Unmanned Aerial Vehicle for Autonomous Navigation In Cluttered Indoor Invironments

Niraj Kumar[1], Vasu Agarwal[1], Divya Dhawan[1], Yash Agarwal[1], Ayush Trivedi[1], Amandeep Singh[1], Nikhil Jain[2], R.D. Roychoudhury[2], Andra Aditya[2], Mohit Jain[2]
[1]University of Petroleum and Energy Studies, Dehradun.
[2] Indian School Of Mines, Dhanbad.

Abstract- This paper describes a four rotor vertical-take-off-and-landing unmanned aerial vehicle, commonly known as a Quadrotor, designed for objective completion in an urban environment. The assumptions, proposed design and project methodology is described in detail.

1. Introduction

This section introduces the overall project and the concepts that are involved.

1.1. Goal

To design and build a small UAV that is capable of competing in the International Aerial Robotics Competition 6th Mission.

1.2. Personal Statement of Interest

Unmanned aerial vehicles (UAVs) are the future of aviation industry. UAVs provide unparalleled performance for dangerous situations, both for military and civilian applications. We can provide new dimensions to the UAV technology while working on this project, which will require substantial work in a variety of engineering fields. It is a comprehensive topic, which gives a chance to learn a lot in the interdisciplinary field of aerial robotics.

1.3. The International Aerial Robotics Competition

The International Aerial Robotics Competition (IARC) is an international competition sponsored by the Association for Unmanned Vehicle Systems International (AUVSI). Competition will celebrate its 21st year of competition with the 6th Mission. In this new 6th mission, a small UAV will be launched to infiltrate a hostile building. Once inside, the UAV is to find and acquire a specific small object without being detected by video surveillance. This is to simulate a covert operation for military usage. These competitions are designed to be challenging and generally beyond the abilities of the current technology. The 4th mission lasted for 8 years, and was never fully completed

1.4. Approach

Simultaneous Localization and Mapping (SLAM) in any robot forms an indispensable part wherever locomotion is involved. Primary objective of this system is to provide the robot with its relative position and secondarily, it takes *the data from the surroundings through sensors and camera placed on board and builds a virtual environment in* where the robot can maneuver. Stereo Imaging is a technology that closely resembles human vision. Using two cameras at a fixed distance from each other and comparing the two images, we create range data. This coupled with laser ranging can be used for creating accurate maps. GPU is a hardware which is also used for image processing thus speeding up processing by an enormous factor. It uses GPGPU computing methods which have proven to be reliable and efficient. Distributing the work on an onboard and off board processor depends on the throughput of Wi-Fi and required accuracy. For a particular task, this gives us a versatile solution where accuracy and speed can be traded for each other. Done efficiently and which much ease without wastage of resources.

2. Problem Statement

The sixth mission of the International Aerial Robotics competition requires that a UAV weighing less than 1.5kg has the ability to enter and navigate within an unknown confined environment in search of a specific marked target without being detected. The mission also requires that the MAV locate and bring back a flash drive.

3. Unmanned Aerial Vehicle

The Quadrotor is a mechanically simple vehicle, requiring only four stationary motors to achieve all flight motions. Despite its simplicity, it is very capable of advanced flight. The key to flight movements beyond hovering is in differential thrust. A simple diagram is seen in Figure 1: Quadrotor Body Diagram.

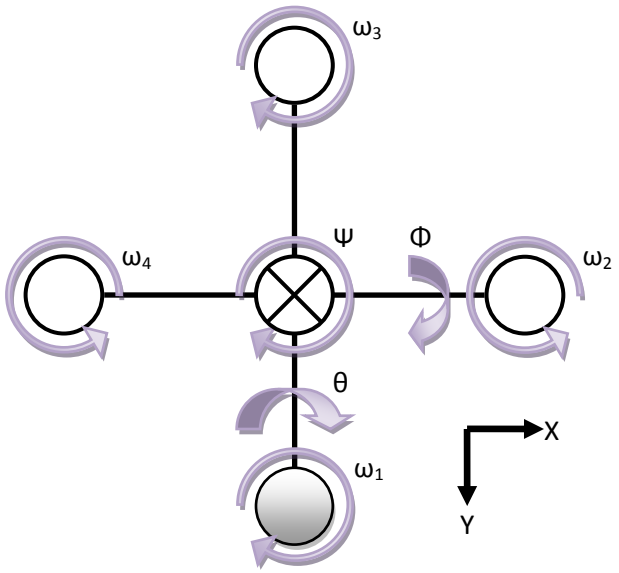


Figure 1: Quadrotor Body Diagram

In the above diagram, where ω_i represents the angular velocity of rotor *i*, the rotation movements roll, θ, pitch, Φ, and yaw, Ψ can be described as functions of ω_i. ω_1 and ω_3 rotate clockwise, while ω_2 and ω_4 rotate counter-clockwise. This balance of rotation allows the Quadrotor to counteract torque from rotor blades. To rotate the Quadrotor in the θ direction, angular velocity ω_4 is increased and ω_2 is decreased. To maintain a stable altitude, ω_1 and ω_3 will have to be increased slightly as the Quadrotor will tilt and start to fly sideways along the X-axis. Differentiating ω_1 and ω_3 will result in movement along the Y-axis. Movement in the Z-axis is simply control by the combined thrust of all the rotors. Yaw movement around the Z-axis is done by differentiating the angular velocities of the pairs of contra-rotating motors. To rotate the Quadrotor clockwise about the Z-axis, the angular velocities of ω_1 and ω_3 need to be decreased while ω_2 and ω_4 need to be increased.

The thrust force from each motor can be described to be related to the size of the rotor blade and the torque caused by the rotor in the air. There are several variations of this formula, and it is premature to decide which variation will be used in this project. The development of these formulas will be critical in developing the aerodynamic model used in simulations.

The control system will have to be an under actuated system. An aerial vehicle composes of at least 12 state variables, three degrees of freedom (DOF) in each position, position rate, rotation, and rotation rate. Inputs, on the other hand, only consist of the acceleration of position and rotation in the three axes. In addition, the flight dynamics of UAVs are typically nonlinear due to the aerodynamics of the vehicle flight as well as from coordinate transforms.

In all implementations, the movement of the Quadrotor is separated into two control systems: attitude and altitude. Attitude refers to the rotation of the Quadrotor, while altitude is specifically for up/down movement. The attitude controller is responsible for the bulk of the changes in motor thrust control. However, the altitude controller is valuable to compensating for the force of gravity as the Quadrotor rotates roll or pitch.

The traditional solution for the control system is a monolithic linearization approach. While this method has its drawbacks, it has been shown to be more than adequate in several papers. The PID control system of a Quadrotor is the most commonly used system in hobbyist Quadrotors as well because of its simplicity. However, the STARMAC group point out that this linear approximation is only suitable for hovering and moving at low velocities. At higher velocities, the flight dynamics begin to resemble an airplane, which necessitates a different approach for aggressive maneuvers. They demonstrated an updated linear controller for aggressive maneuvers in a subsequent paper.

4. Modular design

The short-term goal progression of the project facilitates the modularization of the Quadrotor. Once a system has been completed to satisfaction, it is designed to not need further adjustment. A diagram of the modular design of the Quadrotor system is seen in Figure 4: Modular Design.

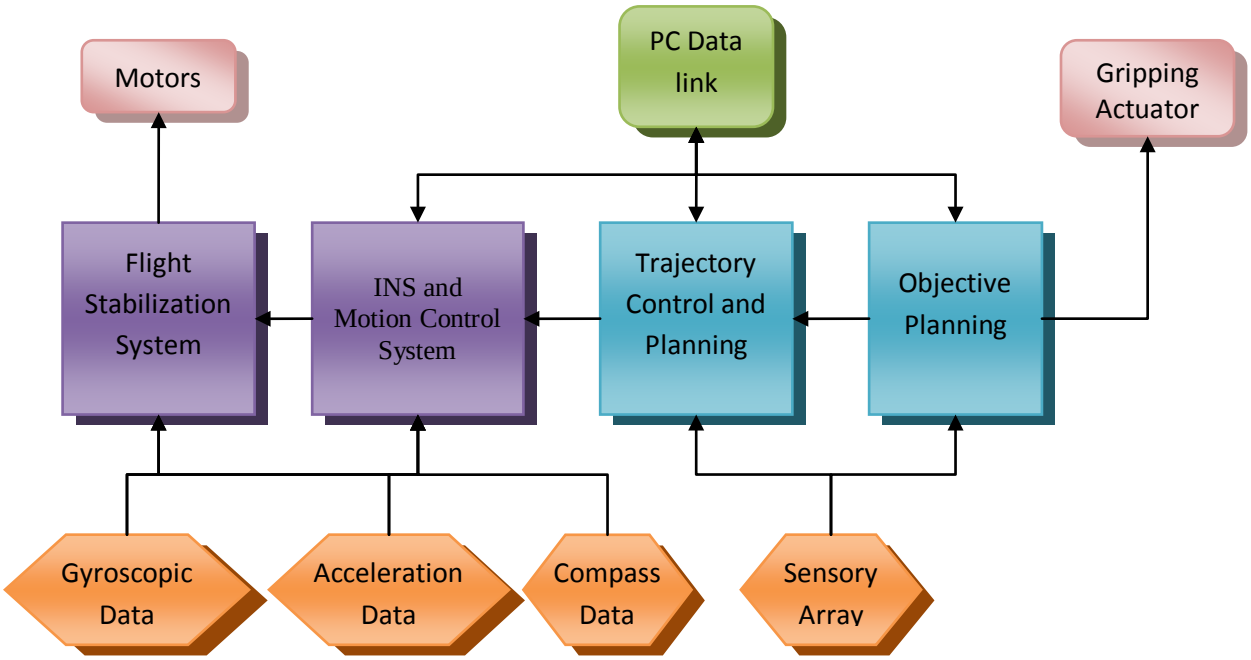


Figure 1: Modular Design

The flight stabilization system takes values from the accelerometers and gyroscopes to maintain a stable flight in regards to the desired direction. Not pictured is the manual control input to the flight stabilization system that would dictate desired direction. The INS and Motion Control System also require the accelerometer's and gyroscope's data as well as navigational information based off an electromagnetic compass. This system controls the basic movement of the Quadrotor and provides live data to the ground station. The sensory array will provide information for the next stage for trajectory control to avoid obstacles. The final component for objective planning will largely be computed by the ground station, though an overlying system for interpreting objective commands is needed on the Quadrotor. This system also includes the control system for controlling the acquisition of specified objects.

4.1. Inertial Navigation System

The Inertial Navigation System (INS) comprises of a 9 degree of freedom (DOF) sensor array, measuring acceleration in x, y and z directions and rotation in roll (θ), pitch (φ), and yaw (ψ). There are a variety of accelerometers, magnetometers and gyroscopes that can be assembled into a 9DOF system.

While accelerometers and gyroscopes are commonly used in this way, they frequently are affected by drift error caused by the integration, which increases drastically over time. Many systems use GPS as a way of keeping the error low, however, due to the competition rules, GPS will not be used. This limits the effectiveness of solely using accelerometers and gyroscopes for positioning information. In this project, an additional set of magnetic compass sensors will be used. This helps reduce the yaw drift error.

5. Visual Odometry and Target Recognition

The vehicle cannot rely solely on positional data derived from motion sensors. Accelerometers are highly susceptible to noise due to vibrations in the frame. Gyroscopes need to be calibrated in regular intervals due to its inevitable tendency to drift.

To calibrate the motion sensors in flight and also to provide accurate localisation, visual odometry is used. The motion estimation is done by using a pair of stereo cameras. This is a better option than using only one camera a in since, sometimes, the motion of the flying vehicle might be too slow or fast to give adequate baseline for reconstruction of the camera positions in space. Also, multi view 3-D reconstruction [1] requires a sequence of overlapping images(seven or more) before it can provide any information. A pair of stereo images offers a sufficiently wide baseline for computation of image points in space.

5.1. Camera Calibration

In case the distance between any two points is not known, the position of the vehicle and its motion can only be determined up to an arbitrary scale factor. For that, the cameras are calibrated after they are attached using OpenCV routines that compute the camera intrinsics and the rotation and translation of the cameras with respect to each other. The calibration uses chessboard patterns having fixed square sizes which define the units in which every distance is measured. The camera model is assumed to be a pinhole camera model[2] with lens distortions taken into account. That is, a scene view is formed by projecting 3D points into the image plane using a perspective transformation. Multiple views of the chessboard in different orientation and distances give data to calculate the camera intrinsic matrices and distortions.

5.2. Feature Extraction

To achieve real time performance, sparse stereo is used. SURF[4] keypoints are chosen because of their robustness under rotation and scaling of a particular point. The keypoints are extracted from both images in the pair. Since the number of keypoints can be high, we used an approximation, the Best-Bin-First (BBF) algorithm[5] to match them from a database. The SURF keypoints are fairly expensive in terms of computation time. Hence, they are computed on a GPU with OpenCV. This speeds up the keypoint extraction by approximately 10 times than its implementation on a standard CPU. The features detected in this stage are further used for recognition of signs and the portable USB Drive.

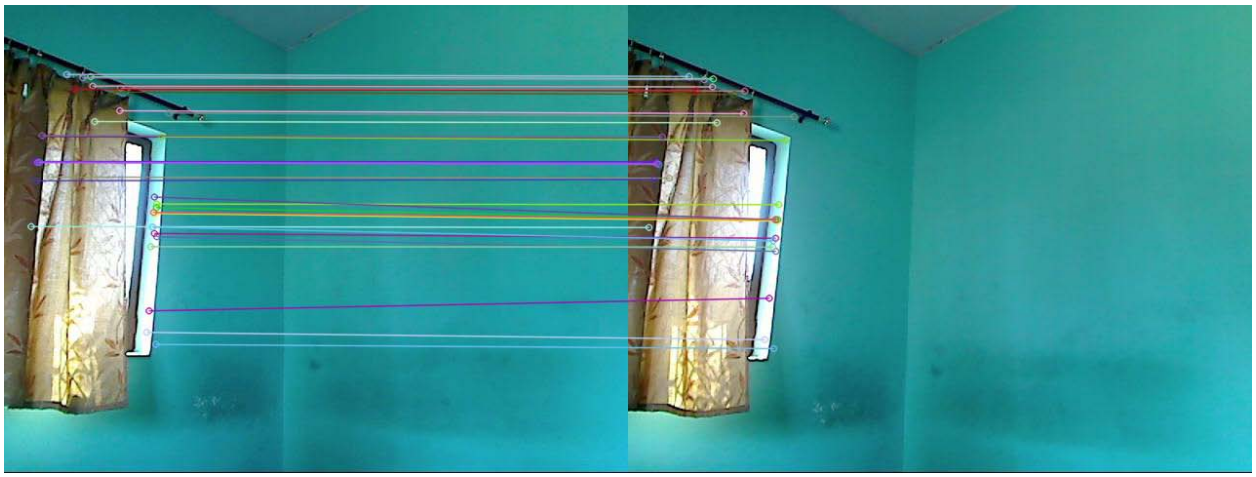
5.3. Stereo Matching

From the two sets of keypoints obtained from the image pairs, correspondences are found by FLANN[10] search algorithm. The pair is called a match only if the next nearest correspondence has a Euclidean distance of more than 1.5 times that between the matched pair. The matched point sets are used to find the fundamental matrix between the two cameras which is calculated using the iterative RANSAC algorithm. The number of minimum required correspondences is 8.

The epipolar geometry is described by the following equation:

$$[\boldsymbol{p_2}; 1]^T \boldsymbol{F} [\boldsymbol{p_2}; \mathbf{1}] = \mathbf{0}$$

Where **F** is fundamental matrix, $\mathbf{p_1}$ and $\mathbf{p_2}$ are corresponding points in the first and the second images, respectively. Since, the real data is not free of noise, there is very less probability of obtaining an exact solution. We find a solution in the least square sense. The RANSAC algorithm is iterative in nature; hence it is done in parallel on a separate thread and does not hinder other computational tasks. Fundamental matrix is calculated at regular intervals so that vibrations of the vehicle or less number of keypoints detected in one frame do not affect the entire navigation. The camera intrinsic are assumed to be same for the entire run which is a fairly valid assumption. Next, the keypoint locations are rectified so that the epilines are parallel to the image horizontal axis. The pixel disparities are calculated as pixel distances between matches from one image to another of the pair.



Correspondences between the SURF keypoints detected in both images

5.4. Depth Reconstruction and Motion Estimation

Stereo calibration gives a reprojection matrix whose inverse is the projection matrix[3] that directly relates the correspondences to the distances of the features in world coordinates. The reprojection matrix is a 4x4 matrix which yields the homogeneous coordinates of a point in terms of the distances we choose to specify for the chessboard blocks during calibration. The disparity information 'd' from stereo correspondence is given as $[x\ y\ d\ 1]$ and the reprojection matrix **Q**.

$$\boldsymbol{Q}\begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

The 3D coordinates are then (*X/W, Y/W, Z/W*).

This data gives local coordinates of particular points with respect to the vehicle. This data fused with the point cloud obtained from the laser sensor is fed to the SLAM algorithm to generate the location and motion data.

5.5. Landmark Recognition

This part is needed specifically for detection of sign boards, lasers, and camera LED. This is done in three stages.

5.5.1. Simple blob detection[6] algorithms implemented in OpenCV search for a region in the image with similar colour. We use the HSI colour model to deal with the effects of illumination and hue individually.

5.5.2. If a sign board is detected, Canny[7] edge detection is used on the Hue component of the image only around the detected region to find its bounding edges followed by Hough Transform[8] to find bounding lines. All the intersecting points of the lines that bound the region given by blob detection are the probable corners of the boards. The corners are further checked for being an affine transform solution in the least squared sense of either a rectangle or an equilateral triangle (The shapes of the sign boards).

5.5.3. If it is an LED which will be apparent if the Hue detected for a blob is close to the blue region and the intensity is higher than the surrounding area by a particular threshold, its position and state is reported back. In case its position changes abruptly (in case of a false detection) or it is not detected for more than 1 second, it is reported to be switched off.

5.5.4. Lasers are by far the most difficult task to be detected. Hence, switching them off is a better alternative. However, if that is not possible, a smoke mechanism along with edge detection and blob detection for colour is used.

5.5.5. After the blob detection, The USB Drive is detected by extracting and matching the SURF keypoints in that region to a database of similar USB drive images. A point is matched with the database, only

if its Euclidian distance[9] to the second nearest keypoint in the database is more than 1.5 times its distance to the match. The fact that the orientation of the detected keypoints in the image must be an affine transformation of the database images is considered. Also, geometric constraints are taken into account such that it cannot be very close to the walls or not on a surface.

6. Simultaneous Localization and Mapping (SLAM)

SLAM which is also known as CML (Concurrent Mapping and Localization) forms an indispensable part of any autonomous moving system. The SLAM technique used in our robot is EKF SLAM in 6d with a graph SLAM approach. Due to the presence of too many variables in the state vector for a UAV, the aforesaid method was found appropriate taking into consideration the amount of time required to converge.

The EKF-SLAM has two important steps. The first step is to create a map using the measurements from sensors such as LIDAR and camera of the landmarks. SLAM using range data from LIDAR derives its essence from DP-SLAM [12]. This is followed by a motion and subsequent observations of readings from the sensors of the same landmarks using the motion update and measurement update of a Kalman filter. Thus, this step helps in increasing the belief in Landmarks and also in its current state. The landmarks arrangement is done in order to obtain the wall and obstacle positions. The M-SLAM model for quick map search is used [11].

The range scan is obtained from the LIDAR, from which the landmarks, basically the corners of the room are derived from the maximas from the range scan. The details of the landmarks are enhanced using visual data, the belief of which is increased with the increasing number of iterations. The initial data received from the range sensors is used to deduce an initial physical map. Then the next reading helps the quad-copter to localize itself in the map simultaneously increasing the belief of the present state and also of the landmarks, which is the essence of SLAM. The landmarks are stored with respect to their coordinates globally, which also includes the Minkowski addition of the quad-copter dimensions to that of the obstacles. This helps us to assume the robot as a point and becomes easy to test a path and other analyses. Graph SLAM based SLAM is used, where essentially two states (present and the previous) are stored along with all the landmarks. An algorithm for landmark management is used which associates different landmarks as wall, obstacles, their position on map and their belief.

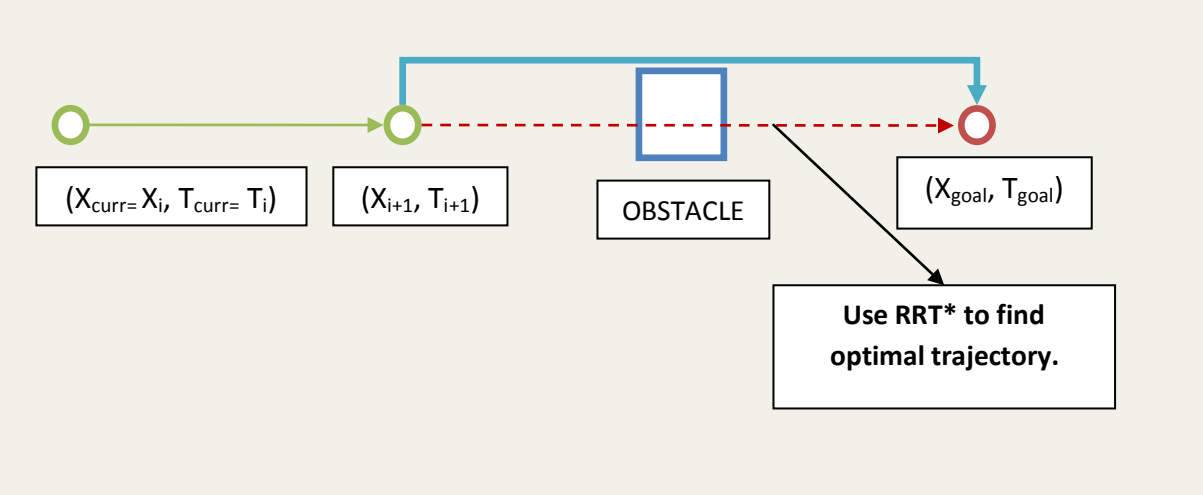
7. Path Planning

Determine optimal FOV (Field of view) that encompasses the centre and the walls. Thus once the room is explored the bot can use its resources into following walls that have a probability of an exit into another room yet unexplored. Follow walls that have a probability of having an exit. Denote these walls by Walls*. Thus the algorithm focuses on moving along walls where there is a chance of an entrance to another room for further exploration.

Formally, Walls* = {w Є W: ρ (w, door) > 0, W is the set of all walls enclosing the room}

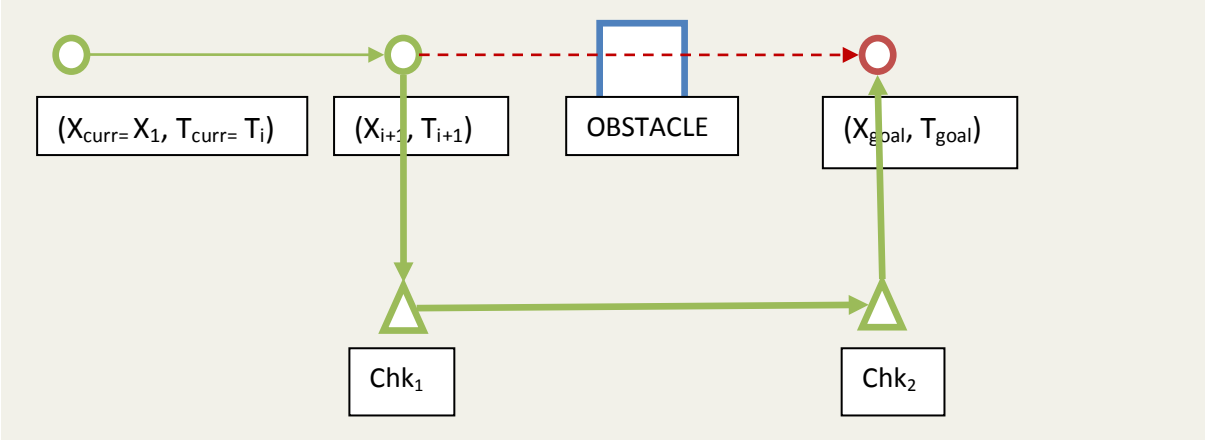
Build a map of all possible exits in a room. Exit the room only if the probability of finding the goal in the room is less than a certain threshold.

Formally, exit room iff ρ (room, goal) < φ (H); where φ (H) is a threshold that depends upon a heuristic H.

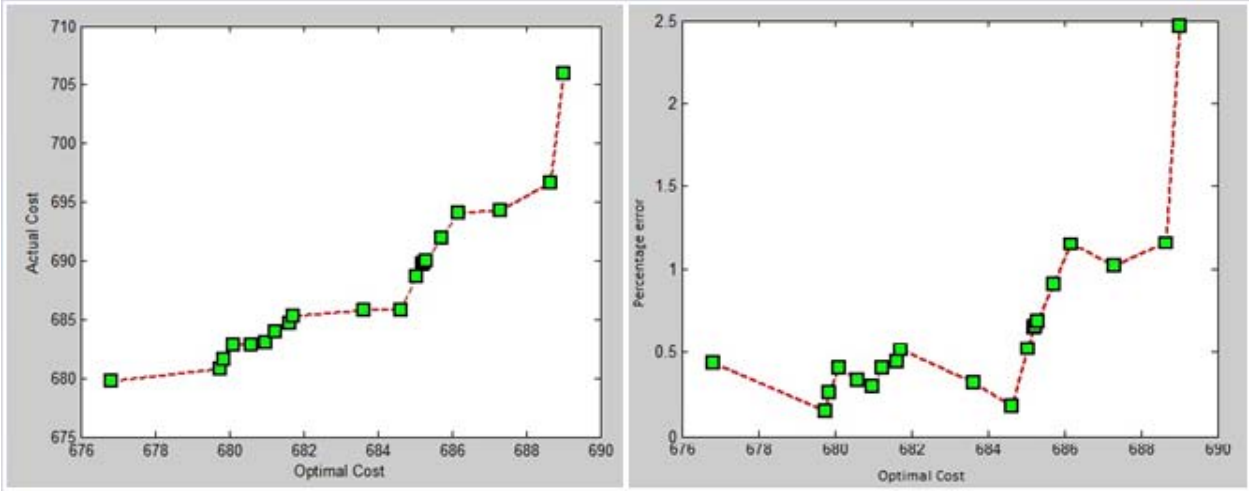


In case of partially known obstacles try to find local trajectories that maximize D_{thresh} and exploration around the obstacle. Thus, the bot tries to gain the maximum insight into finding its way around the obstacle through finding a relatively optimal trajectory in minimal time.

Where, D_{thresh} = Minimum distance to avoid the obstacle



Chk_1, Chk_2 such that: $\delta(X, O) > D_{thresh} + \tau \times v(S_{i+1})$; X= checkpoint. τ= computation time and v is the velocity at state i+1.



Local path planning in clustered environment using RRT*[13]

Maximize trajectories which provide the bot a higher degree of manoeuvrability, to avoid situations that trap the bot into a singular state. The bot moves in a fashion that avoids any decisions, however optimal (locally), that might lead it to a dead state.

Formally, manoeuvrability f_η of a node/sub-trajectory n, is defined as:

$$f_\eta(n, k) = \sum_i (\Phi_i(n \rightarrow \delta(n, k)),$$

Where $\delta(n, k) = \{v \in V : \phi(n \rightarrow v) \leq k\}$,

Φ is the feasibility function,

ϕ is the minimum distance between two nodes,

And k is a threshold.

The cost function defines the cost of reaching a node or pursuing a trajectory. This cost depends upon several factors that need to be either maximized or minimized. We take three factors into consideration, namely Time (T), Energy (E) and the manoeuvrability (f_η). The time and energy spent need to be minimized while the manoeuvrability needs to be maximized.

Formally, the cost function[14] $\Upsilon(T, E, f_\eta) = (C_1 \div a_1) \times T + (C_2 \div a_2) \times E + (C_3 \div a_3) \times (1 \div f_\eta)$

Where T is the time needed to make the transition,

E is the energy spent in the transition,

f_η is the amount of freedom the bot enjoys after the transition,
C_1, C_2, C_3 are constants determined by the weight of respective cost,
And a_1, a_2 and a_3 are the normalization factors determined by the largest estimated cost.

The feasibility function gives an estimate of the feasibility of pursuing the trajectory. The feasibility gives an idea about the amount of time the trajectory can be pursued without being encountered by an obstacle.

Formally, the feasibility function Φ is defined as:

$$\Phi(n_i \rightarrow n_{i+k}) = \sum_k (\Phi(n_{i+k} \rightarrow n_{i+k+1}))$$

. Therefore Φ can be defined recursively.

$$\Phi(n_i \rightarrow n_{i+k}) = \Delta$$

if a unit-route exists between the two nodes.

$$\Phi(n_i \rightarrow n_{i+k}) = 0$$

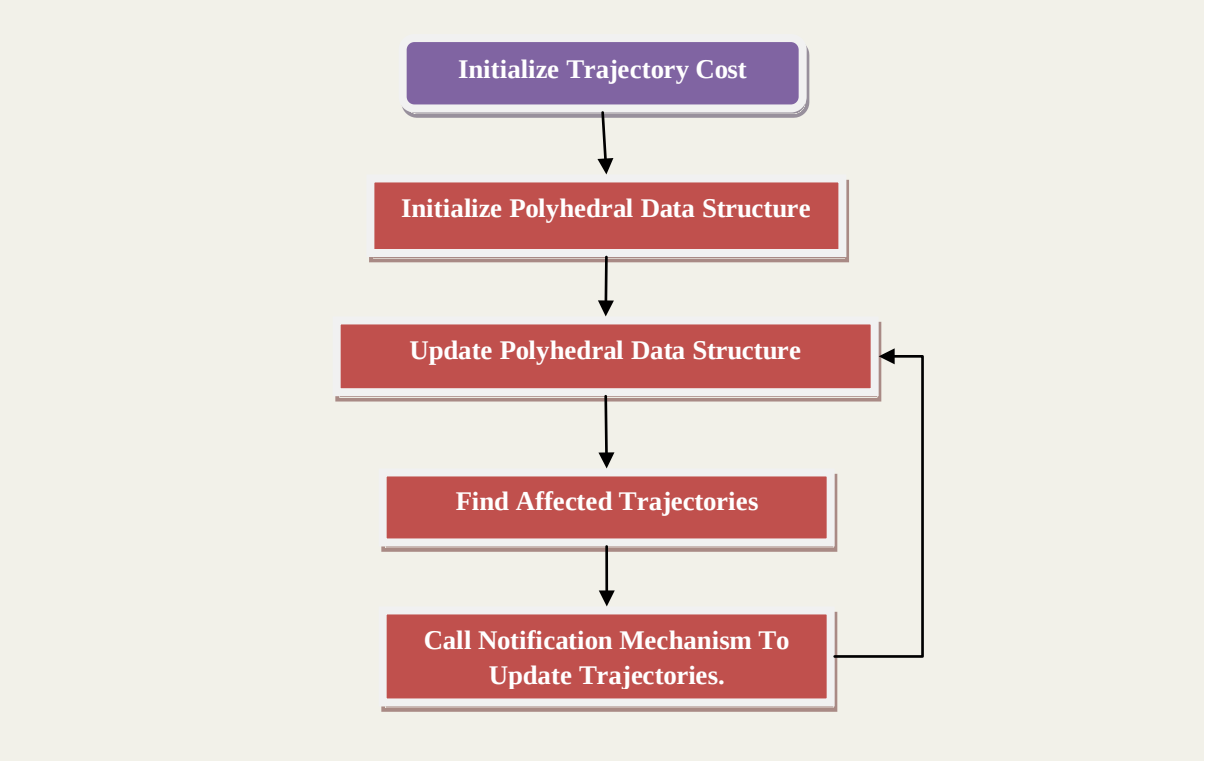
, otherwise.

Where Δ is an appropriately chosen constant.

Initially according to the sensor data all the trajectories are infeasible with large cost values. This is because the major part of the arena is unexplored and not much information about the configuration of obstacles inside the room is available. The algorithm works by selecting the least relative cost amongst all the trajectories so the high cost values pose no threat.

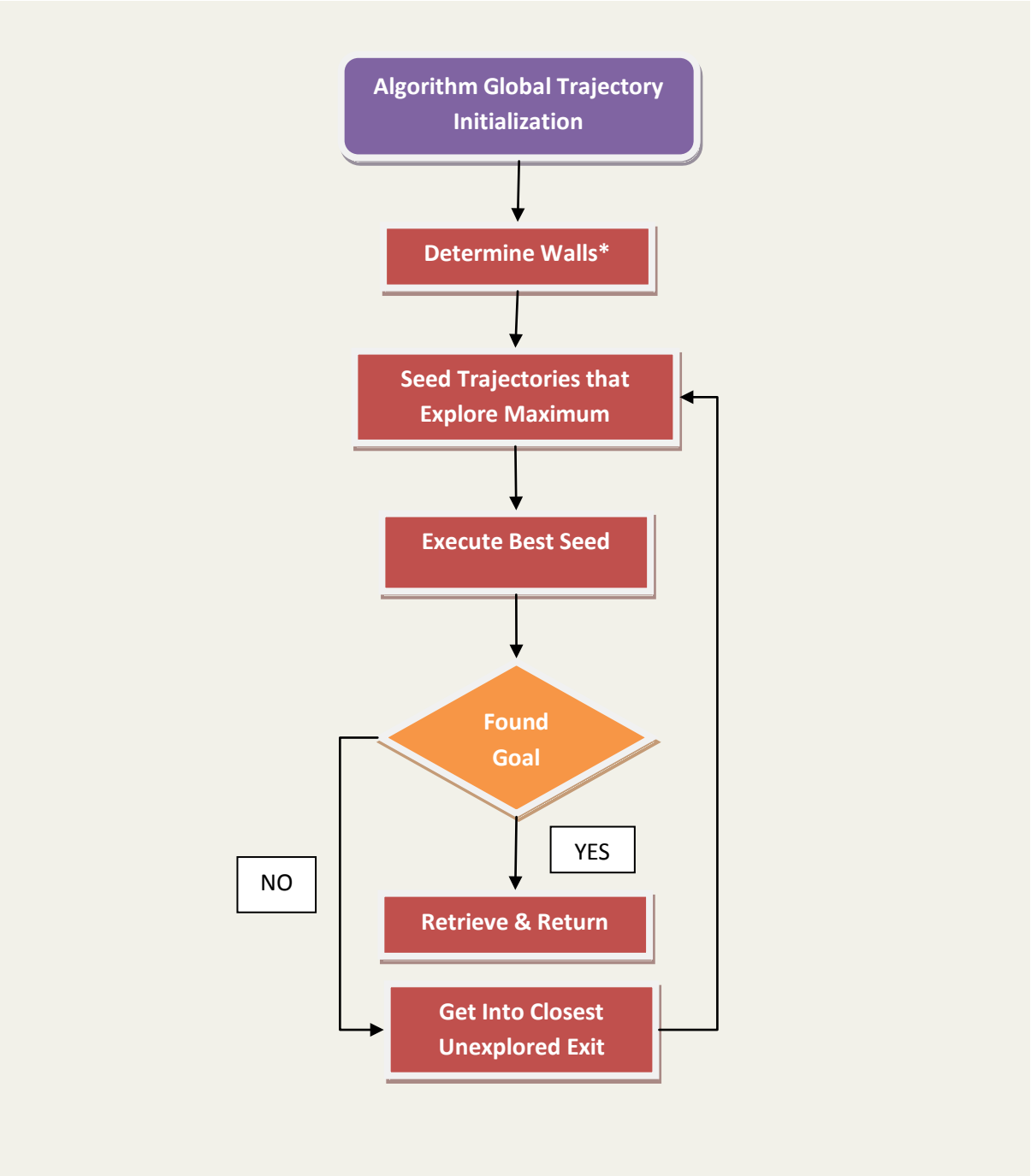
Develop an event handling style algorithm that notifies the system when there is a change in the polyhedron data structure when the sensor detects presence of new obstacles based on the current FOV (Field Of View).

The algorithm then proceeds into finding the trajectories affected locally due to the gain in new information. New paths are searched for using the RRT* (Rapidly exploring random trees) algorithm and the trajectories are appropriately modified.



Initially the feasibility of all the sub-trajectories is calculated to find trajectories that can be represented as a single node of a nested tree data structure. This single node preserves the diversity of the trajectories but makes sure that compartmentalization and local optimization lead to an effective algorithm that can compute relatively optimal paths on-demand. This approach is also called lazy evaluation, thus helping preserve valuable system resources that could be used elsewhere.
Hence, individual sub trajectory nodes can be optimized in a separate thread without affecting the global tree.

Represent subdirectories as the nodes of a tree if and only if $\Phi (S_i \rightarrow S_{i+n}) \geq \varphi_{feasible}$, where $\varphi_{feasible}$ is a threshold.



To join individual nodes to grouped trajectories, we define a function that computes a path from the node to the sub-trajectory which is locally optimal and collision free.
Thus, this function establishes diversity.

Formally the function J: $V \times V = v \rightarrow \eta$

Where $\eta \in \upsilon (S_i \rightarrow S_{i+n})$, to be joined to v iff $v \rightarrow \eta$ is a collision free path.

References

[1] Richard Hartley, Andrew Zisserman : *Multiple View Geometry in Computer Vision.* Cambridge University Press 2003.
[2] Richard Hartley, Andrew Zisserman: *Multiple View Geometry in Computer Vision.* Cambridge University Press 2003. pp. 155–157.
[3] Emanuele Trucco, Alessandro Verri: *Introductory Techniques for 3-D Computer Vision.* Prentice Hall 1998. pp. 132.
[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417,2006.
[5] Beis, J. and Lowe, D.G. 1997. Shape indexing using approximate nearest-neighbour search in highdimensional spaces. In Conference on Computer Vision and Pattern Recognition, Puerto Rico, pp. 1000-1006.
[6] Anne Kaspers: Blob Detection Biomedical Image Sciences, Image Sciences Institute, UMC Utrecht.
[7] Canny, J., *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.
[8] Emanuele Trucco, Alessandro Verri: *Introductory Techniques for 3-D Computer Vision.* Prentice Hall 1998. pp. 132.
[9] Elena Deza & Michel Marie Deza (2009) *Encyclopedia of Distances*, page 94, Springer.
[10] Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, Hong Zhang. Fast Approximate Nearest-Neighbor Search with *k*-Nearest Neighbor Graph. Department of Computing Science, University of Alberta.
[11]Symmetrical model based SLAM [M-SLAM], for a quick map-search. Jung-Suk Oh and Kwee-Bo Sim.
[12]DP-SLAM, Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks, Austin Eliazar and Ronald Parr, Department of Computer Science, Duke University.
[13] RRT*, Planning Algorithms, S.M. LaValle.
[14] Real-Time Adaptive Motion Planning (RAMP), of Mobile Manipulators in Dynamic Environments With Unforeseen Changes. John Vannoy and Jing Xiao, Senior Member, IEEE.