instant, i.e.

$$\mathbf{x}_k = \left[\mathbf{r}_k^{WC} \ ; \ \mathbf{q}_k^{WC} \ ; \ \mathbf{v}_k^W \ ; \ \omega_k^C\right]. \tag{29}$$

A cylindrical section is characterized by a 7 dimension $\mathbf{y}_{nk}$ state vector as an array of the cylinder parameters centre position $\mathbf{p}_{n_k}^W$, orientation $\mathbf{o}_{nk}$, and radius $\log(\rho_{nk})$ expressed in a logarithm form

$$\mathbf{y}_{nk} = \left[\mathbf{p}_{n_k}^W \ ; \ \mathbf{o}_{nk} \ ; \ \log(\rho_{nk})\right]. \tag{30}$$

2) System Dynamics: The non-linear state transition model function $\mathbf{f}$, defined as the compilation of two independent state transition processes, $\mathbf{f}_x$ and $\mathbf{f}_y$, for both camera and cylinder state vector $\mathbf{x}$ and $\mathbf{y}$, which are influenced by an additive zero-mean Gaussian transition noise $\epsilon \sim \mathcal{N}(0, \mathbf{Q}_k)$:

$$\mathbf{f} = [\mathbf{f}_x \ ; \ \mathbf{f}_y]. \tag{31}$$

The constant velocity model allows smooth velocity variations with zero-mean Gaussian distributed acceleration noise $\mathbf{n}_x \sim \mathcal{N}(0, \mathbf{N}_{xk}) = (\mathbf{a}^W \vartheta^C)^T$. Under this model assumption, and defining $\mathbf{V}^W = \mathbf{a}^W \Delta t$ and $\Omega^C = \vartheta^C \Delta t$ as the linear and angular velocities impulse between a transition step Δt, one can express the camera state transition, $\mathbf{x}_{k+1} = \mathbf{f}_x(\mathbf{x}_k, \mathbf{n}_{xk})$ [1], [3]:

$$\begin{bmatrix} \mathbf{r}_{k+1}^{WC} \\ \mathbf{q}_{k+1}^{WC} \\ \mathbf{v}_{k+1}^{W} \\ \omega_{\omega+1}^{C} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_k^{WC} + \left(\mathbf{v}_k^W + \mathbf{V}^W\right)\Delta t \\ \mathbf{q}_k^{WC} \times \mathbf{q}\left(\left(\omega_k^C + \Omega^C\right)\Delta t\right) \\ \mathbf{v}_k^W + V^W \\ \omega_k^C + \Omega^C \end{bmatrix}. \tag{32}$$

3) Observation Model: The observation model describes the process of implicitly representing observed features Λ in a 3-D parameterization as a function of the systems state $\mathbf{s}$ parameters by inferring constraints to the environment structure and consequently to the features 3-D locations. At an instant k the state transition function $\mathbf{f}$ is applied to $\mathbf{s}_k$, retrieving $\mathbf{s}_{k+1} = \mathbf{f}(\mathbf{s}_k, \mathbf{n}_k)$. Λ_k is the input set of features in pixel coordinates acquired at the instant k, whereas $\widehat{\Lambda}_{k+1}$ is the set of features in following instant $k+1$ and the observation model output. The observation model can thus be written as a function $\mathbf{h}$:

$$\widehat{\Lambda}_{k+1} = \mathbf{h}_{k+1}(\mathbf{s}_{k+1}, \Lambda_k) \tag{33}$$

Consider that the set of features Λ_k, 3-D locations on the *TSS* surface, are the intersections of rays leaving the camera centre $\mathbf{r}_k^{WC}$ with the cylinder wall, piercing the 2-D image plane where all features lie. With the information present in the state vector $\mathbf{s}_k$, i.e. camera pose $\mathbf{x}_k$ and cylinders section parameters $\mathbf{y}_k$, one can estimate the 3-D location of every feature in set Λ_k through a back-projection methodology, and compute the re-projection of this 3-D coordinates with the next instant $k+1$ state vector $\mathbf{s}_{k+1}$ to a 2-D image plane, thus acquiring $\widehat{\Lambda}_{k+1}$ features.

C. Dewarping

The process of opening the tube or dewarping the interior of the tube is done by finding the transformation between the pixels in the scene viewed by the camera inside the *TSS* and the planar mosaic we want to build of the inside texture. In essence is a conversion from polar coordinates to cartesian coordinates.

1) Dewarping a VRML Tubular Shape: Datasets with real images, acquired from virtual or real cameras, introduce image processing frameworks to the developed system, incorporating image description techniques such as features descriptor algorithms (e.g. SIFT, SURF), enhanced by the RANSAC methodology for outliers removal. Figure 1b shows the dewarping of an image taken inside a VRML tubular shape.

D. Rectangular Section for a Corridor

To adapt this model from a tubular shape to our corridor model we had to make some changes. The idea was to switch the back-projection function to a rectangular section instead of a circular one.

1) Results obtained with one corridor: For this dataset, a VRML model was used and the camera went from $(1, 1, 2)$ to $(1, 1, 12)$ so it traveled 10 meters in the z direction. There were 80 steps with a step size of 0.125 m. The resulting plot for the camera poses in world coordinates is in Figure 1c. Figures 1d and 1e show how the state of the filter evolved over time, for position and orientation.

We can conclude by watching the plot that the camera moved essentially in the z direction as it was expected. The drift in the orientation of the camera can be associated with the noise present in the quality of the features detected.

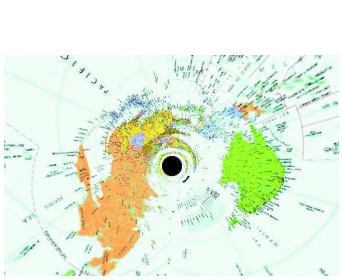
V. Experiments

To validate the mapping and navigation methods studied and introduced in previous chapters, a few experiments were conducted to materialize and put together all these ideas.

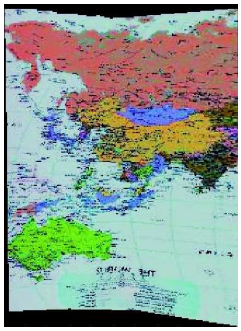
A. Navigation using Color-Depth Camera

A virtual world created in VRML is used for purposes of visualization. An example of a POV from the robot inside the virtual world in consecutive steps can be seen in Figure 1f and Figure 1g. Several 3D points coinciding with the VRML corners make up the facets used to simulate the depth sensor. The virtual world has textures in its walls that we will use to scan for visual features (SIFT).

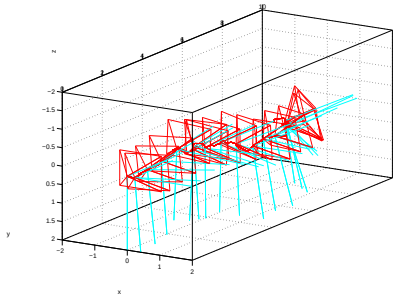
In this experiment we will simulate the navigation of the robot. Using the map achieved by the mapping procedure, we plan our path and send that information at each step to the robot. At each step



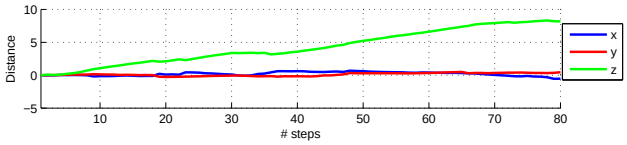
(a) View of the camera inside the tube



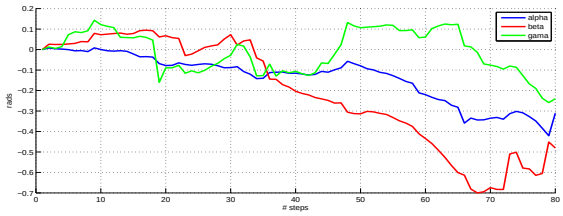
(b) Result of dewarping several steps



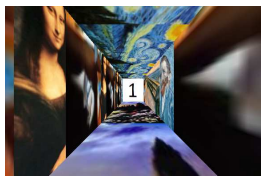
(c) Camera poses returned by the filter



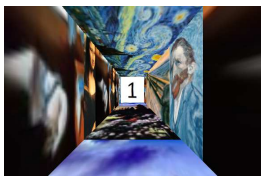
(d) State of the filter for the position of the camera



(e) State of the filter for the orientation of the camera (euler angles)



(f) Step 3



(g) Step 4

Fig. 1

we simulate our sensor as a rangefinder. Our Kinect is used as a LIDAR to detect walls or other obstacles that may traverse our path. A list of 3D points that comprise the facets where our obstacles are simulated is used. When something is detected it is computed its position in the global map using the current estimate of the robot position. Anything that crosses our path is considered an obstacle. Once our robot detects something in its path, it stops and recomputes the path considering the new map with the obstacle in it.

We start by doing some processing to the occupancy grid matrix. This matrix is saturated with a chosen threshold (25%) and we end up with a bitmap with the same size only with zeros and ones. After that, the map is dilated to take into account that the robot is not just a point in the map. Its body has a certain size so the obstacles (walls) are enlarged by 3 pixels $(30cm)$ to take in consideration the robots radius.

We use this map as one of the inputs to the sentinel mode function. The other necessary inputs are the initial pose of the robot, the list of points of interest where we want our robot to go through and the list of facets used to simulate the kinect in measuring depth information.

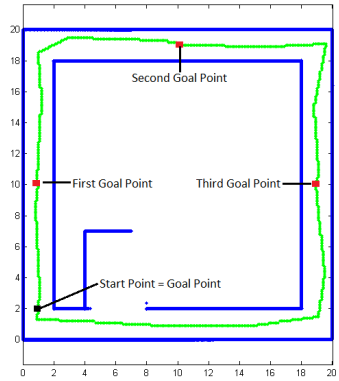
For this first experiment no obstacles were put in place. The list of points that the robot had to visit were: it starts in $(x = 2; y = 1)$ and has to visit in the following order $(10; 1) \rightarrow (19; 10) \rightarrow (10; 19)$ and then return to the starting point $(2; 1)$.

The method chosen for the path planning algorithm was Probabilistic Roadmaps.

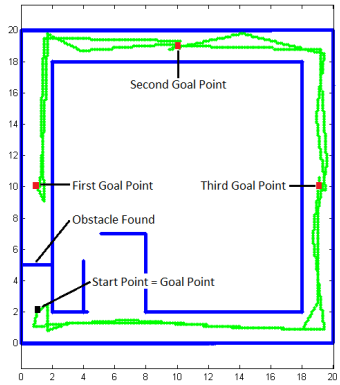
In Figure 2a it is depicted the total path the robot traversed.

In this experiment, we successfully completed our objective of visiting a list of places in the map and also returning to the initial location.

In a second experiment we repeat the same list of points of interest to tour except this time we put an obstacle somewhere in the map. The sentinel function works exactly the same like before except that when a mission detects an obstacle in its path it aborts. In these cases the sentinel function receives an aborted mission and has to re-plan a new path considering the obstacle detected. The obstacle used in this experiment was placed in a manner that blocked completely the first corridor. The method



(a) Total path the robot made with the interest points on the map with no obstacles.



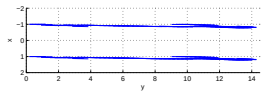
(b) Total path the robot had to make to visit all the points with one obstacle.

Fig. 2: Total path

chosen for the path planning algorithm was again Probabilistic Roadmaps. The total path traversed by the robot can be seen in Figure 2b.

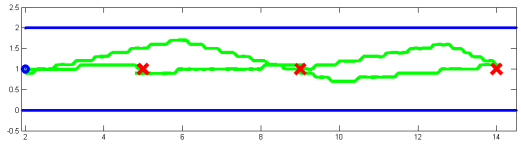
B. Mapping and Navigating with Tube-VSLAM

To test the Tube-VSLAM method for creating a map a test was conducted using the approach explained in section IV. In this test, the camera moves in one corridor and based on the information retrieved by the filter a sketch of the environment is made. It results in a series of line segments that are positioned in the map according to the filter's position of the camera in relation to the walls (Figure 3a). We use those lines and the position of the camera to create and update our occupancy grid map. The result is straightforward and is visible the corridor in the end map in Figure 3b.



(a) Sketch composed of line segments corresponding to parts of the walls returned from the filter for one corridor.

(b) Occupancy grid for one corridor using VLSAM approach.



(c) Total path of the robot after completing all the missions.

Fig. 3: Maps for one corridor.

Using that map we can navigate inside our structure and visit a sequence of points of interest much like taking a tour. The list of points followed were: $(1,5) \rightarrow (1,9) \rightarrow (1,14)$ and then returning to the starting point $(1,2)$. The trajectory planning method chosen was Probabilistic Roadmaps. The total path can be seen in Figure 3c.

VI. Conclusion and Future Work

The work described in this thesis aimed at developing and simulating a robotic sentinel that first, maps an indoors environment and secondly, is able to navigate through the environment, computing alternative trajectories when obstacles block the previously computed trajectories.

The first video camera considered was a Color-Depth (RGBD) camera. With the depth information available we were able to infer the 3D position of the features. By using consecutive image frames we estimated the motion of the camera as a rigid transformation. The process is the following: First SIFT features are computed between consecutive images and matched; Then the matched 2D features are converted to 3D, in the camera coordinate frame; The next step consists simply in computing a rigid transformation between both clouds of points; The computed transformation is the (inverse) motion the camera made to capture the two images.

In addition to estimate the motion of the camera, the color-depth camera allows computing an horizontal depth profile, which can be integrated along time to form an occupancy grid. Given the occupancy grid, allows planing trajectories.

The second video camera considered was simply a Color (RGB) camera. Given the lack of depth information, we have introduced an Extended Kalman Filter based in the camera model and a constant velocity motion model, forming therefore a visual-SLAM framework. In particular we have considered that the scenario can be described by a tubular shape, which gives further consistency to the measurements. Considering the tubular shape hypothesis has the additional benefit of saving and keeping constant the computer memory, as the parameters describing the tube are much less than the number of registered features.

Both types of cameras allowed in the end to compute an occupancy grid and planing trajectories. Navigation results were promising in both cases as the robot visits all the predefined locations and, if an obstacle blocks its path, it tries to go around and reach the objective through an alternative path.

References

[1] Ted J. Broia and Rama Chellappa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE T-PAMI*, 13(6):497–513, June 1991.

[2] Ted J. Broia, Rama Chellappa, and S. Chandrashekhar. Recursive 3-d motion estimation from a monocular image sequence. *IEEE Transactions on Aerospace and Electronic Systems*, 26(4):639–656, July 1990.

[3] Javier Civera, Andrew J. Davison, and J.M.M Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24(5):932–945, October 2008.

[4] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE T-PAMI*, 29:2007, 2007.

[5] João Pedro Borralho Tomaz. Visual self-localization and mapping using the inside texture of a tubular structure. Master's thesis, Instituto Superior Técnico, 2013.

[6] Gem-Sun Young and Rama Chellappa. 3-d motion estimation using a sequence of noisy stereo images. *IEEE T-PAMI*, pages 710–716, June 1988.