GEOMETRICAL METHODS FOR CONSTRUCTING 2D AND 3D MAPS USING SINGLE PERSPECTIVE IMAGE FOR MOBILE ROBOT PATH PLANNING

A THESIS

submitted by

BHANU CHANDER V.

for the award of the degree

of

MASTER OF SCIENCE

(by Research)

DEPARTMENT OF ENGINEERING DESIGN
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

JANUARY 2018

THESIS CERTIFICATE

This is to certify that the thesis titled **GEOMETRICAL METHODS FOR CONSTRUCTING 2D AND 3D MAPS USING SINGLE PERSPECTIVE IMAGE FOR MOBILE ROBOT PATH PLANNING**, submitted by **Bhanu Chander V.**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Science (by Research)**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. T. Asokan

Research Guide
Professor
Dept. of Engineering Design
IIT-Madras, 600 036

Dr. B. Ravindran

Research Co-Guide
Associate Professor
Dept. of Computer Science
and Engineering.,
IIT-Madras, 600 036

Place: Chennai

Date: 10th January 2018

ACKNOWLEDGEMENTS

I am delighted to thank each and every one of them who helped me in completing my research. It's my pleasure in conveying my immense gratitude to my guide **Prof. T. Asokan** for his constant support and suggestions, right from the very first day he permitted me to work under him. It was because of him I was allowed to take up the challenging work that turned out to be novel. I would like to thank again and again for his constant support throughout my research career, which for sure benefited me. I am grateful to my co-guide **Prof. B. Ravindran** who gave me clear idea of current state of the art and also for motivating me in working on the most challenging part of Robotics. His in-time suggestions and feedbacks helped me explore many new areas. I thank all the panel members for their suggestions and inputs during the seminars. I would like to convey my gratitude to then HOD of the department, **Prof. N.J. Vasa**, for helping me choose the guide. I thank **Dr. C. S. Shankar Ram** for his advises at the time of joining.

I am happy to thank my lab mates **Mr. C. Karthik, Mr. G. Nagamanikandan, Mr. R. Thiagarajan, Mr. N. Ganeshram, Mr. Suraj Parameshwaran, Dr. Sourav Chandra, Mr. Sai Sreekar Annamraju** for giving pointers during lab meetings which helped me think on some of the questions and also perceive few problems in different dimension. A special thanks to **Mr. Senthilkumar D.** for his constant support and in-time help till date. I thank project associates **Mr. Arjun, Mr. K. Mithun** and **Mr. Irfan Habeeeb** for encouraging me in doing the challenging work. I thank **Prof. Sandipan Bandyopadhyay** and his students for keeping me motivated of my works.

I am grateful to my family members and other friends whom believed me and kept me positive till date. Finally, I thank all the people whom are involved directly and/or indirectly throughout my journey as a research scholar.

ABSTRACT

KEYWORDS: Single view metrology, Mathematical imaging, Map Reconstruction, Robot Mapping, Computational Geometry, Navigation, Path planning

In the field of mobile robotics, robot tasks usually involve navigation in a known or unknown environment, for which the robot needs a map of that environment and also needs to know its position within the operating area. Although there are techniques like SLAM (Simultaneous Localization and Mapping) where a map is constructed and/or updated while simultaneously localizing the robot, they do need a map for localizing. So, in any case, having a map at hand is advantageous.

Traditional map construction methods depend extensively on costly sensors. For indoor applications, the robot has to wander around the world numerous times to catch as much data as possible to construct a map, mostly supervised by human. Vision based navigation systems are hence gaining popularity because of being cheap and easy to handle. While map building is easier using multiple vision systems or using multiple images, map construction from single image is definitely a harder problem, although its applications are immense.

In the domain of robotics, the idea of constructing maps from a single image taken from monocular cameras might solve several complex issues related to path planning and navigation. Owing to its diverse applications, the current research focus is on developing methods which would utilize the properties of the images. One such property being considered is *Vanishing points (VPs)*. Existing methods on single camera vision systems depend either on multiple images or on prior information to be fed to the system. The focus of the research presented in this thesis is to develop geometrical methods that will take advantage of VPs for constructing scaled 2D free-space maps and 3D models from a single image, the image being either in two-point perspective (2PP) or three-point perspective (3PP).

Given an image with one or more objects in it, the proposed approach is to bring all the objects to a common ground depth w.r.t. the camera view point. This will ensure that all the objects' projections are to a common scale and thus a scaled map can be constructed. The principal contribution of this thesis is the concept of *side view geometry* of an image, which is first of its kind. All the mathematical derivations have been done using the proposed side view geometry. A systematic method has been

developed, which has been proved as a generalized approach for both 2PP and 3PP cases. A generalized algorithm has been developed, coded, tested with some case studies, and the results are compared with the ground truths. The effectiveness of the proposed method is established through the results and the error plots.

A new methodology for path planning of mobile robots in known environments with static obstacles is also proposed in this thesis. A new concept of multi-bug system will be introduced. Multi Bug Path Planning (MBPP) works by assuming a virtual bug that move towards the goal from the start state. If in case it meets an obstacle, then that bug generates a new bug. The two bugs now move along walls of the obstacle in either direction until specific conditions are met. Bug generation continues whenever any of the current bugs hit a new obstacle, until target is reached by all live bugs. MBPP thus evaluates best possible paths for a given environment and chooses the best route that is supposed to be optimal. Experimentally, it will be shown that MBPP finds paths that are shorter and comparable with post-smoothed A* in lesser runtimes. Proposed work on MBPP is an attempt to combine the features of offline and online methods so that the same algorithm could be used for both cases. Current work on MBPP demonstrates the offline case with static obstacles in a priori known environment.

TABLE OF CONTENTS

ABSTRACT

LIST OF FIGURES

ABBREVIATIONS

NOTATIONS

CHAPTER 1 INTRODUCTION

1.1 Literature Survey 4

1.2 Motivation and Research Gap 9

1.3 Objectives of the Research Work 9

1.4 Overview of the Research 10

1.5 Potential Contribution of this Thesis 11

1.6 Organization of the Thesis 11

CHAPTER 2 BASICS OF MAP BUILDING METHODS

2.1 Vanishing Points 13

2.2 Terminology 16

2.3 Cordinate Systems 17

2.4 Side View of an Image 18

2.5 Construction of Top View 19

CHAPTER 3 MAP CONSTRUCTION FROM A SINGLE TWO-POINT PERSPECTIVE (2PP) IMAGE

3.1 Overview of Methodology 22

3.2 Assumptions 23

3.3 Proposed Methodology 23

3.3.1 Finding Relative Position of Objects 23

3.3.2 Recognizing Free Space Using the Floor Corners 26

3.4 Finding Height of Objects for Building 3D Maps 29

CHAPTER 4 MAP CONSTRUCTION FROM A SINGLE THREE-POINT PERSPECTIVE (3PP) IMAGE

4.1 Overview of Proposed Methodology for 3PP Single Image Map Construction 30

4.2 Preliminery Concepts of 3PP 31

4.3 Derivation of Mathematical Relations for Estimaing Geometric Parameters 34

4.4 Side View Geometry of Several Objects in a Given 3PP Scene 37

4.5 Finding Offset Distances of Multiple Objects in the Image 40

4.6 Relative Positioning of Multiple Objects 42

4.7 Mathemtical Relations for Free Space 44

4.8 Finding Height of Objects for Building 3D Maps 45
4.9 THEOREM 1: 2PP is a Special Case of 3PP, When Pitch Angle is Zero 49
CHAPTER 5 MULTI-BUG PATH PLANNING (MBPP) ALGORITHM
5.1 Introduction 52
5.2 Bug Algorithms and Related Works 54
5.3 Notations 55
5.4 Multi-Bug Path Planning (MBPP) Algorithm 55
5.5 Flow Chart 59
5.6 MBPP Conditions 60
5.7 Properties of MBPP 62
5.8 MBPP Pseudocode 63
CHAPTER 6 EXPERIMENTAL RESULTS
6.1 Map Building Experimental Results 64
6.1.1 Error Metrics 66
6.1.2 Results for 2PP Images 67
6.1.3 Results for 3PP Images 74
6.2 MBPP Simulation Results 79
CHAPTER 7 SUMMARY AND CONCLUSIONS
7.1 Summary 82
7.2 Conclusions 83
7.3 Future Scope 84
REFERENCES 85
APPENDICES
A PSEUDOCODES
A.1 Agorithm 1: Generalized Algorithm for Single Image Metrology of 2PP and 3PP
A.1(a) Main Algorithm 90
A.1(b) Function: *Box_Topview()* 92
A.2 Agorithm 2: Multi-Bug Path Planning Algorithm
A.2(a) Main Algorithm 93
A.2(b) Function: *Line_Of_Sight* Algorithm 94
A.2(c) Function: *Path_Smooth* Algorithm 94
B PROCEDURE FOR 2PP TOP VIEW CONSTRUCTION 94
PUBLICATIONS 98

LIST OF FIGURES

1.1	Results on Free Space Recovery from Single Images.……..…….	7
2.1	One-Point Perspective (1PP) Image…….…….	14
2.2	Two-Point Perspective (2PP) Image…….	15
2.3	Three-Point Perspective (3PP) Image.……..	15
2.4	Standard Camera Model showing the notion of top and side views.…….	16
2.5	Camera System DoF… …….	17
2.6	Concept of Side View Geometry for an Example Image.…….	19
2.7	Line Diagram of an Example 2PP Image……..…….....	20
2.8	Top View of the Objects Obtained Using Traditional Method…….	20
2.9	Actual Position of the Objects in Top View…..…….	21
3.1	Flow Chart depicting the Proposed Methodology for 2PP Case……	22
3.2	Line Diagram of an Example 2PP Image…… …….	24
3.3	Side View Geometry for Objects in 2PP……..	24
3.4	2PP Objects Top View……..……	26
3.5	Side View Geometry for Floor Corners in 2PP…….……	27
3.6	Top View of the Floor…….…..…	28
3.7	Complete Free Space Map for the 2PP Example Image in Fig.3.2…..….	28
4.1	Flow Chart Depicting the Proposed Methodology for 3PP Case……..…	30
4.2	An Example Image in 3PP with its Line Diagram…..……..	31
4.3	Finding Image Center and Vanishing Points…….	32
4.4	Geometrical Construction for Top View of Single 3PP Object.……	33
4.5	Imaginative Side View…….…….	34
4.6	Side View Depicting an Object and Camera View Point in 3PP...……	35
4.7	Side View with Construction Details……..……..	35
4.8	Top View Construction……..……	37
4.9	Actual Side View of Two Objects…….	38

4.10 Deduced Configuration……..…….... 39

4.11 Side View with the Transformed Image Plane…….……. 39

4.12 Geometry for Offset Distance Measurement……..…. 40

4.13 Geometry for Case 3……...…. 42

4.14 Top View Construction of Objects……..… 43

4.15 Geometries for Floor Space Estimation……..… 44

4.16 A Complete 2D Map for the example scene shown in Fig. 4.2……. 45

4.17 Notion of Transformed Image Plane for Height Measurement……. 46

4.18 Geometry for Reference Object Height Measurement…….. 46

4.19 Geometry for Measuring the Projected Length l_{new}……. 47

4.20 Geometry for Finding l_{new} in Case 2..…….. 48

4.21 Geometry for Finding l_{new} in Case 3……..…….. 49

4.22 Constructed 3D Model….…….. 49

5.1 A* Algorithm Post Smoothening…… 53

5.2 MBPP Implementation…….. 58

5.3 Paths Found by MBPP…… 58

5.4 Another example for MBPP Implementation…… 59

5.5 An Example Environment with Two Obstacles…… 59

5.6 Flow Chart Depicting the Structure of MBPP…….….. 60

6.1 An Example Image in 2PP…….…… 64

6.2 Notation Used for Finding Error..…….. 67

6.3 2PP Experimental Result – 1……. ….. 68

6.4 2PP Experimental Result – 2……..…. 68

6.5 2PP Experimental Result – 3……..…. 69

6.6 2PP Experimental Result – 4……..…. 69

6.7 2PP Experimental Result – 5……..…. 70

6.8 Box Plot of Percentage Deviations of Centroids…… 70

6.9 Box Plot of Percentage Reductions in Distances to Centroids……..…. 71
6.10 Box Plot of Percentage Area of Overlap…… 71
6.11 Box Plot of Percentage Error in Area Measurement……..…. 72
6.12 Box Plot of Percentage Error in Angular Pose of the Centroids…….….. 72
6.13 Box Plot of Percentage Error in Skew Angles…… 73
6.14 3PP Experimental Result – 1……..…. 74
6.15 3PP Experimental Result – 2……..…. 75
6.16 3PP Experimental Result – 3……..…. 75
6.17 3PP Experimental Result – 4……..…. 76
6.18 Box Plot of Percentage Deviations of Centroids…… 76
6.19 Box Plot of Percentage Reductions in Distances to Centroids……..…. 77
6.20 Box Plot of Percentage Area of Overlap…….….. 77
6.21 Box Plot of Percentage Error in Area Measurement……..…. 78
6.22 Box Plot of Percentage Error in Angular Pose of the Centroids…….….. 78
6.23 Box Plot of Percentage Error in Skew Angles…… 79
6.24 MBPP Implementation on a 3PP Map…...…… 80
6.25 MBPP vs. A* (runtime)…… 80
6.26 MBPP vs. A* (path Length)…… … 81
B.1 Procedure for Station Point Determination……..…. 95
B.2 2PP Object Top View Construction Procedure……..…. 97

ABBREVIATIONS

1PP	One Point Perspective
2D	Two dimension
2PP	Two Point Perspective
3D	Three dimension
3PP	Three Point Perspective
A*	A-Star
DFD	Depth From Defocus
LOS	Line of Sight
MBPP	Multi Bug Path Planning
PS	Post smoothening
SFS	Shape From Shading
SLAM	Simultaneous Localization And Mapping
VP	Vanishing Point

NOTATIONS

Greek Symbols

β	Angle made by the nearest base corner of object along the viewing direction in the side view
β_a	Orientation of the actual object's side (skewness) w.r.t horizontal
β_m	Orientation of the modelled object's side (skewness) w.r.t horizontal
θ	'Image Plane Angle' – Pitch angle of the camera w.r.t. the ground/horizontal floor
θ_a	Orientation of the centroid of the actual object (angular pose) w.r.t horizontal
θ_m	Orientation of the centroid of the modelled object (angular pose) w.r.t horizontal
θ_{roll}	Camera Roll Angle
θ_{yaw}	Camera Yaw Angle

English Letters

A_a	Actual area of the object found using ground truths, scaled by S_k
A_{cm}	Area of the coincident zone between the modelled object and the actual object positioned w.r.t. view point
A_m	Area of the object in map
Bi	i^{th} Object
b_i	Vertical distance of object *i* from station point in side view
C_i	i^{th} corner of floor
C_i^1	Projected i^{th} corner of floor
d, f	Focal length of the camera; distance between the camera centre and the image centre
d_a	Euclidian distance of centroid of the actual object from the station point
d_{cm}	Euclidian distance between the centroids of the actual object (C_a) and the modelled object (C_m)
d_f	Offset distance by which a floor corner has to be shifted to bring it to a common reference
d_m	Euclidian distance of centroid of the modelled object from station point
d_p	Offset distance between the original and the transformed image planes in the side view for object corners

h_d	Vertical distance between two objects base corners
l	Height of reference object in side view of 3PP case
L_i, B_i	Dimensions (length and breadth) of i^{th} object
l_{new}	Projected height of non-reference object in side view of 3PP case
O	Image plane centre
O'	Image centre projection onto the vanishing circle
S_k	'Scale factor' – A Common scaling factor for the whole map
S_S, S_G	Start and Goal states
u, v	Image plane coordinate axes
VP_i	i^{th} vanishing point
X_C, Y_C, Z_C	Camera frame coordinate system
X_W, Y_W, Z_W	World frame coordinate system

CHAPTER 1

INTRODUCTION

In the field of mobile robotics, navigation is an important task for any robot. Robots are meant to perform desired tasks like fetching drinks, shifting objects, guiding people, cleaning rooms etc. In most of the applications concerned, navigation (and hence path planning) plays a vital role in the performance of a robot. For the robot to plan a path, it either needs a pre-loaded map of the environment or needs to build the map on-the-go. In either case, a prior knowledge of the environment in the form of a map is a must, so that the robot can deal with the basic questions like 'where am I ', and 'where am I going'. Robotic mapping addresses the problem of acquiring spatial models of physical environments through sensors mounted on the mobile robots. The mapping problem is generally regarded as one of the most important problems in the pursuit of building truly autonomous mobile robots (Thrun, 2002). Despite significant progress in this area, it still poses great challenges. Several research works exist for depth estimation, image reconstruction as well as for 3D map building. Most of them fall under one of the categories mentioned below:

- Maps built using sensors like LIDAR (Light Detection and Ranging), ultrasonic or range sensors, and sometimes in combination with vision systems (termed as Robotic Mapping) (Thrun, 2002)
- Maps built using depth measuring sensors or RGB-D sensors like Microsoft Kinect (Zhu *et al.*, 2016; Endres *et al.*, 2013)
- Maps constructed using techniques like depth-from-defocus (Zhuo and Sim, 2011)
- Using stereo vision systems (Saez and Escolano, 2004)
- Monocular (single camera) vision system using techniques like depth from motion (taking sequence of images and finding the depth) or using machine learning algorithms; the later part is expensive, in terms of human involvement (Einhorn *et al.*, 2009).

In the traditional way of building maps for robot navigation costly sensors are used and the process is time consuming and laborious. Typically, robots are allowed to wander around to gather cloud points. This will be processed to form images and they are

stitched to build maps. This is apparently a tedious process. Apart from that, most of the times the process demands multiple sensors, especially when constructing outdoor maps. It has to be noted that the usual mapping process through *SLAM* (Simultaneous Localization and Mapping) needs a high end workstation, needing large size memory, graphics and compatibility platforms, and OS for even running the software to process the huge data acquired through the sensors. Considering these factors, any map building process that reduces the dependency on multiple, costly process and also reduces the involvement of humans is considered to be futuristic. So, the research focus is moving towards map construction process using cheaper, alternative solutions. Using vision systems for map building is one of the focus areas due to the fact that they are cheap, readily available and easy to handle.

Cameras are generally used for capturing 2D images of 3D environments. Besides the problem of losing information when 3D worlds are captured in 2D images, problems exist in finding the relative positions of objects as well as in estimating the camera locations. Finding the position of a camera and its properties from images/photographs is called *Camera Calibration* (Zhang, 2000). The process of calibration is quite tedious and involving. Tremendous work has already been carried out in this area. Computer vision researchers are facing a big challenge of depth reconstruction from camera images without the need of calibration process. To the best of our knowledge, very few works have been observed which could address the problem of estimating the properties/measurements from images by eliminating completely the calibration step (images taken either from a single camera or several cameras). The main issue is the difficulties involved in getting a prior knowledge of the world being captured in the image. This requires human interventions and lot of external processing.

Most often, stereo cameras (Scharstein and Szeliski, 2002) are used which incorporates two cameras separated by a known gap and uses triangulation principle (Hartley and Sturm, 1997) for depth information. Finding depths using calibrated stereo cameras are commonly employed for depth-map generation (Kamencay, 2012). Depth-map of an image or scene gives information relating to the distance of the surfaces of scene objects from the viewpoint. In other words, if the camera from which the image was taken was assumed to be the view point, then depth-map just gives the Euclidean distance of the scene points from the camera centre. However, depth maps will not provide information

of free and occupied space as well as the relative positions of the objects on the floor, which is essential for a robot to plan a path in order to navigate in that environment.

Monocular (single camera) vision systems do exist. However, much of the reported techniques (as observed in the literature of single camera vision) focus on one of the following methods:

- Methods based on multiple images (sequence of images) (Einhorn *et al.*, 2009)
- Methods relying on human interaction for inputting necessary data, performed every time when the algorithm is put to work (Horry *et al.*, 1997; Gozali, 2006)
- Methods that needs a one-time processing of huge training data set (priori) by considering several images along with their ground truth, for assisting a learning algorithm (Saxena *et al.*, 2008)

It is clear that the existing methods on single camera vision systems depend either on multiple images or on prior information to be fed to the system. Considering the advantages and diverse applications of using single images for depth/map reconstruction, and with the motto of eliminating laborious human involvement to make the process completely automatic, researchers have been working on single image systems without camera calibration. 'Single View Metrology' are methods that evaluates measurements from single perspective images using minimal geometric information (Criminisi *et al.*, 2000). The problem is tough since information is lost and any point in the image plane would represent infinite number of possible positions in the real world. Using geometrical properties of the image might be a way forward in finding a solution to this problem.

Vanishing point is one such property that is useful in finding the depth and camera position in an image. Any object/scene as seen from human eyes or through camera appears in perspective. Perspectiveness can generally be visualized by the distorted shape of the object. Several methods have been developed for finding the position of the vanishing points (KoSecki and Zhang, 2002; Rother, 2002; Saini *et al.*, 2013) as well as for camera calibrations using vanishing points (Caprile and Torre, 1990; Cipolla *et al.*, 1999; He and Li, 2007). In (Li, 2010), a method for simultaneous vanishing point detection and camera calibration has been introduced. Although the author claims about the ability of the method for camera calibration, it could only measure the focal length,

apart from VP detection. Also, examples shown as experimental images considered only a single object in the image and depth estimation or map building is not considered. Some of the recent and relevant works on single image metrology and map building is discussed in the following section.

1.1 LITERATURE SURVEY

Understanding scene from a single 2D image is a subject matter of interest for researchers, given its applications in wide domains, especially for robot navigation tasks. Robust methods are yet to be developed for constructing 3D maps from single images taken from monocular cameras. In the literature related to single image analysis, few works are observed which dealt with the estimation of spatial layout of indoor scenes. Most of them are based on the common assumption of Manhattan worlds (Coughlan and Yuille, 1999). In an attempt to recover pose and to extract rectangular surfaces of a scene aligned along the three major directions, Kosecka (Kosecka and Zhang, 2005) introduced a method that utilized their previous work on estimating image properties (Jang and Rossignac, 2001) viz. vanishing points and also camera parameters. However, this work gave knowledge about the salient directions in a structured scene and didn't deal with spatial layout estimation.

Pero *et al.* (2011) proposed a generative modelling framework using a top-down Bayesian approach, for understanding indoor scenes to distinguish between scene types such as bedrooms, living rooms etc. Objects were modelled as simple blocks, typically a good approximation or a bounding box for 3D world or object characteristics. Although their work exhibited estimations of room models it couldn't clearly distinguish different objects in the scene. The notion of 2D/3D map of the world specifying the actual free space available in the scene is missing in this work.

The best known attempt on Single View Metrology can be found in (Criminisi *et al.*, 2000) where attempts to get 3D affine measurements from single view of a scene in perspective, making use of minimal information (vanishing line and a vanishing point) a priori, is presented. They applied their method for several architectural images and also showed 3D models of renaissance paintings (Liebowitz *et al.,* 1999). Their work was successful in reconstructing 3D models of the scenes. But the reconstructed 3D models are not helpful in providing floor plans, which specifies the free space available in the given scene. When the technique is to be utilized for robotic navigation tasks, the

robot needs to get information about boundary/size of the objects (and hence the free space), without which the robot cannot plan a feasible path in the environment. In fact, manual labelling is needed for this work (Hoiem *et al.*, 2007).

Similar works are noted in (Hoiem *et al.*, 2005(a), 2005(b)), where authors proposed a technique to estimate scene structure from single image by learning appearance-based models of geometric classes. Each pixel will be assigned a label belonging to any one of the three major classes: ground plane, surfaces at right angles to the ground plane and sky. They focused extensively on outdoor images, added with an important assumption that the camera axis is parallel to the ground plane. This in itself is a serious challenge to indoor environments, especially when the images are taken from surveillance cameras from different tilt positions. In fact, the paper has not given any clear indication of depth maps, but rather gave few results showing novel views of scaled 3D model.

One of the earliest works on modelling the 3D interaction between objects and the spatial layout are observed in (Hedau *et al.*, 2009), where the problem of recovering spatial layouts of indoor cluttered scenes from single images is addressed by modelling the scenes jointly in terms of 3D box layouts and surface labels of pixels. A more closely related approach is observed in (Hedau *et al.*, 2010; Lee *et al.*, 2010). The authors in (Lee *et al.*, 2010) considered parametric representation of objects in 3D to fit box shaped cuboids in the world, by giving volumetric reasoning to the layout. Their work relied on three volumetric constraints of the physical world and performed well on extracting spatial layout of the room and the configuration of objects in the scene. However, their goal was not on free space estimations in the scene. In addition, for predicting the surface geometry of the regions in the image, labelling is to be done for each super pixel which included solid and porous regions.

Recent efforts on extracting spatial layout using classifiers to fit a parametric model of the room was presented in (Pero *et al.*, 2012; Hoiem *et al.*, 2008). This method was successful in predicting surface orientation labels such as floor, ceilings, left, right, centre wall etc. But according to (Lee *et al.*, 2010), labels are limited in the sense that they under-utilize the extractable cues from the configuration of the objects in the room. A dynamic Bayesian network model for recovering 3D reconstruction was done in (Delage *et al.*, 2005(a), 2006(b)), where an approach using Markov random field (MRF) was introduced. These 3D reconstruction works are suited for uncluttered worlds and

the notion of free space finding is missing in their works. In (Saxena *et al.*, 2008), 3D depth maps were constructed from single images using supervised learning technique (a hierarchical multi-scale Markov Random Field model). Their approach needed collection of large set of training data in the form of images (unstructured indoor and outdoor environments covering trees, forests, buildings etc.). But, as stated in (Hedau *et al.*, 2009), depth maps are successful in providing only local information about visible surfaces. For applications like robot navigation, depth maps might not be helpful. Few other approaches on spatial layout estimations from single images are found in (Han and Zhu, 2003; Yu *et al.*, 2008; Lee *et al.*, 2009; Gupta *et al.*, 2011).

Out of the available research works on spatial layout estimation, very few of them have actually worked on the notion of free space maps (Hedau *et al.*, 2012; Herbert *et al.*, 2012). A free space map is a 2D or 3D map illustrating the unoccupied space in the given scene, with near-to-accurate measure of the relative distances between the objects in the cluttered scene. Existing works on floor map buildings modelled objects which are axis-aligned with the identified walls and the floor. Objects' boundaries in the constructed floor maps are made parallel to the walls/scenes even if they are not parallel in the real world.

Figure 1.1 shows the sample results taken directly from (Hedau *et al.*, 2012), which clearly shows how objects are aligned to the side walls. In addition, they had to create a dataset of 592 indoor images taken from several sources to train their model. Some of the floor map results available in (Herbert *et al.*, 2012) didn't even capture accurate free space in the scene, which is quite different from our goal of accurately positioning the cuboid and non-cuboid shaped objects relative to one another to help in robotic path planning.

In (Horry *et al.*, 1997), a new method termed TIP - 'Tour into the picture' was introduced that can create animations from single image/photograph. TIP method uses prior information (manual input) in the form of foreground object and the method just created views (a model) without considering the correct 3D structure of the objects. A similar work was observed in (Gozali, 2006), where 3D modelling of single images were done semi-automatically, i.e. with the help of user interaction with the system. The method requires that the user supply reference lines and points typically related to perspective of the image. The method is applicable only for 2PP structured images and is not applicable for 3PP case.

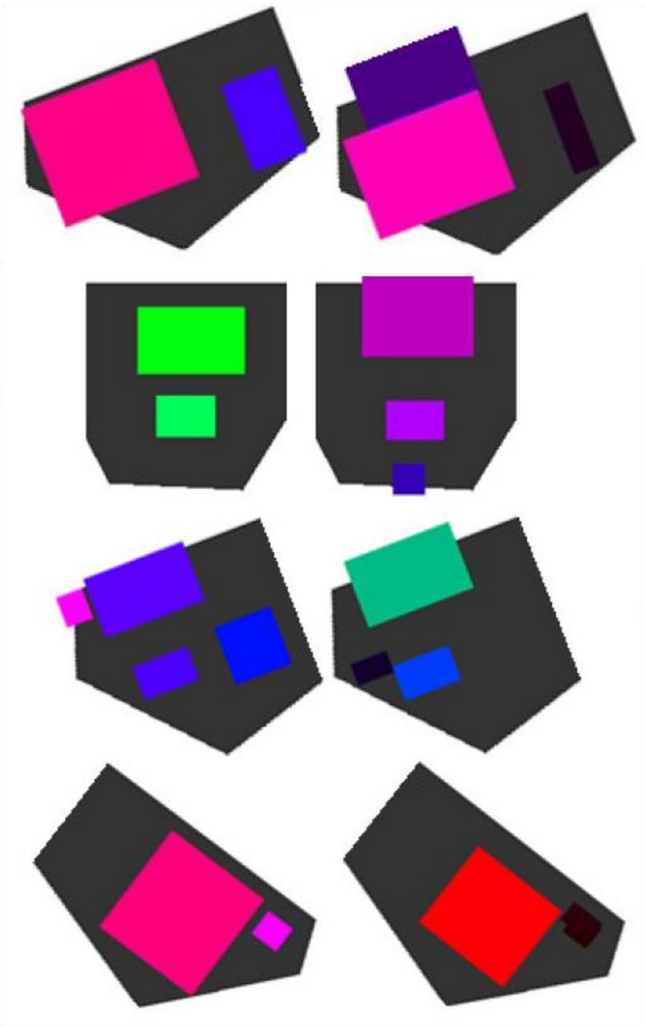


Fig. 1.1: Results on free space recovery from single images (Hedau *et al.*, 2012).

In (Hoiem *et al.*, 2005), an attempt to estimate scene structure from single image has been made by learning appearance-based models of geometric classes. Each pixel has been assigned a label belonging to any one of three major classes: ground plane, surfaces at right angles to the ground plane, and sky. Surfaces sticking up from the ground are further subdivided. One more important assumption considered is that the camera axis is parallel to the ground plane. This in itself is a serious challenge to indoor environments, especially when the images are taken from surveillance cameras. In fact, the paper has not given any clear indication of depth maps but rather gave few results showing novel views of scaled 3D model.

In (Byres and Henle, 2004), the problem of understanding the relative positions of objects and images was discussed along with mathematical relations for architectural scenes, which can also be used for related applications. They considered objects as rectangular solids and their method works only if one side of the object/structure in the scene is measured physically at the site (location of capture). Similar work was done in

(Crannell, 2006). Both the works are seemingly applicable only when the object of interest in the image is in 2PP. A few other works on making measurements using single images can be found in (Proesmans *et al.*, 1998; Kim *et al.*, 1998).

Techniques such as shape-from-shading (SFS) (Horn, 1970) recovers shape from the shading information in an image. SFS methods needs the principle of homogeneity, Lambertian reflectance (Durou, 2008) and works well when surface is lit with single distant light source assuming that the source direction is known. However, for images with many light sources SFS needs assumptions about the surface shape (Tsai *et al.*, 1999). Depth from Defocus (DFD) is another technique for depth measurement using single camera (Chaudhuri and Rajagopalan, 1999) but uses a minimum of two images taken with different focal lengths, which involves modification of camera property every time when images are taken. RGB-D Camera systems are yet another sensing tools for estimating depths (Henry *et al.*, 2014). But they usually depend either on stereo or time-of-flight sensing.

Map building using visual images has been in the research focus for many years. However, map building using single camera image is relatively new topic. In (Davison *et al.*, 2007), an approach dubbed *MonoSLAM* was presented for online building of probabilistic maps using single uncontrolled camera, without the need for any other sensor assistance. But, MonoSLAM gets data sequentially while in transit (during motion) and keeps updating the map. This is the situation of depth from motion. In fact, certain amount of prior information about the scene has been considered as a start-up aid to the system (that they developed). This aid is in the form of a known target (a black rectangle with measured corner positions) placed in front of the camera. So, MonoSLAM is basically not useful when an image taken long back or taken from unknown cameras is provided for analysis.

In the domain of robotics, the idea of constructing maps that can accurately distinguish between the occupied and available space in a scene from a single image might solve several complex issues related to path planning and navigation. This concept has the ability to avoid the dependency of the robotic systems on costly sensors for usual mapping process, which is labour and time intense, apart from cost concern. Auto calibration and map construction from single image has the extended advantages of being useful for state-of-the-art and future technologies like driverless cars, Virtual and Augmented Reality, to name a few. So, the primary concern of the work presented in

this thesis is to develop robust methods for 2D and 3D map building from single images taken from monocular cameras, without the need of camera calibration.

1.2 MOTIVATION AND RESEARCH GAP

The literature survey related to map building has resulted in the following observations:

1 Methods for spatial layout estimations exist but are not applicable for map building and/or robot navigation tasks since they don't address the problem of estimating free space from single images.
2 Most of the existing works addressed the problem of spatial layout estimation for structured rooms, especially of the categories like living room, bed room, and dining room.
3 Existing floor space estimation methods are not complete and consider only objects that are axis–aligned (parallel) to the walls. They don't capture the exact free space in the environment.
4 Depth maps might not be useful for robot navigation applications.
5 Traditional/existing robotic mapping involves costly sensors and the process is cumbersome.

Owing to the concerns listed above, this thesis aims at proposing methods for building near-to-accurate maps using simpler approaches and using minimal resources, without the need of costly process.

1.3 OBJECTIVES OF THE RESEARCH WORK

The overall objectives of this research work are

1. To construct free space maps (2D and 3D) from single images taken using monocular cameras, without the need of calibration and/or pre-learning.
2. To introduce a new path planning algorithm that will take the constructed maps as input, and thereby plans navigable and optimal paths.

The tasks involved in achieving these objectives are:

1. Generation of side view geometries for both 2PP and 3PP configurations.
2. Development of a new generalized methodology for building 2D as well as 3D maps, using simple geometrical information from single monocular camera images.

3. Design of a new algorithm for planning optimal paths in a static environment, utilizing the constructed maps from the proposed map building method.

Two new methods are developed for single image analysis of 2PP and 3PP images. Both approaches take the advantage of geometry of vanishing points and are based on general artistic way of constructing top view of a given perspective object (Ching and Juroszek, 2010). Results show the robustness of the methods introduced. Proposed methods are primarily built to construct a 2D/3D map of the world of interest as seen from the image, such that, if any one object's dimension (say robot dimension) is known, then, we can build the map to the same scale as that of the real world. If prior information is not available, then the methods are able to construct maps that are scaled to a scaling factor. We assume that vanishing points and spatial layouts of the given scene are available and thus proceed on solving the latter part of map building.

However, below are some challenges for the proposed methods.

1. Correct estimation of the vanishing points.
2. Fitting/inscribing a bounding box for identified world object.
3. Dealing with occluded surfaces.
4. Analysis for objects having non orthogonal surfaces, i.e. when *all* the objects in the scene have different shapes other than rectangular cuboid.
5. When objects are far, then their exact edges are tough to identify.

1.4 OVERVIEW OF THE RESEARCH

The foremost work on single image map construction process starts with the concept of side view geometries, which happens to be the core contribution of this thesis. Using the basic notion of the side views, several geometries were introduced, separately, for both 2PP and 3PP cases. Considering one case study for each of the two cases, derivations of the relations had been done in step by step and structured manner. These relations were then employed for the example case study images for finding the objects' scaled dimensions and their relative positions w.r.t. each other, and the position of the camera station point for that specific scene. The geometries were extended further to identify the navigable free space of the given scene. Later, derivations for height measurements were done, using which 3D maps were constructed for the case studies

considered. Map construction results were shown by comparing with their corresponding ground truths for all the considered example single images and discussions on the error plots were carried.

Finally, the map results of the set of 2PP and 3PP single images were considered to test on a new path planning algorithm – multi-bug path planning algorithm (MBPP), and the optimal performance were proved by comparing the simulation results with that of standard A* post smoothening algorithm.

1.5 POTENTIAL CONTRIBUTION OF THIS THESIS

1. To the best of our knowledge, no work has been observed in the literature related to accurate free space floor map construction from single image, except depth estimations and spatial layout. Proposed work is first of its kind which eases the process of map building with just a single image.
2. This thesis will introduce the notion of estimating geometric parameters from side view geometry of a given single image which is not available in the literature till date.
3. The method doesn't need the dependency on preliminary camera calibration process, assuming that the skew and the lens distortion are within acceptable range. In fact, estimation of the focal length and the pose of the camera in the scene are made an integral of the map building process, using just geometries, needing no physical measurements in the world. Therefore, the advantages are many folds.
4. Proposed geometrical approach is generalized and is very simple, easy to understand and implement. The method doesn’t involve any learning process for map construction and hence very efficient in terms of time as well as accuracy.
5. Proposed method eliminate the need of relying on costly sensors for map building. A single image is sufficient and no need for multiple images, taken in sequence, or techniques like depth from defocus.
6. A new Planning and Re-planning algorithm that can help in finding multiple feasible paths, apart from optimal paths.

1.6 ORGANIZATION OF THE THESIS

Chapter 2 introduces the basics needed to understand the map building process. A detailed subsection regarding the introduction and types of vanishing points, the

common terminology employed for mapping geometries and the standard coordinate system considered have been dealt at length. The concept of side view geometry is introduced here which happened to be the core idea using which all the geometries (and hence the mathematic relations) will be derived.

Chapter 3 is dedicated to the discussions on map reconstruction using a single two-point perspective (2PP) image. The chapter begins with the brief introduction on 2PP, followed by the common assumptions made before going to the method. Then, a detailed discussion on the proposed methodology (applied to a case study) will be done with sufficient images, for deriving the mathematical relations that will find the offset distances. Later, relations needed for recognizing free space using the floor corners will be introduced, applying the same to the example image considered.

Chapter 4 begins with introduction to the three-point perspective (3PP) case. Considering only one single object, the mathematical relations needed to evaluate the parameters will be discussed at length with relevant side view geometries. The concept will be extended to the scenario of multiple images and the core mathematical relations for finding the offset distances for all the objects in the given 3PP scene will be derived. This chapter also shows the possibility of 3D map constructions (a model of the given scene) by deriving the relations for height calculations too.

In Chapter 5, a new path planning algorithm called "Multi-Bug Path Planning" (abbreviated hereafter as MBPP) is introduced. We discuss the algorithm at length with the help of an example, followed by a flow chart for clear understanding. Later, we also discuss in detail the various conditions and properties of proposed MBPP algorithm.

In chapter 6, the experimental results of map construction for both the cases of 2PP as well as 3PP will be presented. Experimental results of MBPP algorithm applied on the maps deduced from the proposed map constructions process also will be presented.

Chapter 7 gives a summary of the work and contributions of this thesis and concludes the findings with a brief discussion on the future scope of work.

CHAPTER 2

BASICS OF MAP BUILDING METHODS

This chapter introduces the basic concepts essential for understanding the process of Map building. The concepts introduced include details of Vanishing Points (VPs), the common terminology followed throughout this thesis, and the notion of standard coordinate system. This chapter also discusses about the very important concept of *side view of an image* which has been proposed in this thesis. Using the side view geometry all the mathematical relations needed for constructing a map would be derived. This chapter concludes with a brief discussion on constructing a top view of the objects in a given image.

2.1 VANISHING POINTS

Any object/scene as seen through human eyes or through cameras appear in perspective. When 3D objects are represented in 2D planes, taking into consideration the right details of their dimensions including their positions (relative to each other), then, we can say that the objects are in *perspective*. Perspective is an important concept in several fields that are related to cameras, images etc. Perspectiveness can generally be visualized by the distorted shape of the objects. Based on the property called *Vanishing Point*, one can tell the type of perspective (Jang and Rossignac, 2001). When an object is placed under perspective, a vanishing point is a point at which a set of parallel lines of that object appear to converge. Mathematically, parallel lines meet at infinity and so the vanishing points are supposed to lie at infinity. When seen through image plane, these vanishing points assume to be lying on a straight horizontal line called '*horizon line*', which happens to be the eye level of the camera. Perspective projection differs from orthographic projection in a way that there will be no notion of vanishing points in orthographic projection. In orthographic projections, all the parallel lines will still be parallel in the view unlike in the perspective projection in which parallel lines will appear to converge at vanishing points.

In 2D/3D, based on the angular pose of the camera w.r.t. the object or world coordinate system, three types of perspective is possible viz. '*One-Point Perspective'*, '*Two-Point Perspective'* and '*Three-Point Perspective*' (hereafter termed as 1PP, 2PP and 3PP respectively) (Mauldin, 1985). Figure 2.1 shows the situation of 1PP. When the camera

(and hence its image plane) is parallel to one of the surface of the object with zero angular pose (shown in Fig. 2.1(a)), then the perspective projection obtained will be considered as 1PP. The situation is shown in Fig. 2.1(d). In this case there will be only one vanishing point.

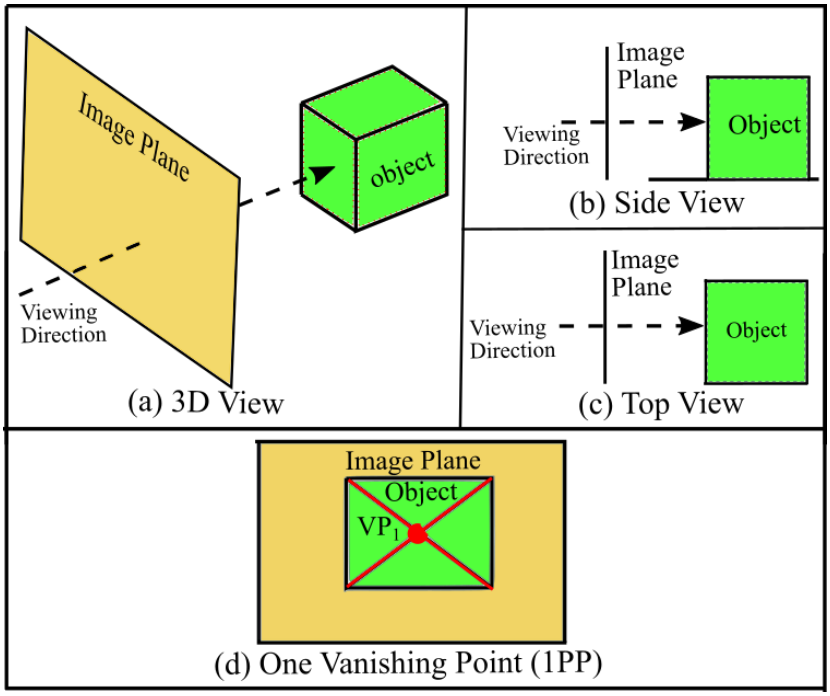


Fig. 2.1: One point perspective (1PP) image. (a) 3D view (b) Right-side view (c) Top view (d) Front view showing one vanishing point

Figure 2.2 shows the situation of 2PP. When one of the edges of the object is parallel to the image plane (as visible in Fig. 2.2(b)), then the perspective projection obtained will be considered as 2PP as shown in Fig. 2.2(d). In 2PP case, object's surface is not parallel to the image plane but only one of the edges is parallel. As shown in Fig. 2.2(c), if we try extending the edges of the object (shown in red colour), they meet at two vanishing points marked as VP_1 and VP_2. Note that these two points lie along the same horizontal line termed as '*horizon line*', which represents the eye level of the viewer or the camera. In 2PP case, camera (hence its image plane) has still zero angular pose.

Figure 2.3 shows the situation of 3PP. In this case, the image plane has some pitch and makes some pitch angle with the world coordinate system. Also, neither any surface of the object nor its edges are parallel to the image plane. As shown in Fig. 2.3(c), the object looks distorted with its side not being parallel. If we try extending the edges of the object (shown in red colour), they meet at three vanishing points marked as VP_1, VP_2 and VP_3. Note that still two vanishing points lie along the horizontal line which

still represents the '*horizon line*', in other words, the eye level of the viewer or the camera.

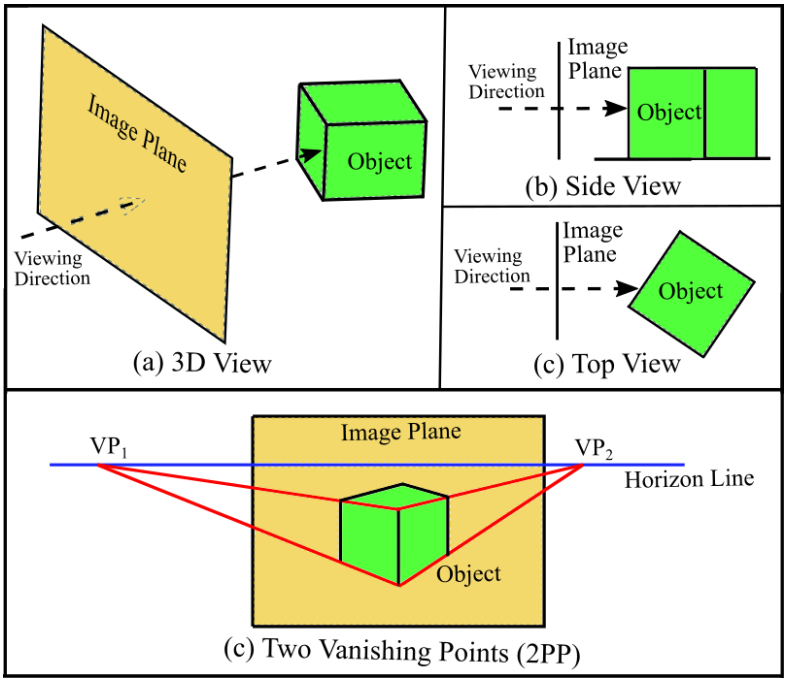


Fig. 2.2: Two point perspective (2PP) image. (a) 3D view (b) Right-side view (c) Top view (d) Front view showing two vanishing points

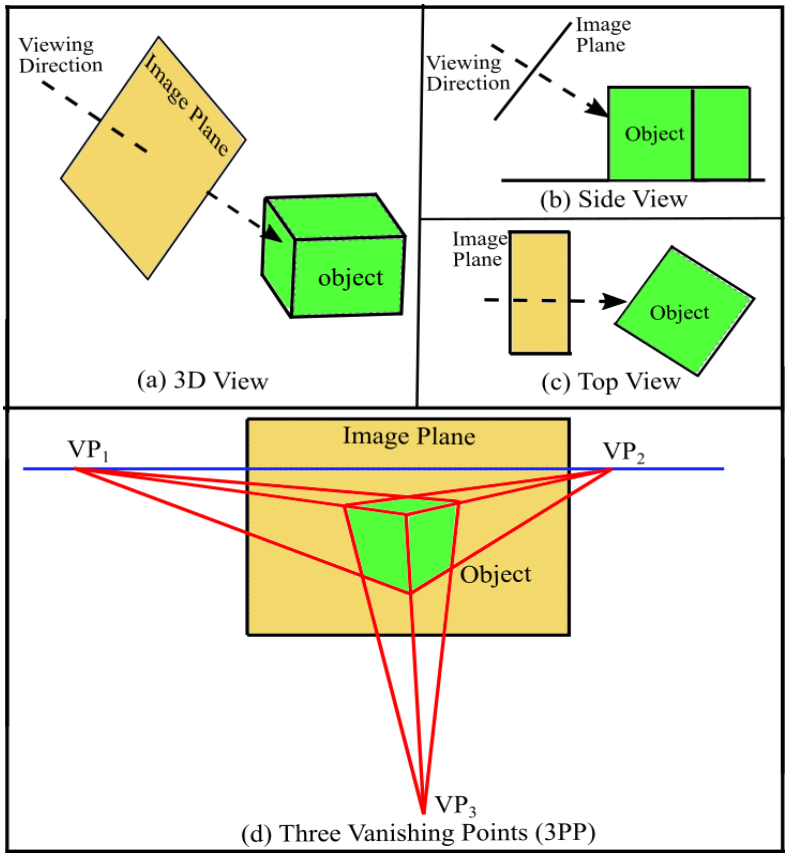


Fig. 2.3: Three point perspective (3PP) image. (a) 3D view (b) Right-side view (c) Top view (d) Front view showing three vanishing points

2.2 TERMINOLOGY

With reference to Fig. 2.4, the following list provides the common terminology considered in literature and this work. Standard pinhole camera model (Eggar, 1998) has been considered as the camera configuration in this work. More insight on basic background and standard assumptions are available in (Byres and Henle, 2004).

1. The point where camera centre (camera axes) lies is termed as '*projection point*' or *'view point*' or even '*eye level*', denoted by *V*.
2. Projections of objects are assumed to be captured on a plane called '*Image plane*', which lies at a distance equal to focal length '*f*' of the camera. Image plane is assumed to be perpendicular to the direction of view.
3. Point of intersection of a line passing through *V* and the image plane such that the line is along the viewing direction (perpendicular to the image plane) is termed as '*image centre*', denoted by O.
4. Camera orientation is represented in the form of *Pitch* (θ_p), *Roll* (θ_{roll}) and *Yaw* (θ_{yaw}).
5. Horizontal line along which vanishing points are lying are termed as *'horizon line*' and represents the eye level or height of the camera (when *roll* of the camera is zero, i.e. if the image plane is perpendicular to the floor, $\theta_{roll} = 0$). If the camera has *roll*, then the horizon line will be tilted by the roll angle.

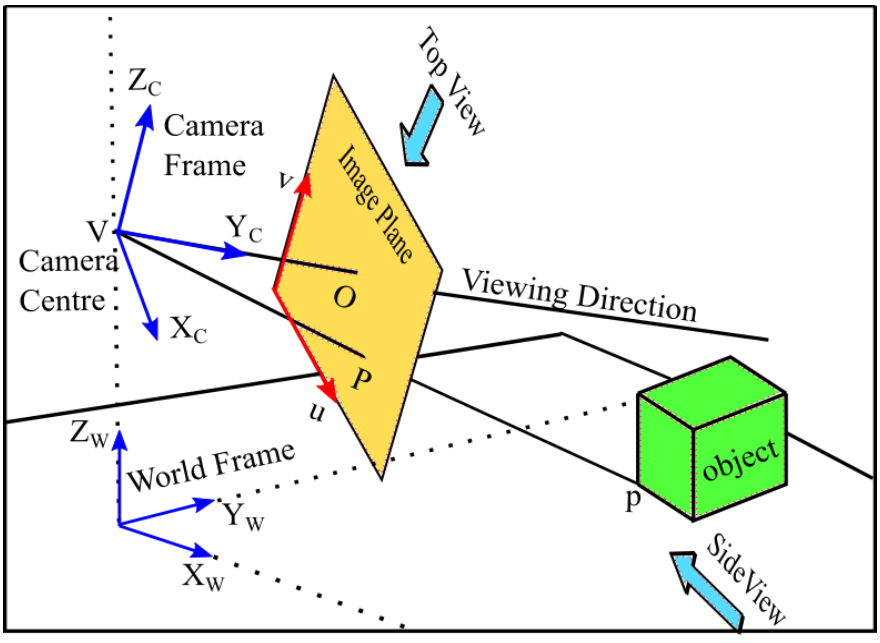


Fig. 2.4: Standard camera model showing the notion of top and side views.

2.3 CORDINATE SYSTEMS

Commonly used coordinate system representation is shown in Figure 2.4. Camera centre is assumed to be at *V* and has its own coordinate system called *Camera Frame*. *World Frame* is the reference coordinate system from where distance to real world objects would be measured. World's *z-axis* is always assumed to be vertical (perpendicular to ground) and will be in-line along the projection of camera centre onto the ground, as shown in Fig. 2.4. Origin of this world coordinate system can be termed as *Station Point*, representing the point on the world where the camera (or the person capturing the image) is stationed. Station point in the top view will be shown as point *O'*. The other two axes (*x* and *y-axis*) are assumed to be on the floor.

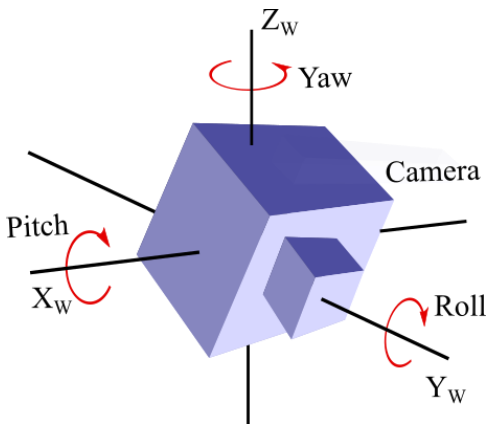


Fig. 2.5: Camera system DoF.

Image plane will itself be having its own 2D coordinate system given in *u* and *v* axes, origin at its bottom left corner as seen from the camera centre. This thesis requires the notion of top and side views, which are also shown in Fig. 2.4. Top view of the model means orthographic view from the top such that the viewing plane is perpendicular to the image plane. Side view of the camera model means viewing from the right side of the camera centre such that the viewing plane is perpendicular to the image plane. Camera frame is aligned according to the 3 DoF of the camera viz. *pitch*, *roll* and *yaw* as shown in Fig 2.5.

Pitch can be considered as the amount of tilt (angle) that the camera exhibits w.r.t. x–axis of the camera coordinate system. It can also be viewed as the tilt angle of the image plane with XY plane. Assuming ground surface to be flat, pitch angle θ_p determines the perspective type of the image (either 2PP or 3PP). If θ_p is zero, then the image captured could be either one-point or two-point perspective. If θ_p has some value, then the image

captured would be three-point perspective in general. Likewise, *Roll* (θ_{roll}) and *Yaw* (θ_{yaw}) of the camera can be considered as the amount of tilt (angles) that the camera exhibits w.r.t. z–axis and y–axis, respectively on a standard camera coordinate system. Coordinate system will be chosen such that z-axis is always perpendicular to the image plane i.e. camera view direction will be assumed to always lie along z-axis and so *Yaw* has no significance ($\theta_{yaw} = 0$). It doesn't affect the type of perspective (2PP or 3PP), except that the direction of view will be changed. Only pitch and roll angles contribute to the proposed mathematical relations.

If the camera has some roll, then the *horizon line* (also the eye level) will not be horizontal, but will be inclined at an angle with the image plane, denoted by θ_{roll}. Other than that, there will be no difference in the method being followed. Throughout this paper, we explain the methods considering horizon line with no roll, i.e. $\theta_{roll} = 0$, for better understanding. However, θ_{roll} has also been considered for the generalized pseudocode (Algorithm 1) given in Appendix A.1.

2.4 SIDE VIEW OF AN IMAGE

The notion of side view for finding the relations and the required parameters in building a map will be introduced in this section. Imagination of side view is bit confusing (without proper explanation with images) and hence the concept will be explained here, considering 2PP case; the same is applicable for 3PP case too, with the only difference that the image plane will have some pitch. In Fig. 2.6, image is shown on the right side with two objects. The green enclosed borders denote the image.

Let the vanishing points of the objects lie along the line connecting VP_1 and VP_2, which in the side view will be at *V*, the view point, as shown in the figure 2.6. This line indicates the eye level or height of the camera from the ground. Note that ground level is not known to us in the side view and it will be taken at some level. Image plane in the side view is shown by a green vertical line at a distance '*d*' from the view point. Let P_1 and Q_1 are the bottom and top nearest corners of the first object near to the camera, as shown in the image. The same points in the side view are shown with the same notations as P_1 and Q_1.

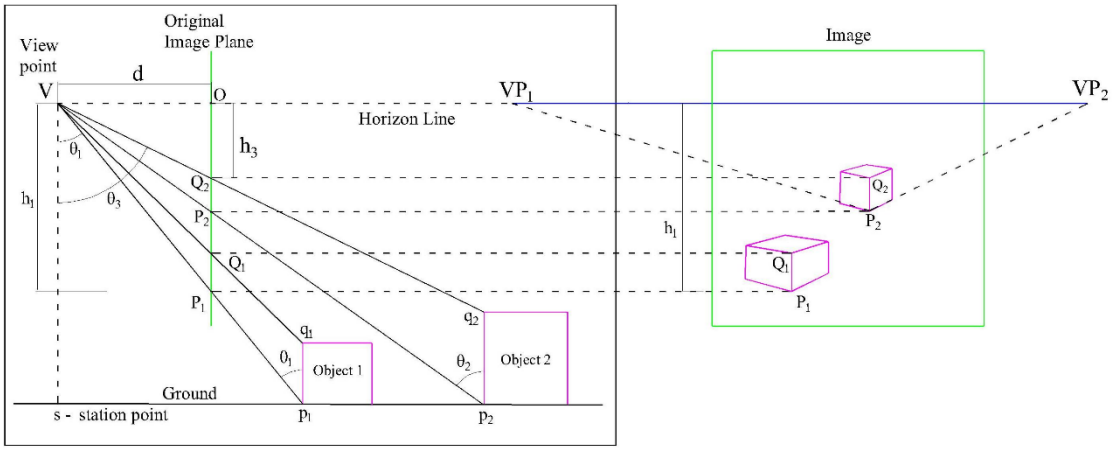


Fig. 2.6: Concept of side view geometry for an example image.

As can be seen, if the image plane is moved farther from *V*, the vertical distance P_1Q_1 will increase, owing to the fact that the image plane is getting closer to the object. The closer the image plane to the camera (or farther from the object), the smaller will be the object captured in the image. Similarly let P_2 and Q_2 be the bottom and top nearest corners for the second object, the same will be captured in the image plane as shown in the side view. The side view thus generated can be used for further analysis of the image for various measurements, and the details are discussed in Chapter 3.

2.5 CONSTRUCTION OF TOP VIEW

There is a standard procedure in the literature for constructing a top view for a given perspective object (considering only one object), for both two-point and three-point perspective geometries (a step-by-step procedure with details needed for top view construction for 2PP image is given in Appendix B). By following this method, top views of 2PP and 3PP images can be created. However, all the objects will be positioned along the horizon line and they appear to be equidistant from the camera. This situation is explained through an example here.

Consider the example of a single 2PP image with three objects as shown in Fig. 2.7. If we follow the standard procedure of construing top view for all the three objects, we will get a new top view of individual objects as shown in Fig. 2.8 denoted as B1, B2 and B3, with their nearest corners having the same vertical distance from the station point.

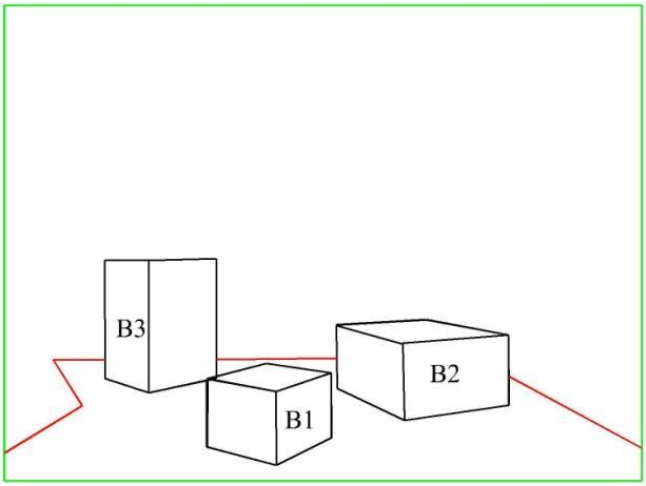


Fig. 2.7: Line diagram of an example 2PP image.

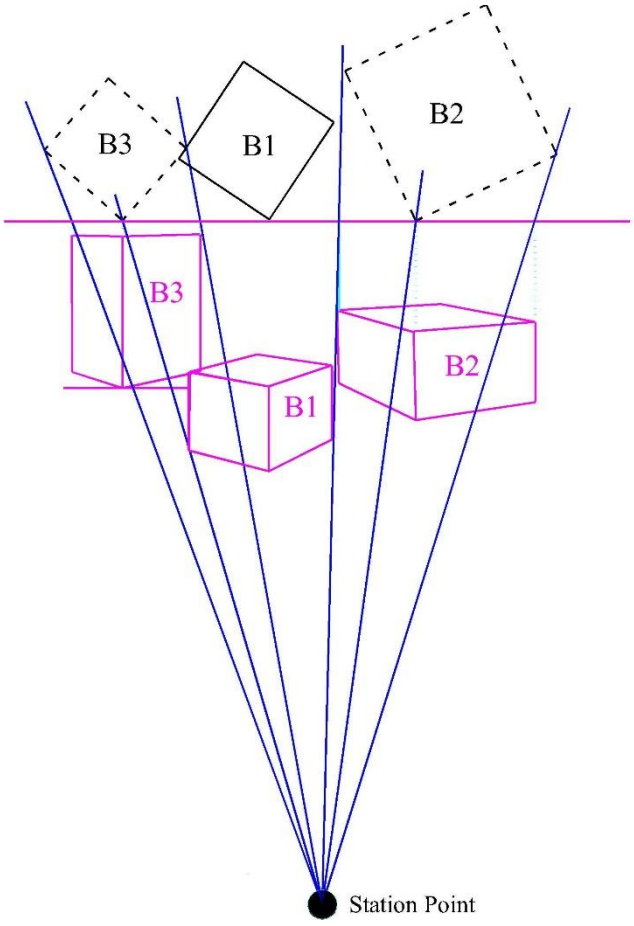


Fig. 2.8: Top view of the objects obtained using traditional method.

However, their actual positions will be as shown in Fig. 2.9 (red coloured rectangles). In order to create a realistic top view, we need a transformation by which they are placed more or less at the same relative position w.r.t. a reference object (we considered the first object, which is nearer to the camera as reference having least ‘y’ distance w.r.t. image plane coordinate system (u,v)). An important task of map building is finding this transformation.

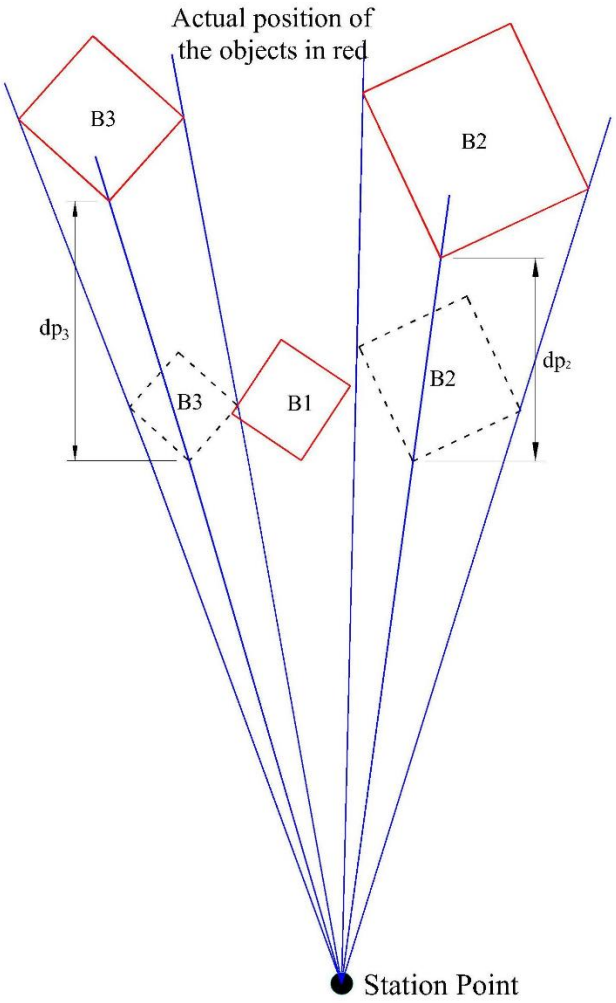


Fig. 2.9: Actual position of the objects in top view.

Denoting the distance to be shifted as d_p, we need to find d_p for all the objects. In addition to this, for 3PP case, actual top view distances of all the objects need to be found, along with other parameters like camera pose (in particular, pitch angle of the camera), and depth of all the objects in the side view. The questions that needed to be answered in this regard are:

1. How to solve for shift distance for all the objects?
2. How to project and bring all the objects to a common scale?
3. How to find the relative pose of all objects w.r.t. a reference object or station point?
4. How to find the camera pitch angle w.r.t vertical, for 3PP case?

Answers to the above questions will help us to generate a 3D map of the environment and the procedure developed is explained in the following chapters.

CHAPTER 3

MAP CONSTRUCTION FROM A SINGLE TWO-POINT PERSPECTIVE (2PP) IMAGE

Using vision/camera for robot navigations in indoor applications when a camera is mounted on the robot usually provide images in two-point perspective. The basic geometrical background required for finding the top view of a given 2PP object in an image and for finding the camera view point (available in the literature) are explained in Appendix B. This chapter discusses the overview of the methodology proposed for building floor maps from single 2PP image using the side view geometry, assuming that the distortion and the skew of the camera with which the image was taken are insignificant..

3.1 OVERVIEW OF METHODOLOGY

Figure 3.1 shows the structure of the procedure that will be followed in the proposed generalized algorithm for getting floor maps. Inputs for the system will be a single captured image from which the details of vanishing points and objects corners will be extracted and fed to the algorithm.

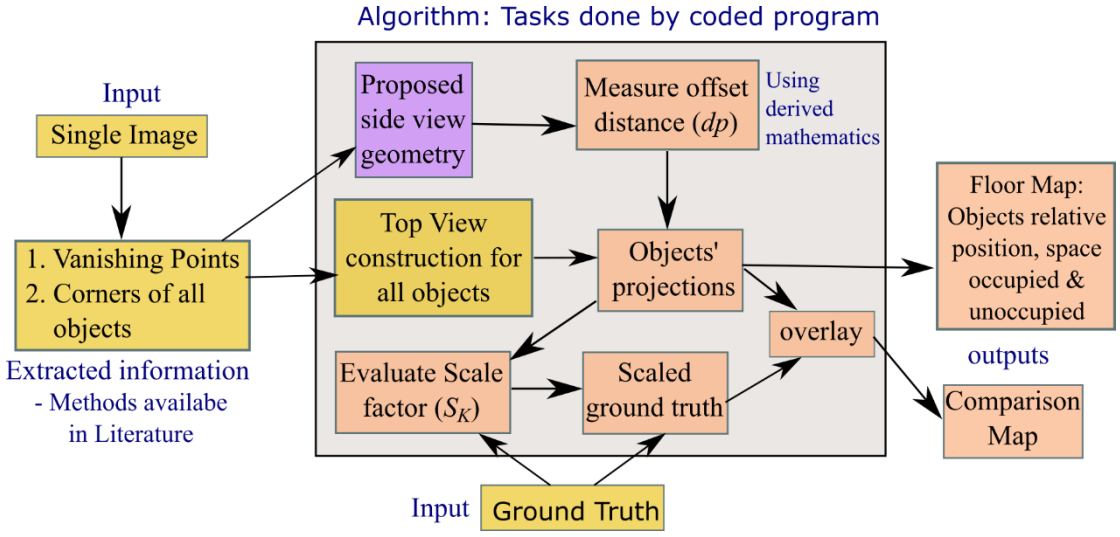


Fig. 3.1: Flow chart depicting the proposed methodology for 2PP case.

Since this thesis is not concerned with getting the required inputs but is rather concerned only with building of floor maps, the inputs will be fed manually to the algorithm. Proposed algorithm builds the top views for all the identified objects, employs the concept of side view geometry and the mathematics related to that for evaluating the

offset distance d_p for all the objects. The scale factor is evaluated by using the input of ground truths and overlays the found object top views onto the scaled ground truth view, taking into consideration the measured distances d_p. The output from the program will be a floor map showing the scaled objects (with a scale factor S_k) positioned relative to one another at a scaled distance (by the same factor S_k).

3.2 ASSUMPTIONS

The following important assumptions are considered while deriving the proposed methods:

1. Position of the image centre *O* and view point *V* are assumed to be same for all the objects w.r.t. the image.
2. Since all the objects in the given single image are on the same image plane, we assume that they exhibit the same orthogonal distances from the view point.
3. Horizon line should be same for all the objects, although vanishing points may be located anywhere along this line.
4. Image plane is always assumed to be along z-axis of the standard camera coordinate system, perpendicular to the camera or viewing direction.
5. Ground (floor) is assumed to be flat and all the objects are assumed to be lying on the same flat floor.
6. At least one object in the image is assumed to be in perspective.
7. Object (in perspective) nearer to the camera will be taken as reference.
8. Left base corner of the given image will be considered as the origin for the uv image plane and all other coordinates (like vanishing points, image centre, objects in top view etc.) will be referenced from this origin.

3.3 PROPOSED METHODOLOGY

3.3.1 Finding Relative Position of Objects

Consider an example image with two objects, base corners of which are denoted as P_1 and P_2 as shown in Fig. 3.2. Figure 3.3 was introduced that depicts the side view of the considered image. The side view shows the position of the camera view point, original image plane and also the positioning of the objects' corners on the original image plane. Depth of first object's base corner, P_1, (on the original image plane) from the horizon is taken as h_1 as shown in Fig 3.3.

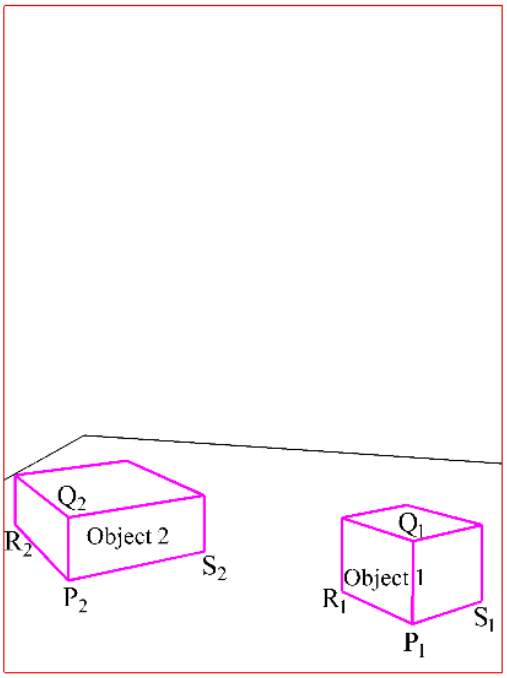


Fig. 3.2: Line diagram of an example 2PP image.

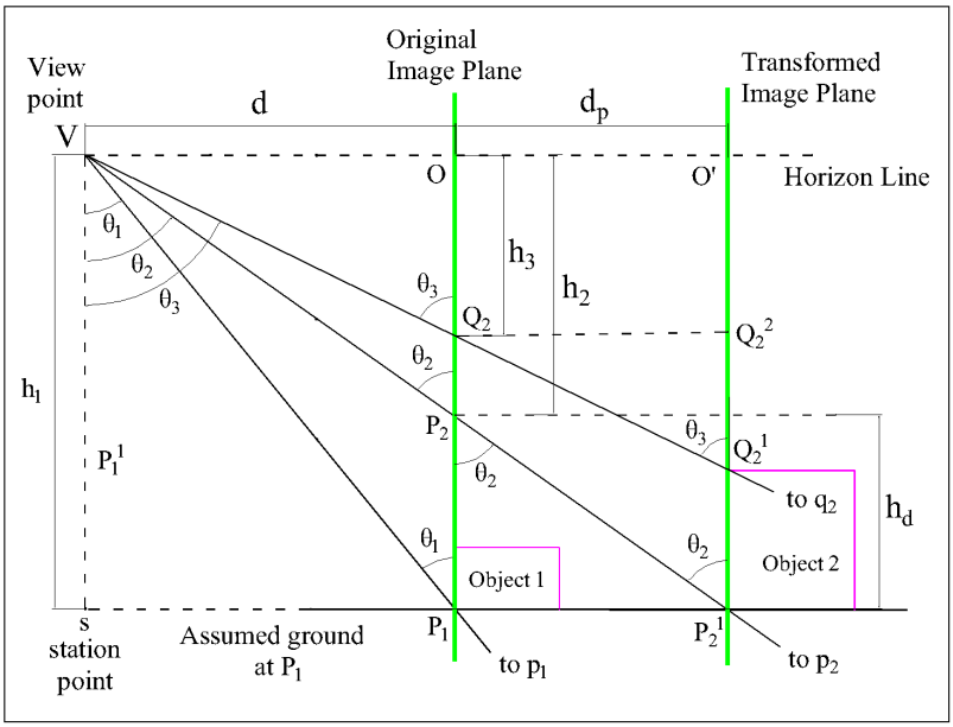


Fig. 3.3: Side view geometry for objects in 2PP.

Depth of second object's base corner P_2 from the same horizon line is denoted by h_2, which is lesser than h_1. But in the real situation, both the objects will be lying on the same flat surface. So, if the second object was made to project such that its depth also measures h_1, then the required offset distance can be obtained. To achieve this, the proposed concept of side view geometry will come handy.

Trying to project P_2 to the same ground depth as P_1 will result in a new image plane positioned parallel to the original image plane at a distance d_p from it, as shown in Fig. 3.3, denoted as '*Transformed Image Plane*'. Let θ_1 and θ_2 be the angles made by the objects' base corners with view point *V*, as viewed from the side view. Let the perpendicular distance between the reference object's base corner and the second object base corner measured by P_1P_2 in the original image plane be h_d. Q_2 in the original image plane denotes the top corner of the second object.

Distance '*d*' of the object from the view point can be obtained from the top view construction. Apart from that, we can get objects' base corners vertical depths (w.r.t. the horizon line) from the image, denoted by h_1 and h_2 as shown in Fig. 3.3, considering two objects. Our problem of interest is to find the shift distance d_p, using which the second object's imaginary plane would be transformed such that the object measures the same depth as h_1.

From triangle VOP_2 in Fig. 3.3,

$$tan\,\theta_2 = \frac{OV}{OP_2} = \frac{d}{h_2} \tag{3.1}$$

Now, from triangle $P_2P_1P_2^1$ and using (3.1), we get

$$tan\,\theta_2 = \frac{d_p}{h_d} \tag{3.2}$$

$$d_p = h_d.tan\,\theta_2 = \left(\frac{h_d.d}{h_2}\right) \tag{3.3}$$

With the finding of d_p, it is possible to list out the steps to be followed for constructing the map, given that there are two objects in the example image considered here.

1. Construct the top view for the first object, measure the depth h_1 and make it as reference depth.
2. Construct the top view for the second object (with the base corner at P_2), and find the ground depth h_2 w.r.t the horizon line.
3. Determine the distance d_p using equation (3.3) for the second object.
4. Marking the station point at *O'*, position the top views of both the objects along the same horizon line. After that, draw construction lines for the second object.
5. Offset top view of the second object such that its base corner, denoted by P_h is now at P_p^1. This transformation is done such that the vertical distance between P_h and P_p^1 is d_p as shown in Fig. 3.4.

6. Identify the visible corners of the ground in the image. For the example considered in Fig. 3.4, corners are denoted by C_1, C_2 and C_3. Find the distances d_p for all the corners and project them with respective distances in the top view. This will be at C_1^1, C_2^1 and C_3^1. Join them by straight lines to denote the boundary of the map.

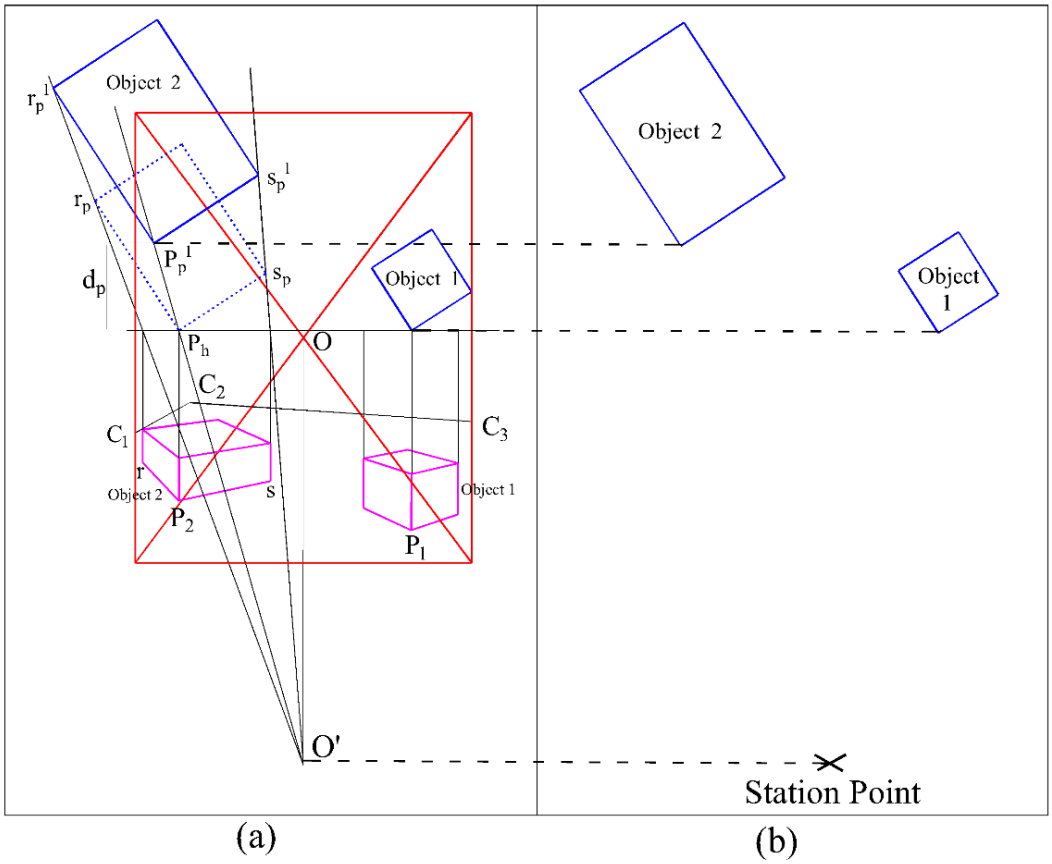


Fig. 3.4: 2PP objects top view. (a) Geometrical projections (b) Relative positioning of objects neglecting all the construction lines w.r.t. the station point

If there are multiple objects (more than two) in the image, repeat the process by keeping the nearest object's depth as the reference. Thus, a relative scaled map can be constructed with ease, utilizing just geometrical projections and using some simple mathematical relations that are proposed here.

3.3.2 Recognizing Free Space Using the Floor Corners

Now that the top view of the given scene showing the relative positioning of the objects w.r.t. the station point has been constructed, the next step in building the complete map would be to find the available free space in the scene. Free space means the unobstructed space on the ground as visible in the given single image on which the robot can navigate. Consider the case of a 2PP image with a floor corner denoted as f_1

on the original image plane, which is positioned at a depth h_f from the horizon line, as shown in Fig. 3.5.

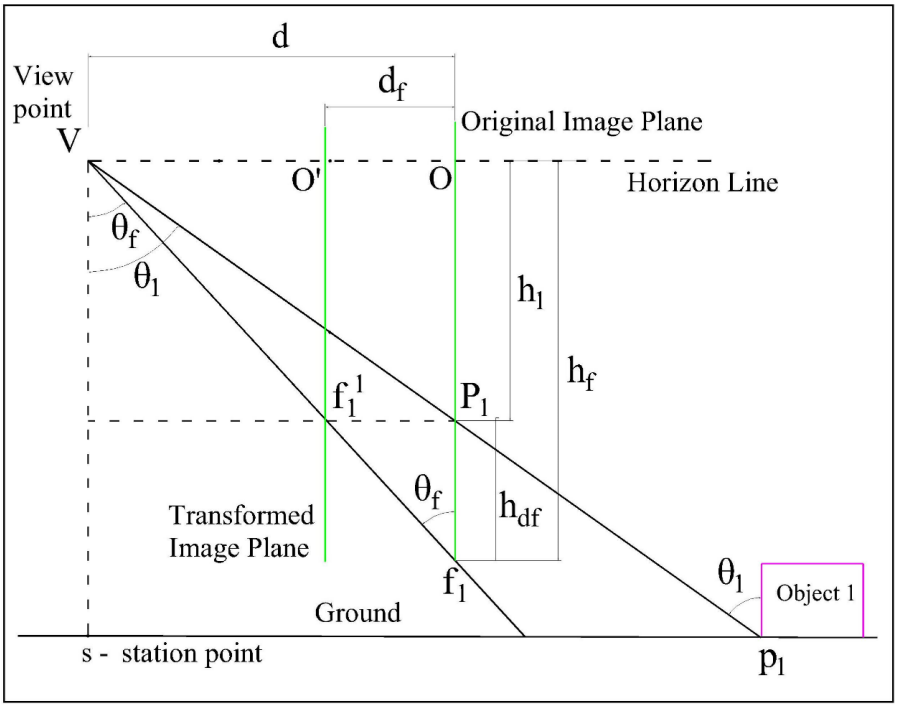


Fig. 3.5: Side view geometry for floor corners in 2PP.

From triangle VOf_1

$$tan\,\theta_f = \frac{OV}{Of_2} = \frac{d}{h_f} \tag{3.4}$$

Now, from triangle $f_1^1 P_1 f_1$ and using (3.8), we get

$$tan\,\theta_f = \frac{f_1^1 P_1}{P_1 f_1} = \frac{d_f}{h_{df}} \tag{3.5}$$

$$d_f = \left(\frac{h_{df}.d}{h_f}\right) \tag{3.6}$$

Equation (3.6) is similar to (3.3) except that the values of distances h_{df} and h_d vary. Using (3.6) we can find the offset distance of the floor corners. Fig 3.6(b) shows the positioning of the floor points in the top view for the example scenario considered in Fig. 3.6(a). Figure 3.7 shows the complete 2D map clubbing identified free space and the occupied space of the objects, along with the relative positioning of the station point.

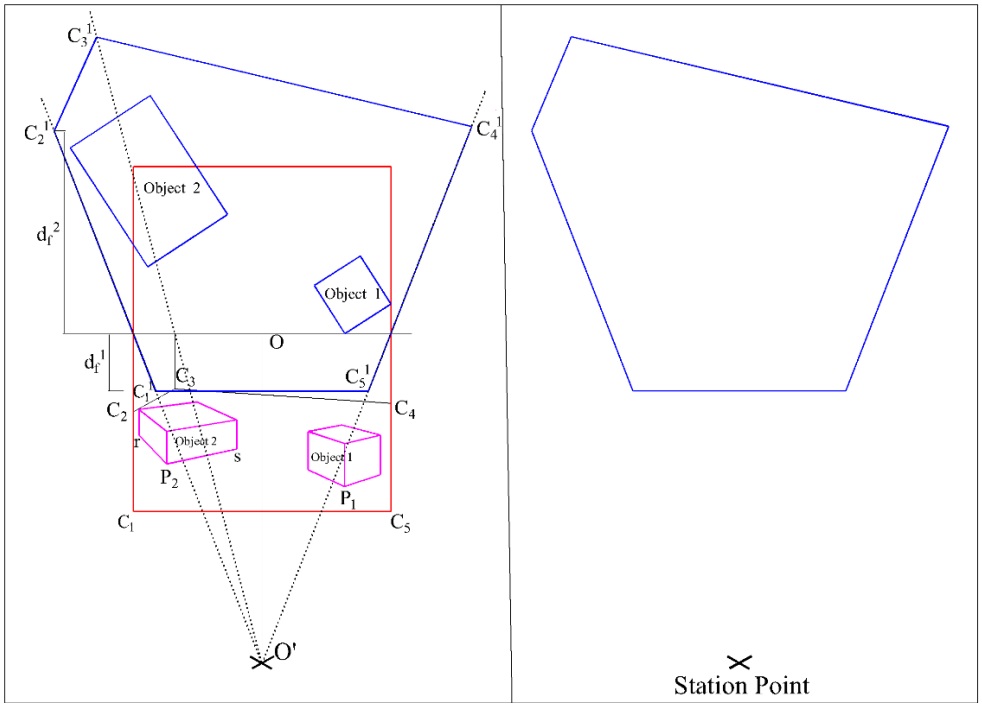


Fig. 3.6: Top view of the floor. (a) Geometrical projections of floor corners. (b) Top view of the floor w.r.t. station point

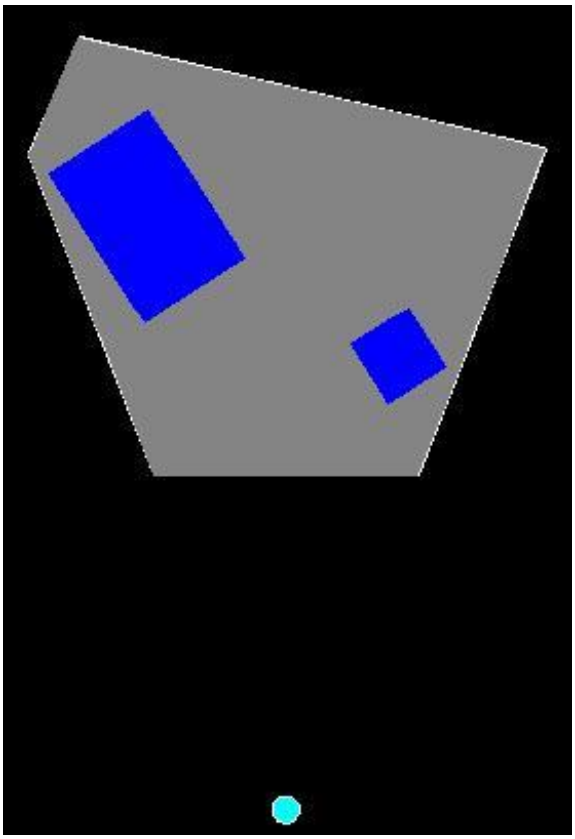


Figure 3.7 Complete Free space map for the 2PP example image in Fig. 3.2.

3.4 FINDING HEIGHT OF OBJECTS FOR BUILDING 3D MAPS

Apart from getting dimensions of the objects for building a 2D map, we can also obtain scaled heights of all the objects by introducing few more relationships. This shows the ability of the proposed methods for constructing 3D maps too. From Fig 3.3, it is clear that the second object's height is measured by the distance P_2Q_2 directly from the image, before projection. But, we have projected the object onto the new plane at *O'*, the object height will now be $P_2^1Q_2^1$.

From triangle VOQ_2 in fig. 3.3,

$$tan\,\theta_3 = \frac{d}{h_3} \tag{3.7}$$

Also, the height of the object is given by

$$P_2^1Q_2^1 = h_1 - O'Q_2^1 = h_1 - (h_3 + Q_2Q_2^1) \tag{3.8}$$

Using triangle $Q_2Q_2^1Q_2^2$, we get

$$P_2^1Q_2^1 = h_1 - \left(h_3 + \frac{d_p}{\tan\theta_3}\right) \tag{3.9}$$

Using (3.7) in (3.9), we get

$$P_2^1Q_2^1 = h_1 - h_3\left(1 + \frac{d_p}{d}\right) \tag{3.10}$$

$P_2^1Q_2^1$ is the required object height scaled by the same factor S_k to the original value. This projected height estimation is required for objects other than the reference one (first object). For the reference object, scaled height can be obtained directly from the image. It has to be noted that all the distances that has been obtained so far are scaled by the same scaling factor S_k. If at least one actual dimension (say length or height of the object/robot) is known, then it is possible to get the actual dimensions of all the objects, as in real environment. Finding S_k will be the key decision factor in determining the actual position and dimensions of the objects, if in case we wish not to use any external measurements like the robot size as an input to the model.

The methods presented above can be used for 3D map building from 2PP image, as well as for estimating the free space in an environment for robot navigation. Experimental results of 3D map constructions, considering heights of the objects for several example scenes are provided in chapter 6 under 2PP results (section 6.1.2).

CHAPTER 4

MAP CONSTRUCTION FROM A SINGLE THREE-POINT PERSPECTIVE (3PP) IMAGE

The procedure for reconstruction of floor map from 3PP image is slightly more cumbersome compared to the 2PP image. The step-by-step process involved in building maps from 3PP images is presented in this chapter along with the derivations of geometrical and mathematical relations. Also, it is shown towards the end of the chapter that the map construction using 2PP image is a special case of that of 3PP case.

4.1 OVERVIEW OF PROPOSED METHODOLOGY FOR 3PP SINGLE IMAGE MAP CONSTRUCTION

Figure 4.1 shows the procedure starting from inputting a single 3PP image till the algorithm generates a floor map, including a comparison map by using ground truth. Inputs for the system considered from a single captured image consist of vanishing points and objects' corners. The inputs will be fed manually to the algorithm.

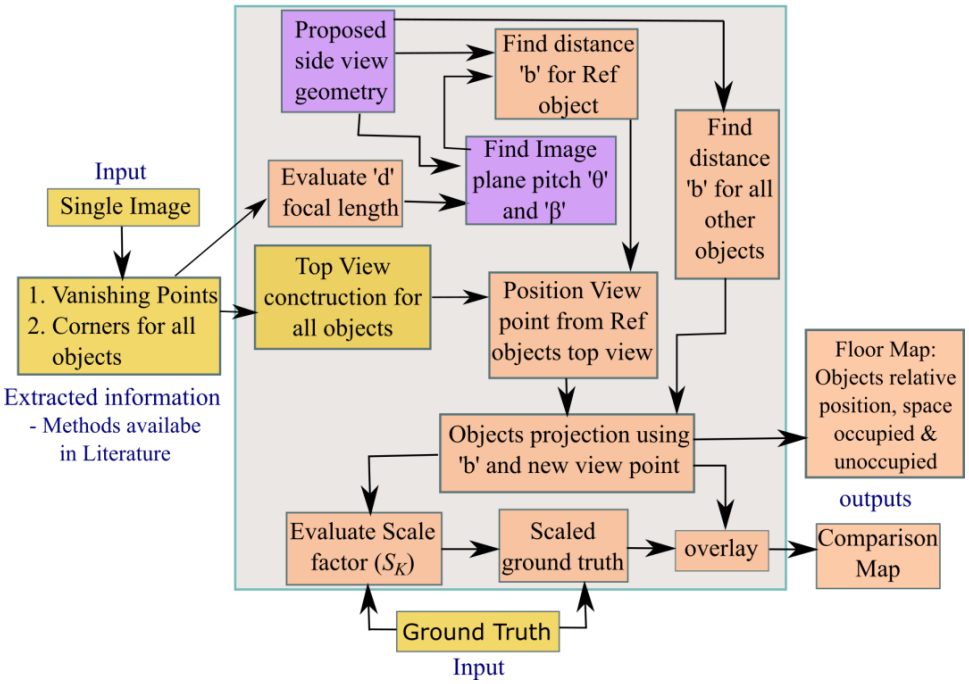


Fig. 4.1: Flow chart depicting the proposed methodology for 3PP case.

The system finds the pitch angle of the camera w.r.t. the world reference frame. It then builds the top views for all the identified objects, employs the concept of side view geometry and the mathematics related to that for finding offset distance d_p for all the objects. The system then evaluates scale factor by using the input of ground truths and

scales down the actual ground truth of objects. Finally, the found object top views will be overlaid on top of the scaled ground truth view, taking into consideration the measured distances d_p. The output from the program will be a free space map showing the objects' scaled dimensions (say with scale factor S_k) positioned relative to one another, distance of which is also scaled by the same factor S_k.

4.2 PRELIMINERY CONCEPTS OF 3PP

Considering a simple case study of a single three-point perspective (3PP) image with two objects in it, this chapter explains the step by step process involved in building maps by introducing and deriving geometrical relations whenever needed. 3PP case arises when the camera (image plane) is neither parallel to any of the surfaces nor parallel to any of the edges of the object. The image looks distorted and vertical edges of objects will not look vertical but rather appear slanted. Figure 4.2 shows an image with two objects in 3PP, along with the object's line diagram. Let us denote the two objects as B1 and B2.

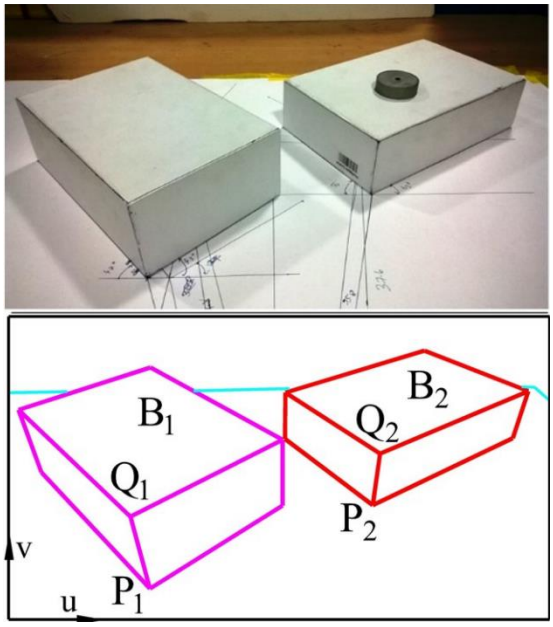


Fig. 4.2: An example image in 3PP with its line diagram.

Before proceeding to build a map, it's necessary to know the camera (hence image plane) pitch angle θ as well as the relative position of the station point. To accomplish this, we need to take each object in the image and analyse for the necessary parameters. The common procedure for the first object (object B1) will be discussed and then the same will be extended for other objects in the given single image. The object with its base corner (denoted by P_1) nearest to the image bottom will be considered as the

reference object. In other words, the object which appears to be near to the camera is considered to be the reference (we can consider any object as the reference, the only difference is that the remaining objects in the image should be mapped based on this new reference).

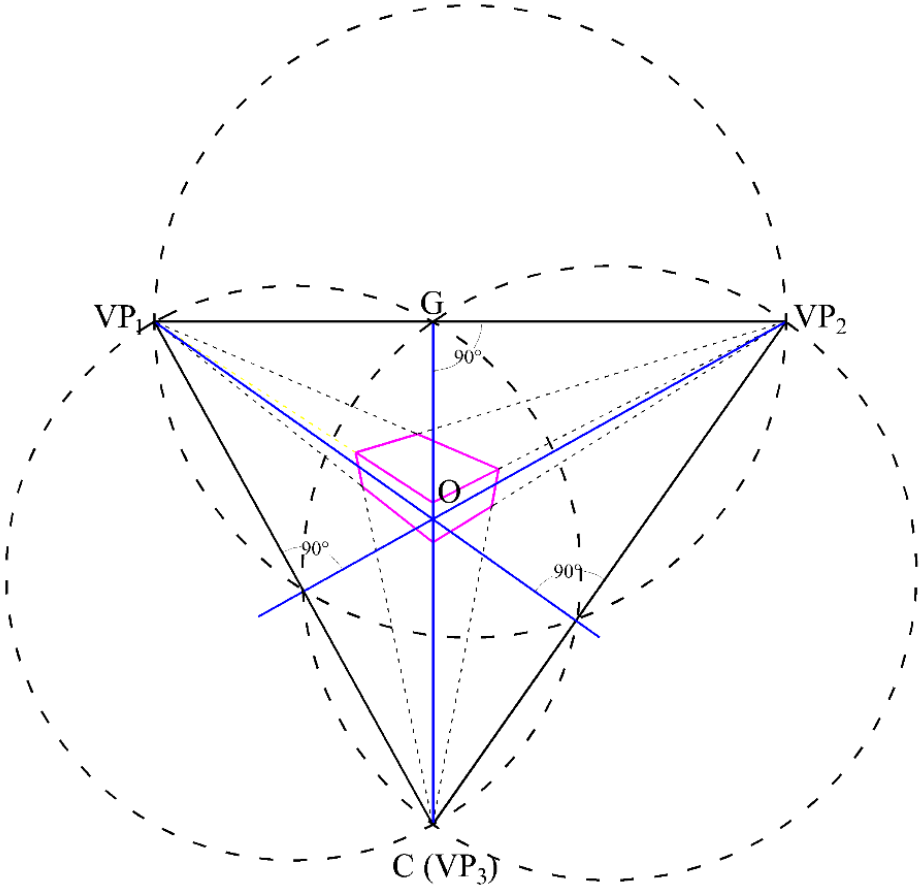


Fig. 4.3: Finding image center and vanishing points.

As a first step, the view point (camera location, denoted by *V*) needs to be established. Mauldin (Mauldin, 1985) gave a simple mathematical relationship for finding the view point (camera location) in 3PP images using vanishing points. The view point projection onto the image plane (termed as image centre, denoted by capital *O*) should lie at the point of intersections of altitudes of the triangle formed by joining the three vanishing points as shown in Fig. 4.3. The view point will lie on the plane passing through *O* and orthogonal to the image plane. The distance between the view point and the image centre, denoted by *d*, depends on the vertical lengths *OG* and *OC* and is given as (Mauldin, 1985):

$$d = \sqrt{||OG||.||OC||} \qquad (4.1)$$

Once the position of the view point is known, a simple architectural method (Ching, 2010) exists for constructing the top view of a given single parallelepiped object by using the object's vanishing points. Since the position of the image centre (O) is known, we can now position its vertical projection onto the vanishing circle as O', as shown in Fig. 4.4 (procedure for constructing top view of a single object for 2PP is given in Appendix B, which is also applicable for 3PP except that the image centre is found using the procedure described above). Figure 4.4 shows the top view of object B1. This construction is of not much help as it gave only a scaled rectangular top view which is just positioned along the horizon line. Apart from that, the camera pitch θ and the position of station point is not yet known. If we somehow get θ and the relative distances of all the objects w.r.t. the view point, then we can position them accordingly and get a floor map.

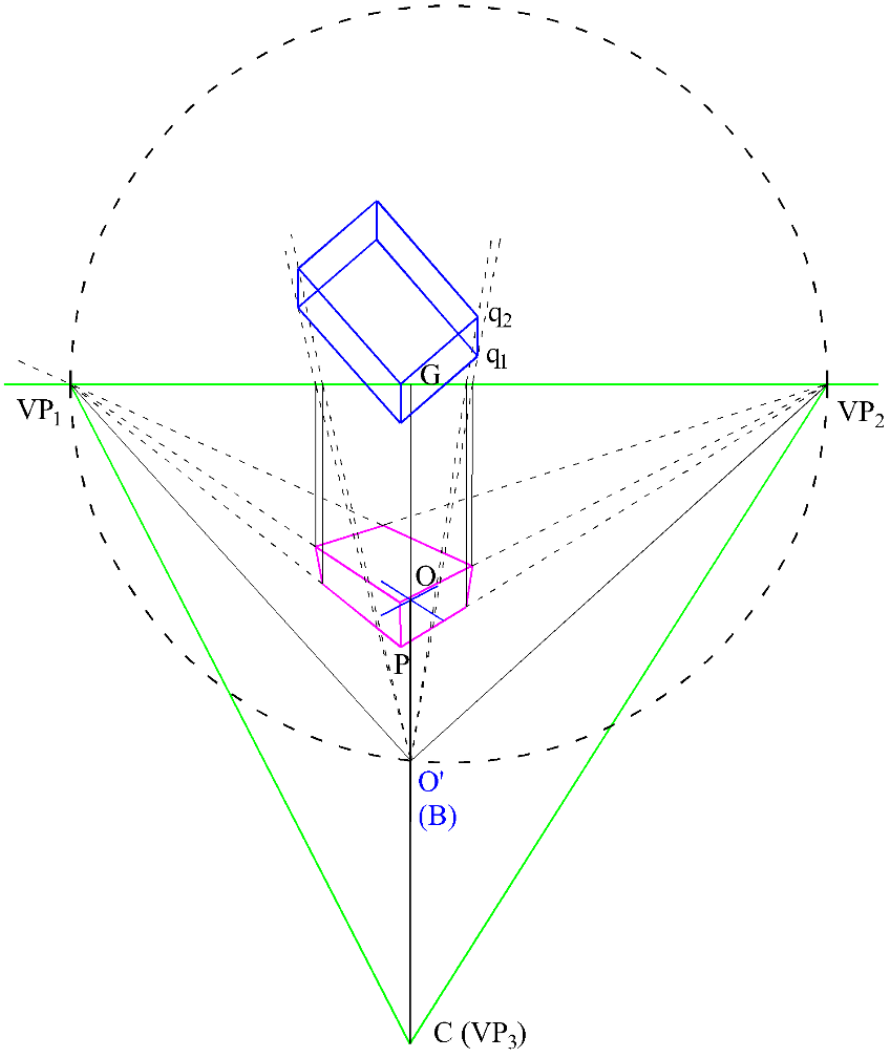


Fig. 4.4: Geometrical construction for top view of single 3PP object.

4.3 DERIVATION OF MATHEMATICAL RELATIONS FOR ESTIMAING GEOMETRIC PARAMETERS

For finding the camera pitch (θ) and to derive all the other necessary mathematical relations, we need to construct a geometrical side view of the given scenario. Figure 4.5 shows an imaginative deduction of the side view of the image plane relative to the view point. What we see in the image plane will consist of the scene with objects B1 and B2 (for the example considered in Fig. 4.2), but the same image plane in side view will look like a tilted line. For the object B1, the method for generating the appropriate side view can be explained with the help of Fig. 4.6 and Fig. 4.7.

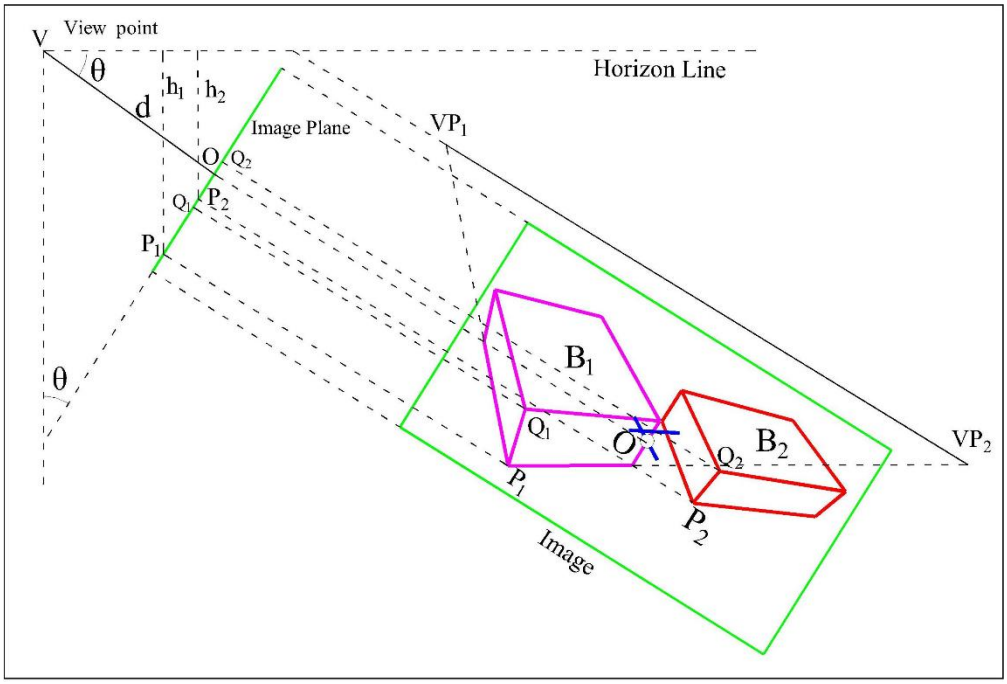


Fig. 4.5: Imaginative side view.

In Fig. 4.6, the nearest bottom and top corners of B1 are marked on the image plane as P_1 and Q_1. So, the actual position of the base corner P_1 in the world should lie along the line passing through P_1 (on the image plane) and the view point V as shown in Fig. 4.6. This will have an infinite number of possible positions. Assuming some arbitrary ground level (which is not known), base corner projection of P_1 onto the ground can now be deduced by the point at which the extended line meets this arbitrary ground level, marked as p_1 in Fig. 4.6. The intersection point of the vertical line through p_1 (lying on the ground) and the line passing through Q_1 and V will give the position of the top corner of the object, denoted by q_1 in Fig. 4.6. By similar procedure, we can now construct the whole object, assuming that it is lying on that arbitrary ground level.

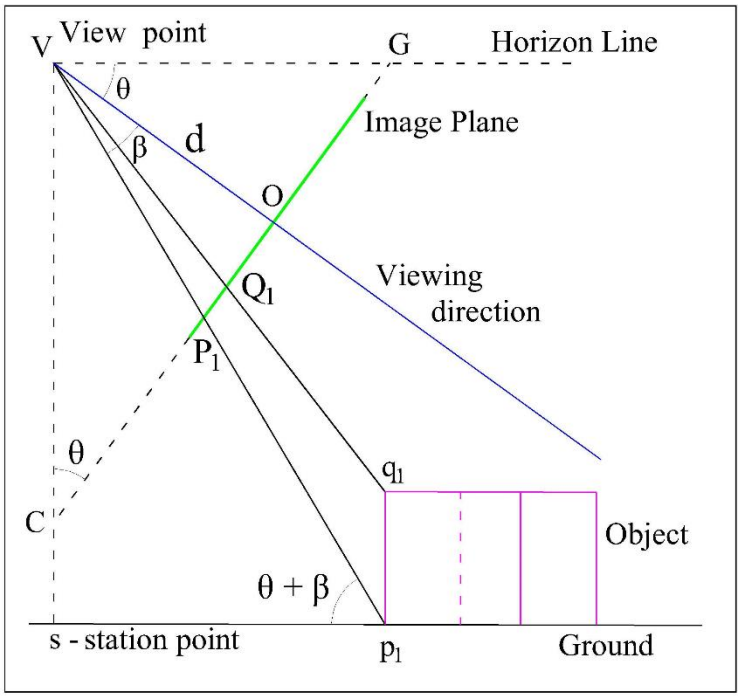


Fig. 4.6: Side view depicting an object and camera view point in 3PP.

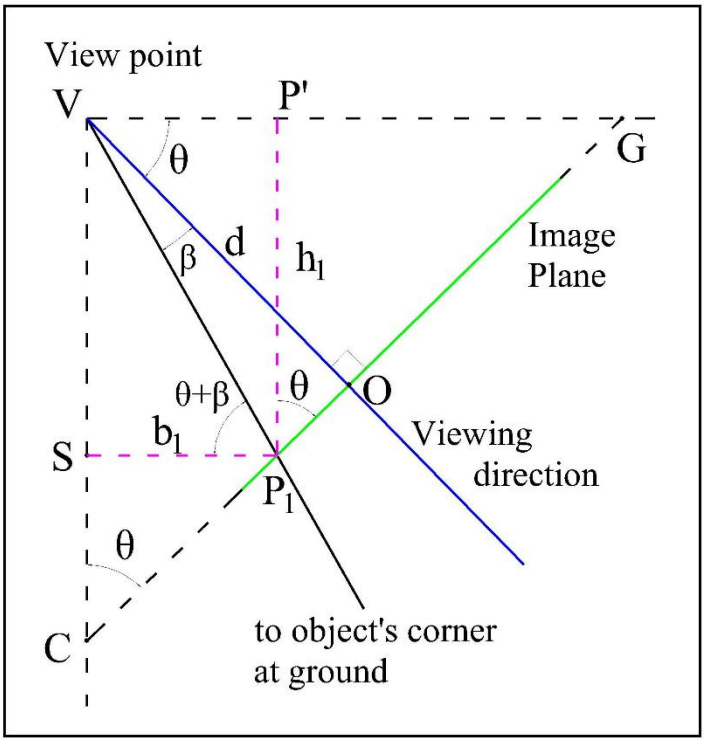


Fig. 4.7: Side view with construction details.

Figure 4.7 shows the same side view as in Fig. 4.6 with more details. From the perspective image with vanishing points being given, we can get the measures *OG*, *OC*, OP_1 and *d*. It has to be noted that G will lie on the horizon line while *C*, the third vanishing point (VP_3) will lie on the vertical line through *V* (Jang, 2001). From triangle *VOC*, we can get the value of camera inclination θ, using the relation

$$tan\,\theta = \frac{VO}{OC} = \frac{d}{OC} = \frac{\sqrt{OG.OC}}{OC} = \sqrt{\frac{OG}{OC}} \quad (4.2)$$

Or, from triangle *VOG*

$$tan\,\theta = \frac{OG}{OV} = \frac{OG}{d} = \frac{OG}{\sqrt{OG.OC}} = \sqrt{\frac{OG}{OC}} \quad (4.3)$$

So, in both cases, we get the same result for measuring θ, which signifies that VP_3 lies on the same vertical as *V*.

Finally,

$$\theta = tan^{-1}\left(\sqrt{\frac{OG}{OC}}\right) \quad (4.4)$$

Now that the camera pitch angle θ is known, the depth of the object base corner (same as depth of ground plane) from V can be obtained to some scale (say by a scaling factor S_k) from the triangle P_1GP' as

$$cos\,\theta = \frac{PP'}{PG} = \frac{h_1}{P_1G} \quad (4.5)$$

$$h_1 = P_1G.cos\,\theta \quad (4.6)$$

From triangle *VOP*

$$tan\,\beta = \frac{OP_1}{OV} = \frac{OP_1}{d} \quad (4.7)$$

In Fig. 4.7, from triangle VSP_1, we can get the horizontal distance from station point to object's nearest corner, again, distance scaled by the same scaling factor S_k. Note that $VS = P_1P' = h_1$.

$$tan(\theta + \beta) = \frac{VS}{SP_1} = \frac{h_1}{SP_1} \quad (4.8)$$

$$b_1 = SP_1 = \frac{h_1}{tan(\theta + \beta)} \quad (4.9)$$

Here, b_1 denotes the distance between the station point and the object's nearest base corner in the side view as shown in Fig. 4.7. The above relationships can be used for generating the top and side views of an image with multiple objects, as explained I nthe following section.

4.4 SIDE VIEW GEOMETRY OF SEVERAL OBJECTS IN A GIVEN 3PP SCENE

In section 4.1 and 4.2, the mathematical relations for finding camera height, pitch angle, object's relative distance and dimensions from station point have been derived considering a single object. However, trying to follow the same procedure for all the objects in the given image will result in the top view with all the objects aligned side by side along the horizon line, as shown in Fig. 4.8(a) for the image in Fig. 4.2. The finding of b_1 for the reference object B_1 in Fig. 4.7 simply means that in top view, the perpendicular distance from the station point to the base corner will be b_1.

So, marking a point at a distance b_1 from the base corner of B_1 as shown in Fig. 4.8(b) will give the exact position of the station point relative to the first object's base corner. It has to be noted that the object as well as the distances are all now scaled by some factor, which is not to the actual scale as in real world. Trying to visualize only the two object projections and the station point will look something like shown in Fig. 4.8(b), which clearly is not much useful. The actual position of the second object B_2 relative to B_1 and station point is shown in Fig 4.8(c). We know b_1 for the first object but we are yet to get b_2 and B_2's dimensions that need to be scaled to the same scale factor as B_1. Only then, we can say that a meaningful map is developed. So, further mathematical relations are needed in order to find out this offset distance and the scaling of the rest of the objects.

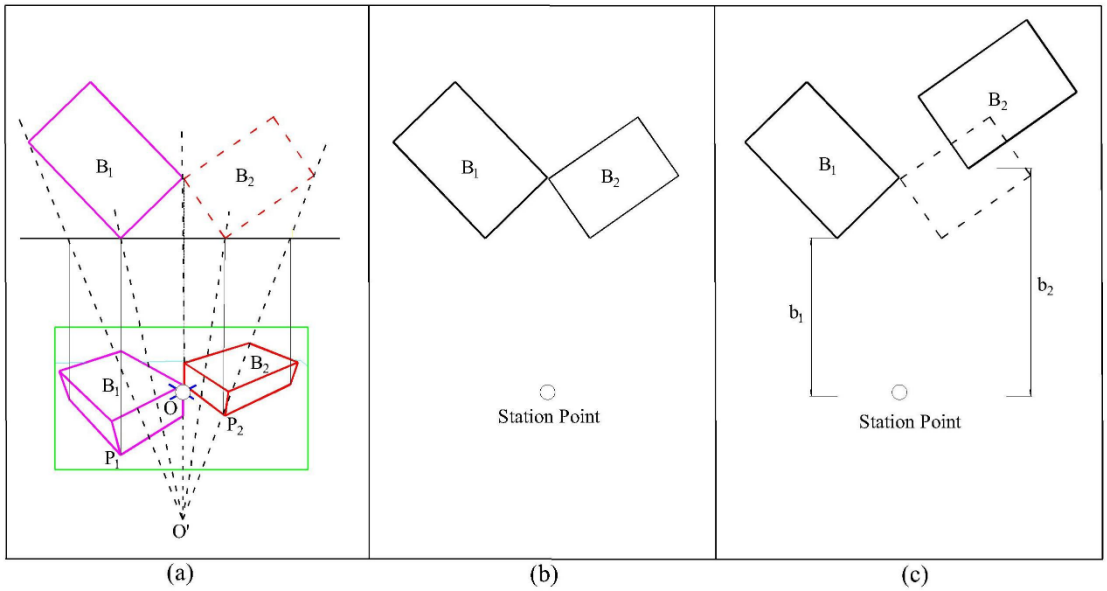


Fig. 4.8: Top view construction. (a) Traditional geometrical construction (b) Obtained top view (c) Actual top view

All the important assumptions introduced for 2PP case in section 3.3 are applicable for 3PP case too. Apart from that, we consider an additional assumption of image plane angle θ being same for all the objects under consideration. The same idea proposed for 2PP case will also be followed for 3PP case too, i.e. bringing all the objects to a common reference ground depth (scaled) w.r.t. the horizon line as viewed in the image. This will be explained with several figures.

Figure 4.9 shows the actual configuration of side view, considering two objects, as shown in Fig. 4.2. It is obvious that the two objects are lying on the same ground surface. Although the ground level (in other words, the height of *V* from ground surface) is not known, we assume some random height just for visualization purpose and for deriving the relations. Projection of the objects' corners onto the image plane has been marked in Fig. 4.9. Figure 4.10 shows what actually projections of objects mean relative to the image plane. According to this configuration, it is clear that the depth of the second object is less than the depth of the reference object.

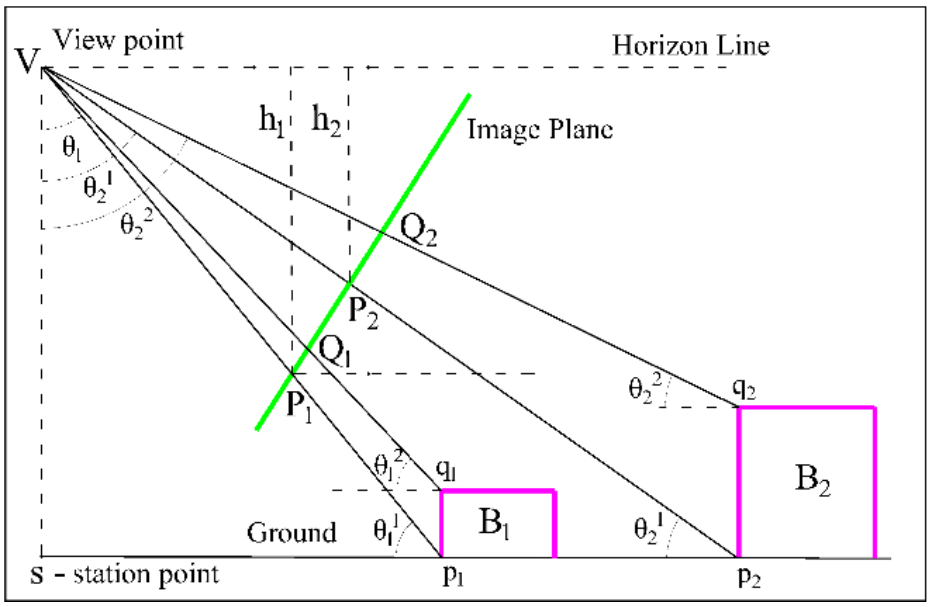


Fig. 4.9: Actual side view of two objects.

This is applicable even for any scene with several objects. Denoting the depth of B_1 from the horizon line as h_1 and likewise denoting the depth of B_2 from the horizon line as h_2, it is clear from the side view that $h_1 > h_2$. But in the real world, the two objects are lying on the same floor. So, the basic idea that we have developed here is to bring all objects (for now only two) to a common depth relative to the horizon. Since, for the reference object B_1 we already have the measurements and distance, it is very

convenient to project other object (B_2) until it measure same depth as B_1, i.e. we aim at $h_1 = h_2$ and if there are several objects we make all depths to be same as h_1.

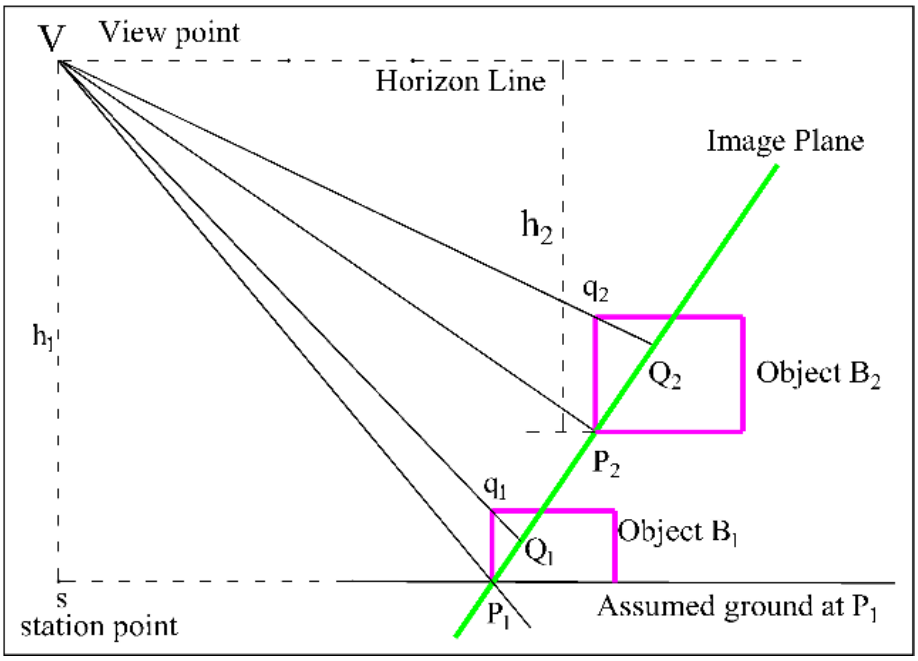


Fig. 4.10: Deduced configuration.

Figure 4.11 shows the basic concept that is proposed to achieve this. Assuming an imaginary image plane (denoted as transformed image plane) which will be shifted such that P_2 is now at P_2^1. The two objects are projected to the same ground level. Now, the proposed configuration will be used in the further sections to find the offset distances of multiple objects, their relative positioning and their projected dimensions.

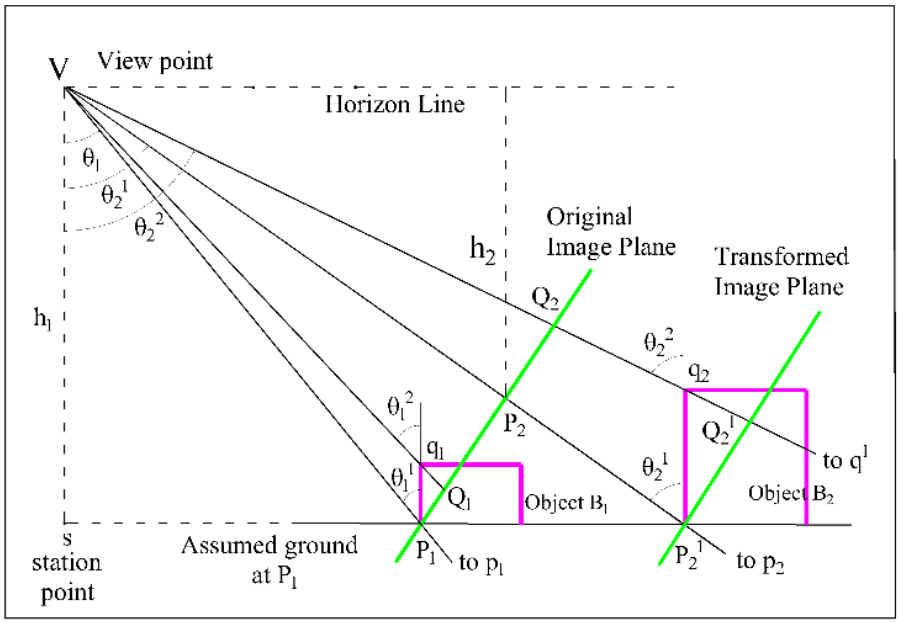


Fig. 4.11: Side view with the transformed image plane.

4.5 FINDING OFFSET DISTANCES OF MULTIPLE OBJECTS IN THE IMAGE

The following methodology has been proposed for projecting the second object B_2 to the same depth as the depth of the first object B_1:

1. For the first object, find the scaled depth h_1 and horizontal distance b_1 using equations eq. (4.6) and eq. (4.9) respectively.
2. For the second object, find h_2 using eq. (4.6).
3. Determine the difference of the depths ($h_1 - h_2$).
4. Transfer the image plane to a new location such that the new transferred image plane is parallel to the actual image plane and the new projection of the object's corner on this image plane i.e. P_2^1 lie on the same horizontal as P_1, as shown in Fig. 4.12. This is a detailed view of the triangle $P_1P_2P_2^1$ shown in figure 4.11. By this way, the object has been artificially projected onto the new image plane such that they now have the same scale.
5. Project image centre O until it meets this new image plane at O'.

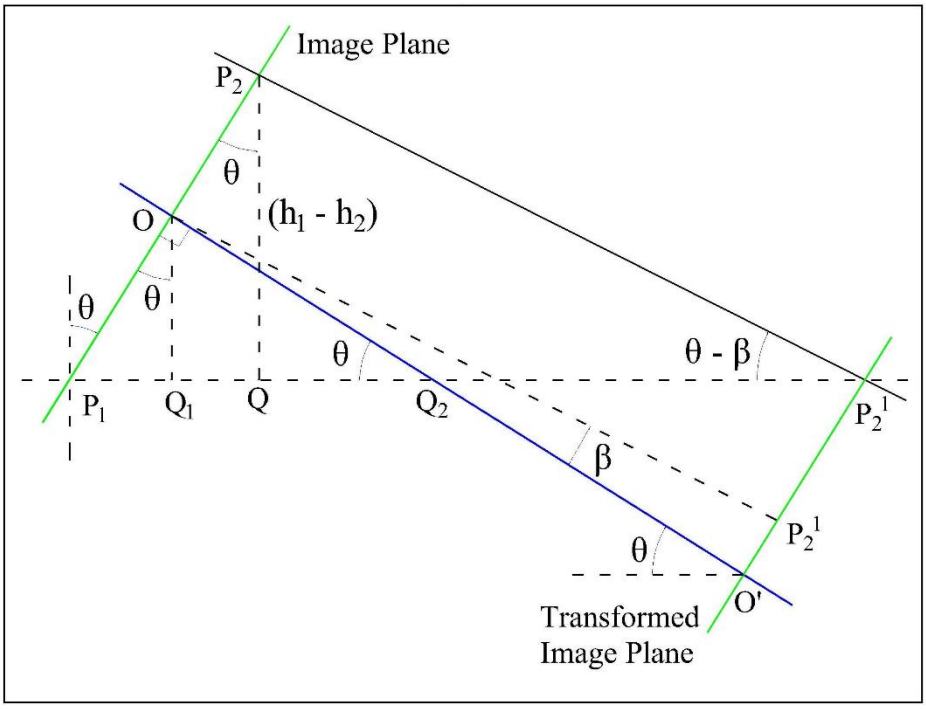


Fig 4.12: Geometry for offset distance measurement.

Figure 4.12 shows all other angles and notations considered for deriving mathematical relations for the case when the image centre is positioned somewhere between the two objects corners. Here $P_2Q = (h_1 - h_2)$.

From triangle $P_2P_2^1Q$ in Fig. 4.12, we get

$$sin(\theta - \beta) = \frac{P_2Q}{P_2P_2^1} \tag{4.10}$$

$$P_2P_2' = \frac{P_2Q}{sin(\theta-\beta)} = \frac{h_1-h_2}{sin(\theta-\beta)} \tag{4.11}$$

$$tan(\theta - \beta) = \frac{P_2Q}{QP_2^1} \tag{4.12}$$

$$QP_2' = \frac{P_2Q}{tan(\theta-\beta)} = \frac{h_1-h_2}{tan(\theta-\beta)} \tag{4.13}$$

From triangle $O\overline{P_2^1}O'$ in Fig. 4.12, and observing that $O\overline{P_2^1} = P_2P_2^1$

$$cos\,\beta = \frac{OO'}{P_2P_2^1} \tag{4.14}$$

Thus,

$$OO' = P_2P_2^1 cos\,\beta \tag{4.15}$$

New distance *d'* from view point *V* to the transferred image plane centre *O'* is therefore given by

$$d' = d + OO' \tag{4.16}$$

From triangle P_1P_2Q, we get

$$tan\,\theta = \frac{P_1Q}{P_2Q} \tag{4.17}$$

$$P_1Q = P_2Q\,tan\,\theta = (h_1 - h_2)tan\,\theta \tag{4.18}$$

The horizontal distance b_1 has already been measured from the first object's base corner (i.e. P_1) to the station point. P_1Q and QP_2^1 are also determined from equations (4.18) and (4.13) respectively. Therefore, the second object's base corner horizontal distance (to the same scale as the first object) can be measured as:

$$b_2 = b_1 + P_1Q + QP_2^1 \tag{4.19}$$

Equations (4.10) – (4.19) are applicable for two cases:

- Case 1: When the image centre *O* lies in between P_1 and P_2.
- Case 2: When the image centre *O* lies below both P_1 and P_2.

Case 3 happens when the image centre O lies above P_1 and P_2 . Equations (4.10) – (4.19) are all applicable except for a small change. In-place of β, we use $-\beta$ and all other equations remain same. This situation is shown in Fig. 4.13. In general, for images with

several objects, the first object (farthest most from horizon) has been considered as the reference. With the help of the reference, and considering each object one at a time, it's relative position w.r.t. P_1 and O will be found, then β or $-\beta$ cases will be considered accordingly; to finally find the base distance b_i (for i^{th} object).

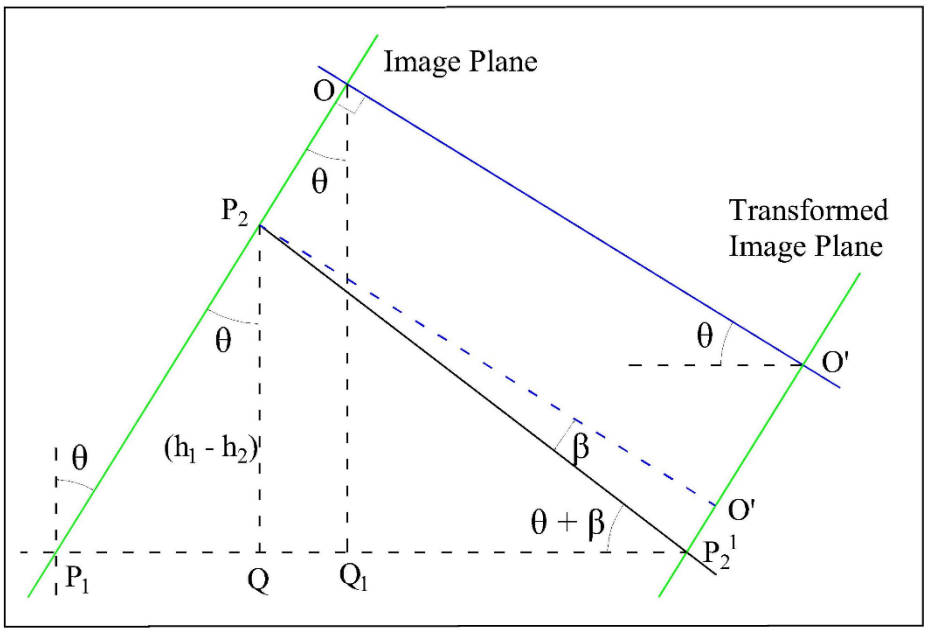


Fig. 4.13: Geometry for case 3.

4.6 RELATIVE POSITIONING OF MULTIPLE OBJECTS

Now that the horizontal distances for all the objects are found, a 2D map can be constructed using the relations (and hence the distances) derived. Figure 4.14 shows the construction process with two objects as considered in the example image. The basic procedure proposed is:

1. For the first object B_1, construct the top view with image centre being at O.
2. Find d, θ, h_1, object dimensions and finally horizontal distance b_1 as given in equations (4.2) – (4.9).
3. Since b_1 is known, position the station point in the top view, denoted by O_n in Fig. 4.14(a). Position O_n such that its vertical distance is b_1 from the first object's base corner.
4. For the second object B_2 also, construct the top view, three corners of which will be positioned at r_p, P_h *and* s_p.
5. Find b_2 using (4.19) for the second object.
6. Now, in the orthographic view, project the second object corners r_p, P_h *and* s_p along their corresponding projection lines such that the vertical distance between O_n and

the new position of P_h (call it P_P^1) is b_2. The respective transferred positions of the other two corners for the second object will then be at r_P^1 and s_P^1 as shown in Fig. 4.14(a).

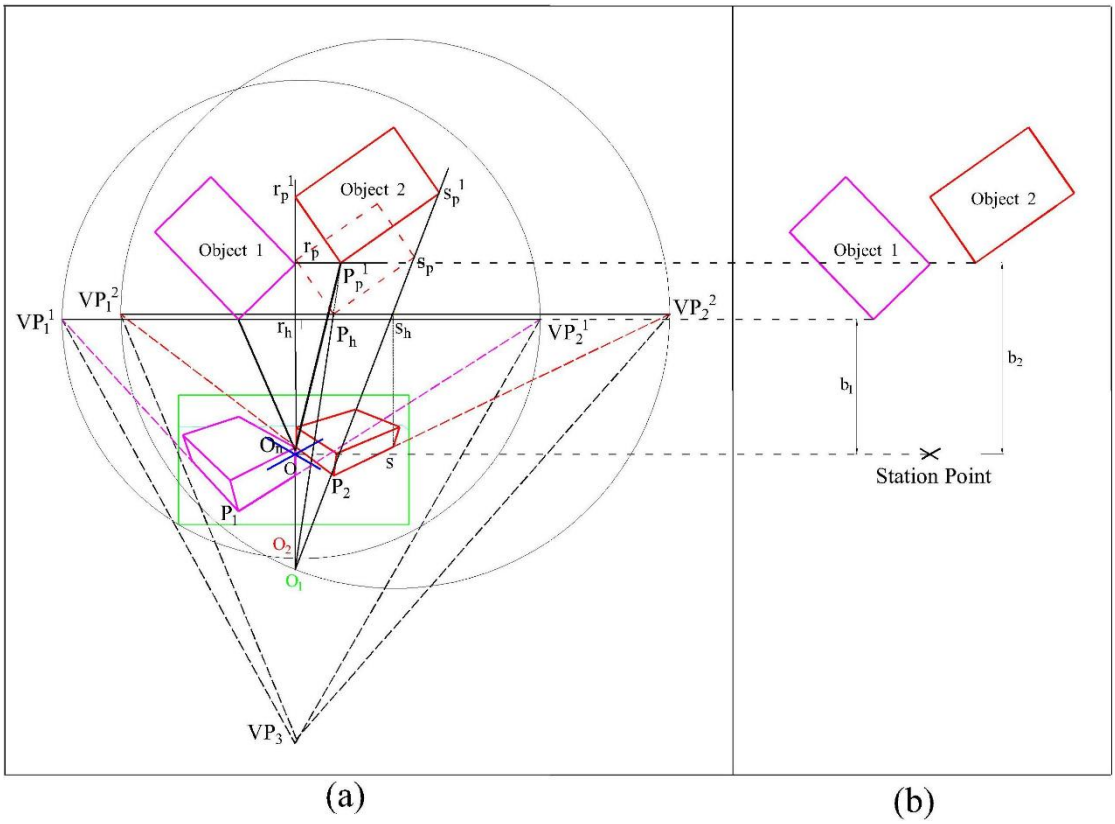


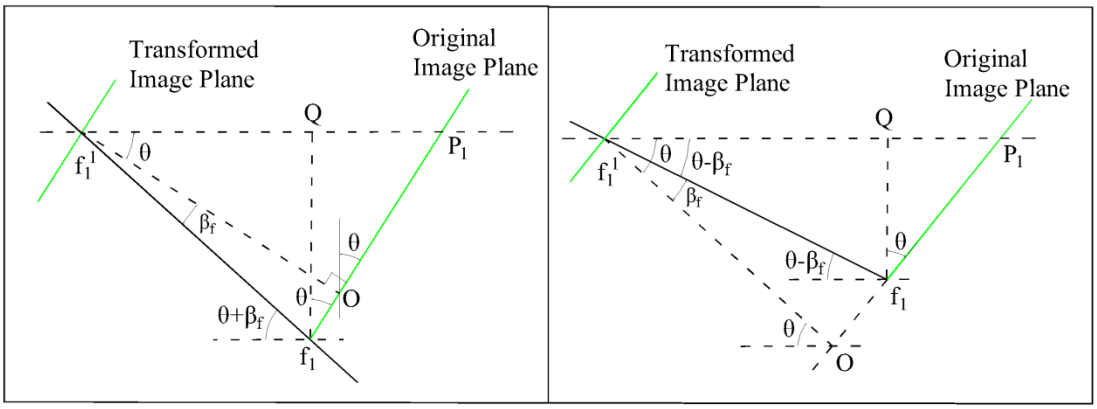
Fig. 4.14: Top view construction of objects. (a) Geometrical construction (b) objects relative positions in top view

It should be noted that, because the vanishing points detection are challenging and are not always accurate, horizon lines appear to be different for both the objects. But, in reality, they should be along the same horizontal since they indicate the eye level of the viewer/camera. We now have top views for the two objects, both being to the same scale factor S_k. Figure 4.14(b) shows just the objects and the view point, neglecting all the construction lines. By measuring the relative distance between the objects, it can be verified that all the object dimensions are scaled to the same scale by which the objects are positioned. In other words, a floor plan has been constructed, scaled to S_k. If one real dimension is known in the environment, then, the whole constructed map can be scaled by the ratio and hence a map of real-world scale can be easily obtained.

4.7 MATHEMTICAL RELATIONS FOR FREE SPACE

Until now, discussions have been carried on how to construct objects relative to one another and w.r.t. the camera station point. In order to complete the process of building *free space map*, it is also necessary to use the visible floor corners of the given scene, in order to make the boundary of the space where the objects will be positioned according to the given scene.

The free space boundary map can then be used by the robot for its navigation. This subsection deals with derivation of mathematical relations relevant to floor space estimation. When the image is in 3PP, three cases (two cases will have a common configuration) will arise, depending on the position of the floor corners with respect to the image centre (O), as shown in Fig. 4.15.



(a) case 1 and case 2 (b) case 3

Fig 4.15: Geometries for floor space estimation.

Consider Fig. 4.15(a) which is applicable for both case 1 and case 2, where the floor corners of interest are below O. From triangle P_1f_1Q, we get

$$P_1f_1 = \frac{f_1Q}{\cos\theta} \tag{4.20}$$

New distance d' from view point V to the transferred image plane centre O' is therefore given by

$$d' = d - OO' \tag{4.21}$$

From triangle $O'f_1Q$, we get

$$tan(\theta + \beta_f) = \frac{f_1Q}{O'Q} \tag{4.22}$$

Equation (4.22) is same as eq. (4.12). Similarly, from triangle P_1f_1Q, we get

$$P_1Q = f_1Q \tan\theta \quad (4.23)$$

Equation (4.23) is same as eq. (4.18). Finally, just like in eq. (4.19), we can measure the floor corners horizontal distance (to the same scale as the first object) as

$$b_2 = b_1 - O^1Q - P_1Q \quad (4.24)$$

For case 3, the same equations (4.20) to (4.24) applies except that instead of β_f we use $-\beta_f$ in all the relations, because of the geometrical arrangement as shown in Fig 4.15(b).

Figure 4.16 shows the result of the complete 2D-map constructed by the proposed algorithm for the image shown in Fig. 4.2. Grey area in the map represents the boundary within which the robot can operate. In other words, it represents the free space available for the robot. Blue coloured polygons represent the relative positioning of the objects w.r.t. the station point which is shown as cyan coloured blob.

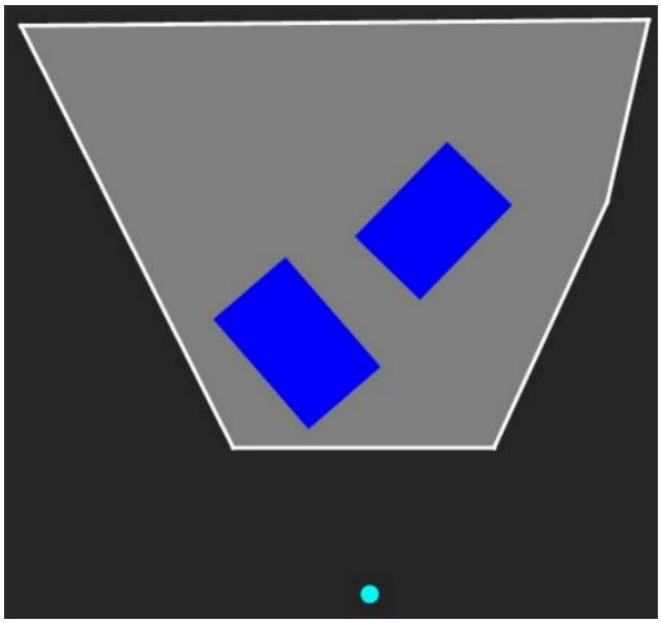


Fig. 4.16: A complete 2D map for the example scene shown in Fig. 4.2.

4.8 FINDING HEIGHT OF OBJECTS FOR BUILDING 3D MAPS

Evaluating height of the objects in a given image would help in 3D reconstruction. Figure 4.17 shows side view of a situation considering two objects on the ground. Let the first object's (object 1) bottom and top nearest corners projections on the original image plane be P_1 and Q_1 and let the second object's (object 2) bottom and top nearest corner projections on the original image plane be P_2 and Q_2. After the transformation of image plane to get second object's height same as first objects', maintaining common

scale S_k, distance between P_2 and Q_2 will now be P_2^1 and Q_2^1, which needs to be obtained geometrically. It has to be noted that, if the image plane is passing through say P_2 (the actual object's base corner on the ground) then the height of that object is given by p_2q_2, which will be captured as p_2q^1 in the image plane.

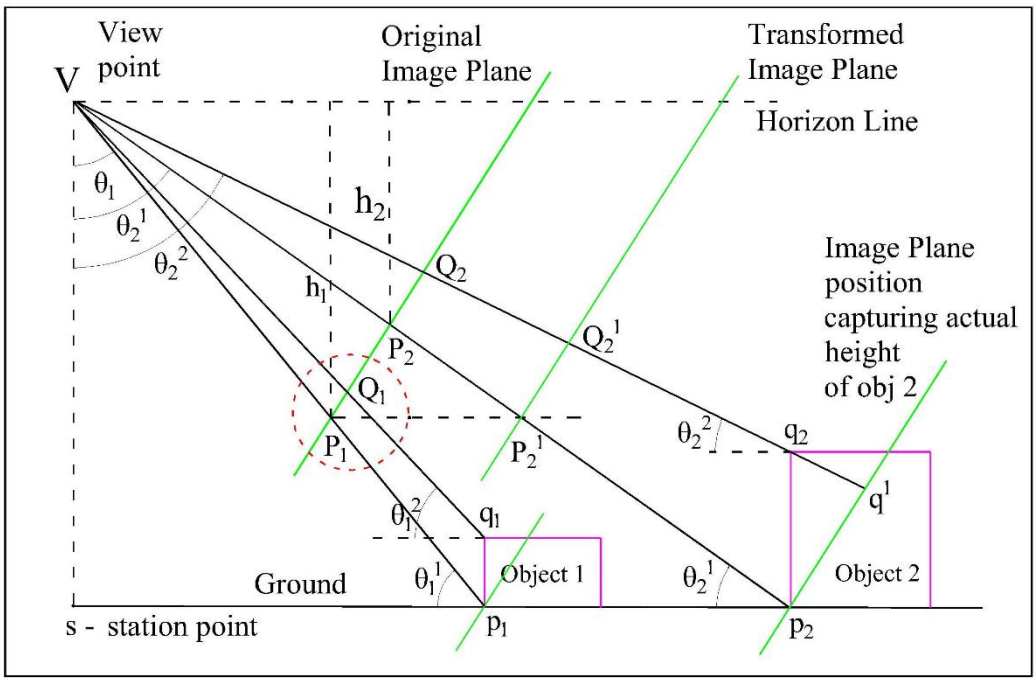


Fig. 4.17: Notion of transformed image plane for height measurement.

Let us first find the height of the reference object (object 1). Figure 4.18 shows the detailed view of the highlighted area in Fig. 17. Note that the height of the first object when the image plane passes through P_1 will be P_1Q^1 and $\theta_2 = \theta + \beta_q$, where β_q is similar to β but for the top corner Q of the object, obtained using eq. (4.14).

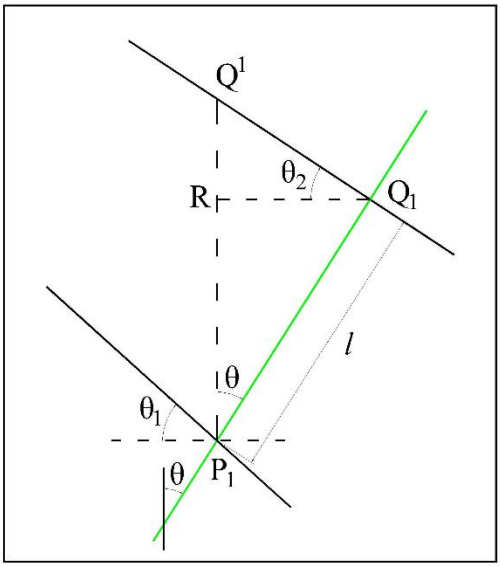


Fig. 4.18: Geometry for reference object height measurement.

From triangle P_1Q_1R in Fig. 4.18,

$$P_1R = l.cos\,\theta \tag{4.25}$$

$$Q_1R = l.sin\,\theta \tag{4.26}$$

From triangle Q^1RQ_1

$$tan\,\theta_2 = \frac{RQ^1}{Q_1R} \tag{4.27}$$

Using eq. (4.26) in eq. (4.27),

$$RQ^1 = l.sin\,\theta.tan\,\theta_2 \tag{4.28}$$

Thus, height of the reference object (object 1) can be obtained as

$$P_1Q^1 = P_1R + RQ^1 = l.cos\,\theta + l.sin\,\theta.tan\,\theta_2 \tag{4.29}$$

The height of other objects cannot be obtained in the same way as obtained for the first object. After projection of the objects (other than the first object), the relative distance between top and bottom corners will increase. This situation is shown in Fig. 4.19 for object 2, where *l* denotes the actual length of P_2Q_2 in the original image plane. But the same distance in the projected image plane is represented by $P_2^2Q_2^2$ for the case when both P_2 and Q_2 lie below *O*. P_2O_1 and Q_2O_2 are both same and equal to *OO'*, representing the perpendicular distance between the two parallel image planes, which is already calculated using eq. (4.15).

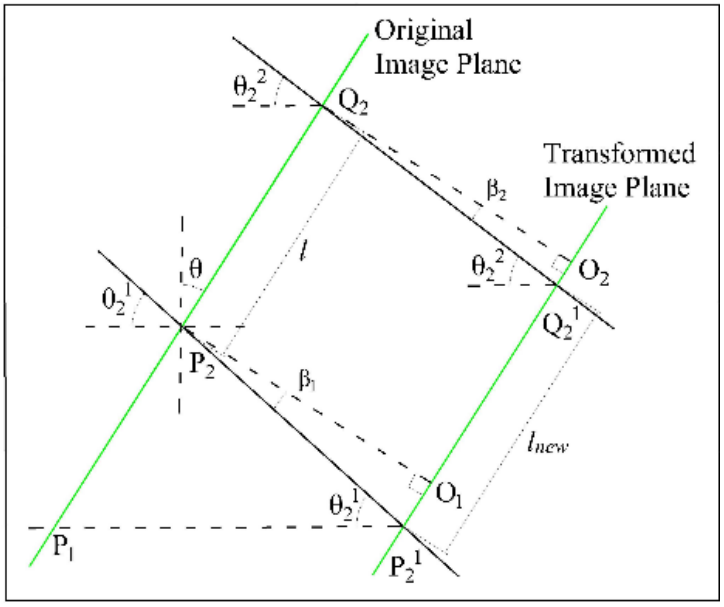


Fig. 4.19: Geometry for measuring the projected length l_{new}.

Let β_1 and β_2 are the angles made by P_2 and Q_2 respectively with the viewing axis (perpendicular to image plane) in the side view, obtained similarly using eq. (4.7). Note that $\theta_2^1 = \theta + \beta_1$ and $\theta_2^2 = \theta + \beta_2$. From Fig. 4.19, we have

$$l_{new} = P_2^1Q_2^1 = O_1O_2 + O_1P_2^1 + O_2Q_2^1$$
$$= O_1O_2 + k.tan(\beta_1) - k.tan(\beta_2) \quad (4.30)$$

Equation (4.30) gives new length of the second object between top and bottom corner projections on the new image plane. Now that the new length l_{new} is available, replacing *l* with l_{new} and plugging this in eq. (4.29) will result in computing the height of second object, to the same scale S_k as the first object. This will look like:

height of the second object $= l_{new}.cos\,\theta + l_{new}.sin\,\theta.tan\,\theta_2$ (4.31)

The procedure will be repeated if there are multiple objects. Discussion until now considered that P_2 and Q_2 are both below *O,* in the image plane. But there are two other cases: second case when both P_2 and Q_2 are above *O* in the image plane (Fig. 4.20), third case when the image centre *O* is in between P_2 and Q_2 in the image plane (Fig. 4.21). Fortunately, the same relation (4.26) applies for all the three cases, with the only difference in β values. β will be considered negative when the corresponding corner (for which we are finding β) lies above the image centre *O* and positive otherwise.

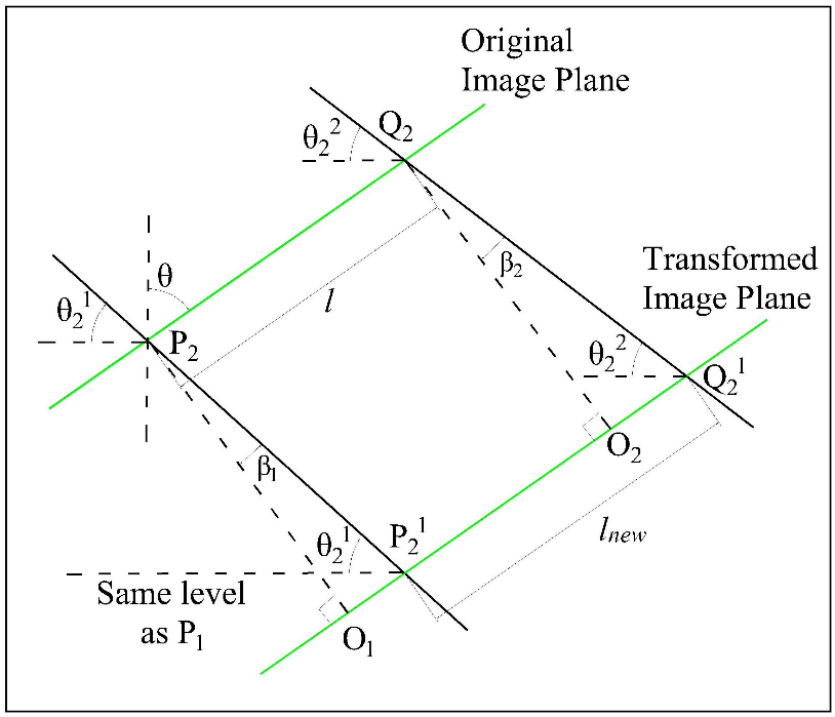


Fig. 4.20: Geometry for finding l_{new} in case 2.

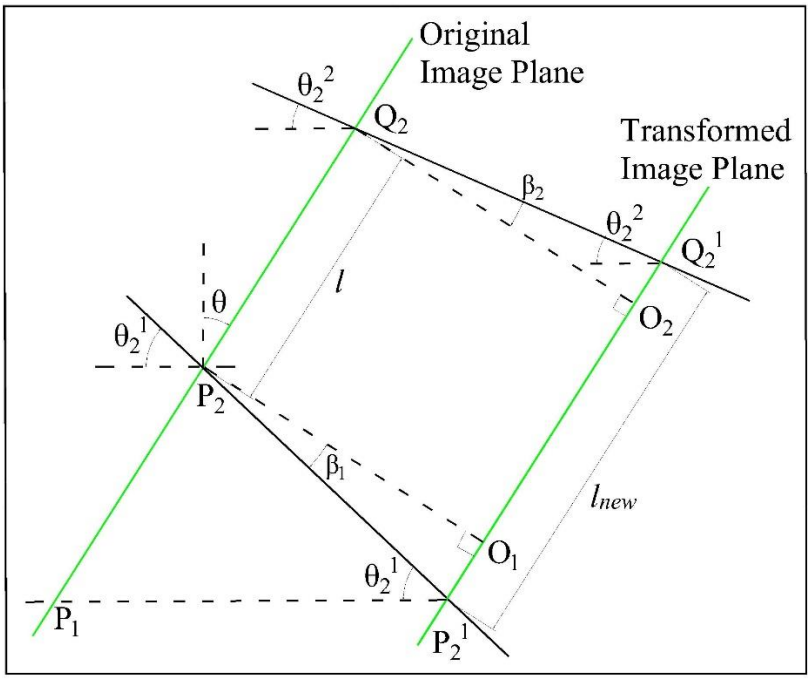


Fig. 4.21: Geometry for finding l_{new} in case 3.

Figure 4.22 shows the 3D model of the sample map constructed, taking into consideration the heights of the objects found using the relations derived here. More experimental results on 3D construction is presented in the following chapters.

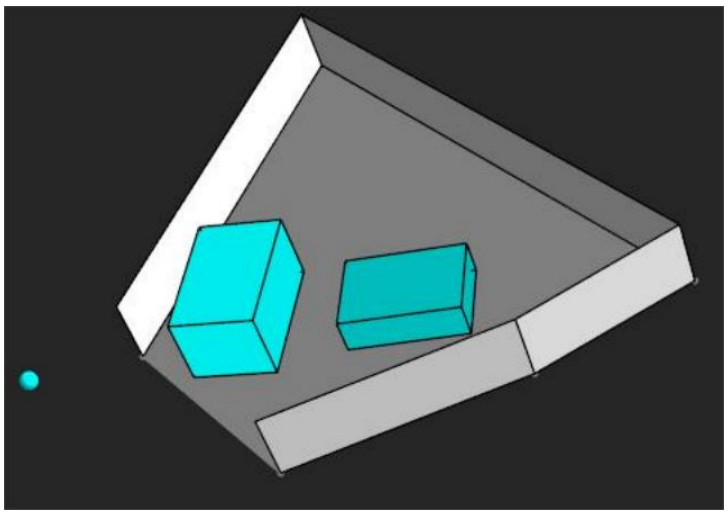


Fig. 4.22: Constructed 3D model.

4.9 THEOREM 1: 2PP is a special case of 3PP, when pitch angle is zero

Proof: 2PP is a situation when the image plane is positioned parallel to any one of the objects' surface. So, in this case, viewing direction will be horizontal in the side view. Hence, the camera pitch angle will be given by $\theta = 0$. As is clear from Fig. 4.6 when $\theta = 0$, OG will also be zero and hence $P_1G = OP_1$.

From (4.6) and using $\theta = 0$, we get

$$h_1 = P_1G.cos\,\theta = P_1G = OP_1 \tag{4.32}$$

$$tan\,\beta = \frac{OP_1}{d} \tag{4.33}$$

From (4.9), we can say

$$b_1 = \frac{h_1}{tan(\theta+\beta)} = \frac{h_1}{tan(\beta)} = \frac{h_1.OV}{OP_1} = OV \tag{4.34}$$

So, b_1, the horizontal distance of the base corner from the station point is equal to OV which is the perpendicular distance from the view point to the image centre. This distance is nothing but the perpendicular distance '*d'* which we measure from the 2PP top view construction. Let us consider an image consisting of two objects, with image centre *O* positioned in between the base corners of the two objects as shown in Fig. 4.12. Now, for the other object (second object with base corner at P_2) other than the reference object, ground distance of P_2 from the station point is taken from (4.19) as

$$b_2 = b_1 + P_1Q + QP_2' \tag{4.35}$$

Where P_1Q and QP_2' are taken from equation (4.18) and (4.13) respectively as

$$P_1Q = P_2Q\,tan\,\theta = (h_1 - h_2)tan\,\theta \tag{4.36}$$

$$QP_2' = \frac{P_2Q}{tan(\theta-\beta)} = \frac{h_1-h_2}{tan(\theta-\beta)} \tag{4.37}$$

Taking $\theta = 0$, we get

$$P_1Q = 0 \tag{4.38}$$

$$QP_2' = \frac{h_1-h_2}{tan(-\beta)} \tag{4.39}$$

Note that the second object's base corner is above the image centre as visible in the side view shown in Fig. 4.12. So, the sign of the angle β should be taken as negative. Hence (4.39) will become

$$QP_2' = \frac{h_1-h_2}{tan(\beta)} \tag{4.40}$$

$h_1 - h_2$ is the difference in heights of the two base corners which is nothing but h_d as discussed during 2PP derivations. Substituting for $tan\,\beta$ we get

$$QP_2' = \frac{h_d.d}{OP_2} = \frac{h_d.d}{h_2} \tag{4.41}$$

Substituting for P_1Q and QP_2' in (4.35), we get

$$b_2 = b_1 + 0 + \frac{h_d.d}{h_2} \tag{4.42}$$

So, the shift distance for the second object from the view point (station point in top view) should be $\frac{h_d.d}{h_2}$ more than b_1 (vertical distance of first object from view point). Note that this offset distance is same as that obtained for 2PP case given in equation (3.3), proving our claim that 2PP is a special case of 3PP by keeping the pitch angle as zero. Thus, the mathematical relations given for 3PP can be considered as a generalized one, applicable for all cases.

CHAPTER 5

MULTI-BUG PATH PLANNING (MBPP) ALGORITHM

Techniques for building maps of given scenes from single images were presented in the previous chapters. Once a map is ready, path planning can be done for a robot to navigate by avoiding the obstacles. There are several methods that addresses the problem of planning an optimal path, given a map of the environment. In this section, we introduce a new path planning algorithm, named as 'Multi-Bug Path Planning' (MBPP). MBPP has several advantages over traditional techniques which will be explained in the following sections. Input maps from the mapping algorithms will be converted to grid maps and will be fed to the algorithm which will process it and give a feasible path, avoiding obstacles.

5.1 INTRODUCTION

In the field of mobile robotics, path planning is essentially one of the core research areas. An optimal path is generated for the robot to move from the start position to goal position, taking into account many constraints such as distance travelled and time. In spite of emergence of several techniques and algorithms during the past years, two popular approaches are almost always studied in literature for path planning problem: deterministic algorithms (that uses some kind of heuristic) and sampling based algorithms (probabilistic). Most of the research techniques focus on global path planning wherein the environment is completely known to the robot. Usually, global planning for the given start and goal states are done offline i.e. before the start of the robot. There are several approaches to local path planning too, wherein planning is carried online (referred as reactive), while the robot is actually in transit.

Bug algorithms are a class of local path planning and online techniques wherein the robot uses its sensors to get information while navigating through real worlds. Once the robot reaches an obstacle, it exhibits wall-following phenomena in order to reach the destination (Ng and Braunl, 2007). Paths found in online mode are usually not optimal and are mostly sub-optimal.

The fact that the environment is known in advance to the robot motion in deterministic case helps the A* (read as A-Star) algorithm to find shortest paths through grid/graph

worlds (Hart *et al.*, 1968). Because of its optimality and simplistic approach, A* is almost always the first choice for path planning and hence it is still used extensively in robotics and by video gaming community. But the paths found by A* are unrealistic when applied to real robot navigations; because A* paths exhibits several heading changes and are constrained to move along grid edges (Daniel *et al.*, 2010). Usually, post-smoothing techniques are required to overcome unrealistic and optimal issues. But they still don't change the path topologies and results in sub-optimal paths. This is explained in Fig. 5.1 (Ferguson and Stentz, 2006), where post-smoothed A* (A* PS) finds the path along the red line while the actual shortest path is along blue dashed line.

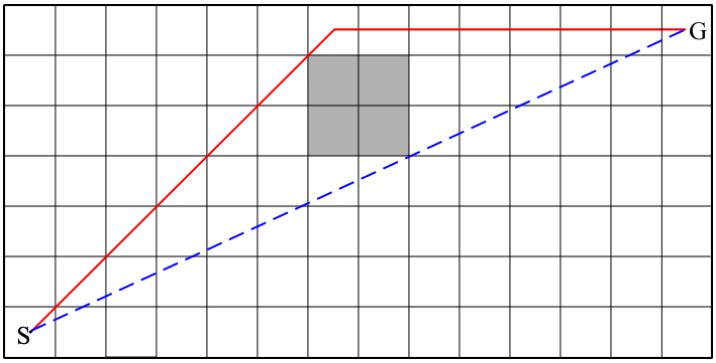


Fig. 5.1: A* Algorithm Post Smoothening.

Researchers came up with several variants of A* for real time implementations including any-angle path finding methods. D* (Stentz, 1994), incremental A* (Koenig and Likhachev, 2002(b)) and D* Lite (Stentz, 1994) are the major variants found in literature. (Ferguson and Stentz, 2006) proposed an interpolation based planning algorithm called Field D* which is an extension to D* and D* Lite algorithms, that allows path to take any angle. Currently, many mobile robots employ Field D*. In (Daniel *et al.*, 2010), authors introduced two variants of any-angle path planning algorithms called Basic Theta* and Angle-Propagation Theta*. These algorithms showed great results in terms of realistic as well as optimal path finding capabilities**.**

In-spite of the improved performances, A* variants still employ the fundamental A* concepts. They still need CLOSED and OPEN lists in order to construct a path, to avoid revisiting of states as well as to expand the next node (while planning) by sorting the list using costs and heuristics. These lists consume memory, especially when the search space is large. Pruning (reducing the size) of OPEN and CLOSED lists are always a concern in Artificial Intelligence studies. Although many researchers came up with

several A* pruning techniques, still not much work on applications of pruned A* methods for robotics has been found.

A new global path planning algorithm (MBPP) that employs approaches similar to bug algorithms is proposed here. MBPP algorithm eliminates the need for OPEN and CLOSED lists. This algorithm basically takes greedy actions in the search space. Greedy behaviour of algorithms often results in the problem of getting stuck in local optima. But the proposed method uses conditions to move from greedy to non-greedy behaviour as and when required, thereby eliminating the danger of giving local optimal results.

5.2 BUG ALGORITHMS AND RELATED WORK

In Bug algorithms, given the start and target points, the shortest straight line connecting them is referred as *m_line*. Robot starts moving along *m_line* and uses its on-board sensors to sense for any obstacles. Once it reaches an obstacle, it starts following the wall in any one direction (left or right), specific to that algorithm. Bug1 and Bug2 (Lumelsky and Stepanov, 1987) are the two basic Bug-algorithms that allow robots to follow in any one specific direction throughout their travel. But other variants of bug-algorithms for example ABUG and Bug2+ (Antich *et al.*, 2009) allows robot to change directions based on the pre-set conditions. ABUG algorithm is an extension of Bug2+ algorithm with conditions derived from bug family. In (Ng and Braunl, 2007), performance comparisons of eleven variations of Bug algorithms using EyeSim simulation platform was presented. MBPP is similar to ABUG in the sense that it also uses the concept of multi-path, but MBPP has its own conditions viz. leaving conditions, waiting conditions, multi-bug generation, corner finding and realistic implementation of the path.

Proposed work employs the online behaviour of bug algorithms to offline planning with static obstacles. This work aims to derive a method that is possibly a hybrid of both deterministic and non-deterministic methods so that the same program could be used in both cases. Additionally, MBPP employs *line-of-sight* algorithm making use of well-known Bresenham line-drawing algorithm, which was initially proposed for digital plotters (Bresenham, 1965).

5.3 NOTATIONS

S_S → start position of the robot

S_g → desired goal position of the robot

m_line → set of states representing the path taken by robot whenever there are no obstacles

O_i → obstacle *i*

v_list → list that stores identified obstacle corner states by the bug

rr_{mxn} → state values matrix, where *m* and *n* are size of the map *M*

LOS(S, S') → line of sight check between *S* and *S'*

neigb(S) → neighbours to state *S*; every *S* will have eight neighbours

select_list → list of states representing the best possible path chosen by algorithm at every episode

hit_list, *leave_list* → lists that store states whenever bug hits/leaves O_i respectively

Given *M*, the map of the environment with position of the obstacles O_i marked in it and given the start S_S and target positions S_g for the environment, the goal of the algorithm is to find a shortest feasible path p:

$$p = \{p: (p \in P)\ and\ (eucld(p) \leq eucld(p_i)) | S_s, S_g, M\}\ \forall\ i = 1,2 \dots n$$

where *n* denotes the number of feasible paths, *P* denotes the set of all paths i.e. $P = \{P_1, P_2 \dots, P_n\}$ and *eucld*(p_i) denotes sum of Euclidean distance between successive connecting states for the path p_i.

5.4 MULTI-BUG PATH PLANNING (MBPP) ALGORITHM

MBPP algorithm starts by initializing the start and goal states for the given environment and assigns state values to each state using distance transform method. S_g is assigned a larger rr_g value and all other remaining states are assigned lesser values according to their distance from S_g i.e. $rr_{i,j} = rr_g - dist(S_g, S_{i,j})$ for $(i, j)^{th}$ state. The program then sets *m_line* which is a list of states visited by navigating in the environment without any obstacles. Once the *m_line* is set, the actual program starts. A bug moves along the *m_line* by taking action at each step and moving to any one of the neighbouring states

with a branching factor of eight. Greedy actions are taken at each step by choosing the state with largest *rr* value, until the bug meets an obstacle. If no obstacle is found and the bug reaches the goal position, then the algorithm will smooth the path using *LOS* algorithm and reports the smoothed-path.

During the bug travel, if it hits an obstacle, then the current bug generates a new bug and that state is added to *hit_list* for both the bugs. Both the bugs now adopt wall-following phenomena such that the parent bug takes its usual greedy action, while the newly generated bug instead of taking greedy action chooses the state that is on the other side of its parent bugs state at the instant of hit. After this hit step, the generated bug also chooses greedy actions. Also, the bugs are avoided from taking actions that result in states - already visited by that bug, by maintaining a list of two recent states. This procedure allows the bugs to move in either direction of the wall so that they can exploit possible paths around the obstacle, thus avoiding the danger of getting trapped in local optima as well as reducing time for bug travel.

At each step during the wall travel, *LOS* check is done to see if the new state is visible from the start state. Visibility between two states means that the straight line path between the two states is obstacle free. If there is no visibility between the current and start states, then the algorithm marks the common neighbour for both previous and current states in *v_list* for that bug such that the noted state is visible from the start state as well as visible to the current state. The bug continues moving along the wall and now the *LOS* check is done between the current state and the recently added state in *v_list* for that specific bug and noting corner states in *v_list* continues if there is no line of sight.

The bug drops the wall following behaviour if it meets *m_line* along the wall. At this instant, it notes the state detail in its *leave_list* and now starts moving along *m_line* towards the goal. If during its travel, it again hits an obstacle, then bug generation continues and wall following phenomena repeats. If in case any bug ends in a wall corner with no direct path towards the goal, then that bug is terminated. If the bug meets its own latest hit or leave states as noted in their respective lists, then the bug is looping around the obstacle and is useless. So MBPP algorithm terminates this bug. Also, any bug that ends in a state which is noted in the *leave_list* of other bugs will wait without any further iteration. Since further movement of this bug result in moving along the

same states as taken by some other bug (that has already travelled this path) which will ultimately end up in noting same corners, hit points as well as leave points if any. Keeping this bug idle till other bugs are done saves computational time as well as memory. Later, after all bugs complete their travel, additional states are added to this bugs *v_list* making use of the *leave_list*. The latest state in *leave_list* of this bug is taken and a check is done among all other bugs *leave_list* to see if this state is available. If that state is found in say bug-k, then all the corner states that are after this leave state will be added directly to the idle bug to make it complete.

The whole process continues and the algorithm runs until all the bugs either terminates or waits or reaches the target state and not a single bug needs iteration. Once all the bugs are done with their travel and if it is found that no bug reaches goal even after completion of the process, then it literally mean that there are no feasible paths between the given states. MBPP algorithm reports this and terminates. If goal is reached by at least one bug, then MBPP algorithm adds additional states to the waiting bugs, as discussed previously, and evaluates Euclidean distance between each states of the *v_list* for all the selected bugs. Then, the one resulting in least distance (call it *select_list*) is chosen for further check.

A line of sight check is performed for successive states of *select_list* and if it is clear, the algorithm terminates outputting the *select_list*. Otherwise, if the path is not clear and *LOS* algorithm senses any occupancy of the states during the check, then the algorithm re-plans the path by starting the loop again with the state just before the first identified occupied state as the start state. Now, bug generation, corner identification, leaving and termination conditions are carried as usual, until the newly generated bugs are visible to the next state in the *select_list.* The *v_list* of all the earlier bugs are modified with the introduction of newly identified corner states. Then the process of Euclidean distance evaluation, line of sight and feasibility check are repeated until shortest feasible path is found. As can be seen, MBPP algorithm finds all feasible paths and employees re-planning strategy if in case no feasible or shortest path is identified. Thus the proposed algorithm is good enough to be considered for its feasibility for online case too.

Figure 5.2 shows a case study with start, goal and obstacles indicated. For simplicity, illustration, and easy understanding, path was shown in single line and the robot was

shown as point size. In actual implementation, the robot dimensions was considered as square cell size and the planning was done such that it finds path for the cell as a whole. As can be seen, the bug $\boldsymbol{b_1}$ starts from *S*, moves along *m_line* and hits the obstacle at H_1. Now $\boldsymbol{b_2}$ is generated and moves downwards while $\boldsymbol{b_1}$ continues moving in upward direction, both along the walls. After bug $\boldsymbol{b_1}$ crosses S_3, it loses line of sight to *S*. So the corner state S_3 is noted to *v_list* of bug $\boldsymbol{b_1}$. Bug $\boldsymbol{b_1}$ moves further and now the line of sight check is done between S_3 and the current state. Since there is no visibility from S_3 to states after S_3, the corner state S_4 gets noted in $\boldsymbol{b_1}$ bugs *v_list*. This continues and once it reaches the *m_line* at L_1, $\boldsymbol{b_1}$ becomes idle and waits. The reason is because bug $\boldsymbol{b_2}$ has already reached this state and noted L_1 in its *leave_list*. So $\boldsymbol{b_1}$ waits till other bugs ($\boldsymbol{b_2}$ in this case) either reaches *G* or terminates. Bug $\boldsymbol{b_2}$ reaches the goal and the *v_list* state after L_1 for $\boldsymbol{b_2}$ i.e *G* is hence added to $\boldsymbol{b_1}$'s *v_list*, thus making it complete. Figure 5.3 shows the result of line of sight check between successive states in *v_list* for both bugs. MBPP terminates by outputting the shortest path identified (*S*, S_1, S_2, *G*).

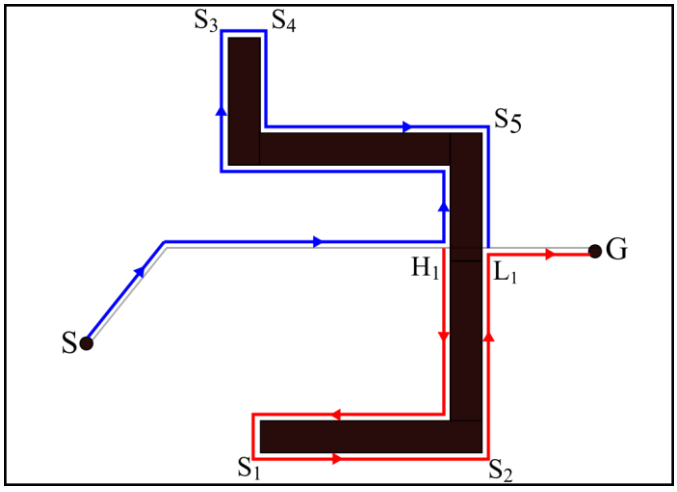


Fig. 5.2: MBPP implementation.

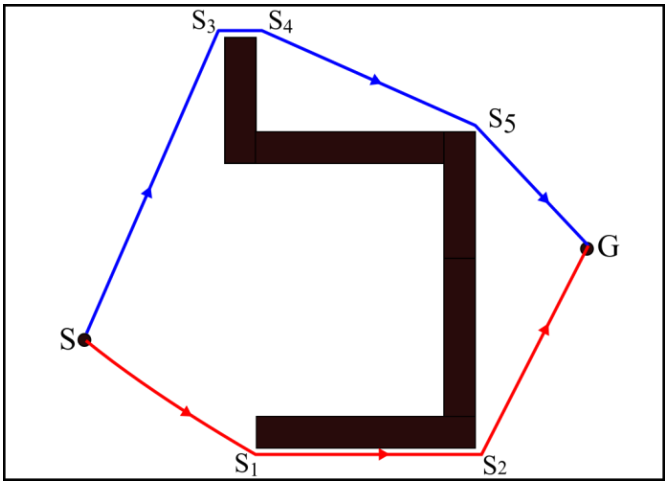


Fig. 5.3: Paths found by MBPP.

Figure 5.4 shows one more example which intuitively explains how search in MBPP algorithm is propagated, by way of multiple bug generation. After making a loop, bug ***b₁*** (path shown as blue line) meets its own hit point at H_1 and hence is terminated. Bug ***b₂*** (path shown as dashed line in red) meets the hit point H_2 that has already been reached by ***b₁*** and hence it waits. Bug ***b₃*** (path shown as green) actually reaches the goal, which is generated after ***b₁*** hits at H_2.

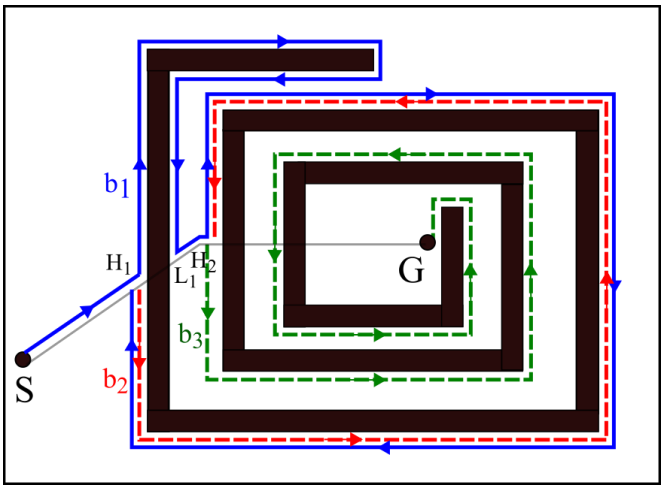


Fig. 5.4: Another example for MBPP implementation.

5.5 FLOW CHART

Figure 5.5 shows an example environment with two obstacles in-between the start and the goal state. Three bugs will be generated as the algorithm propagates. Neglecting the last part of finding the optimal path, the flow chart in Fig. 5.6 illustrates the step by step process on how the bugs are generated and propagated. The algorithm runs until the third bug (b3) reaches the goal state, while the other two bugs (b1 and b2) are made to wait.

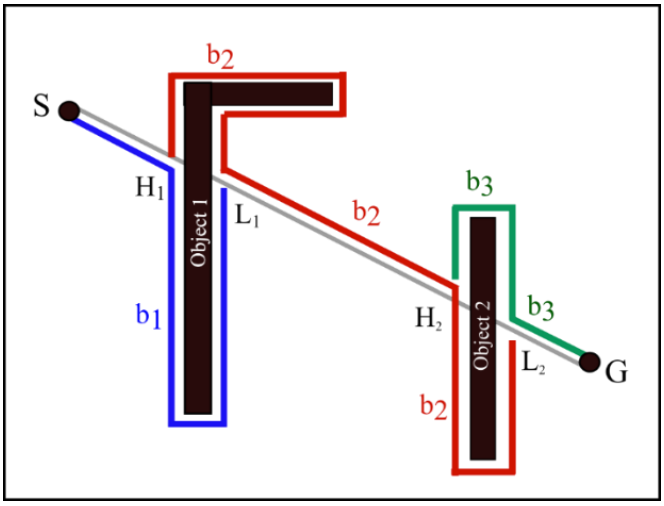


Fig. 5.5: An example environment with two obstacles.

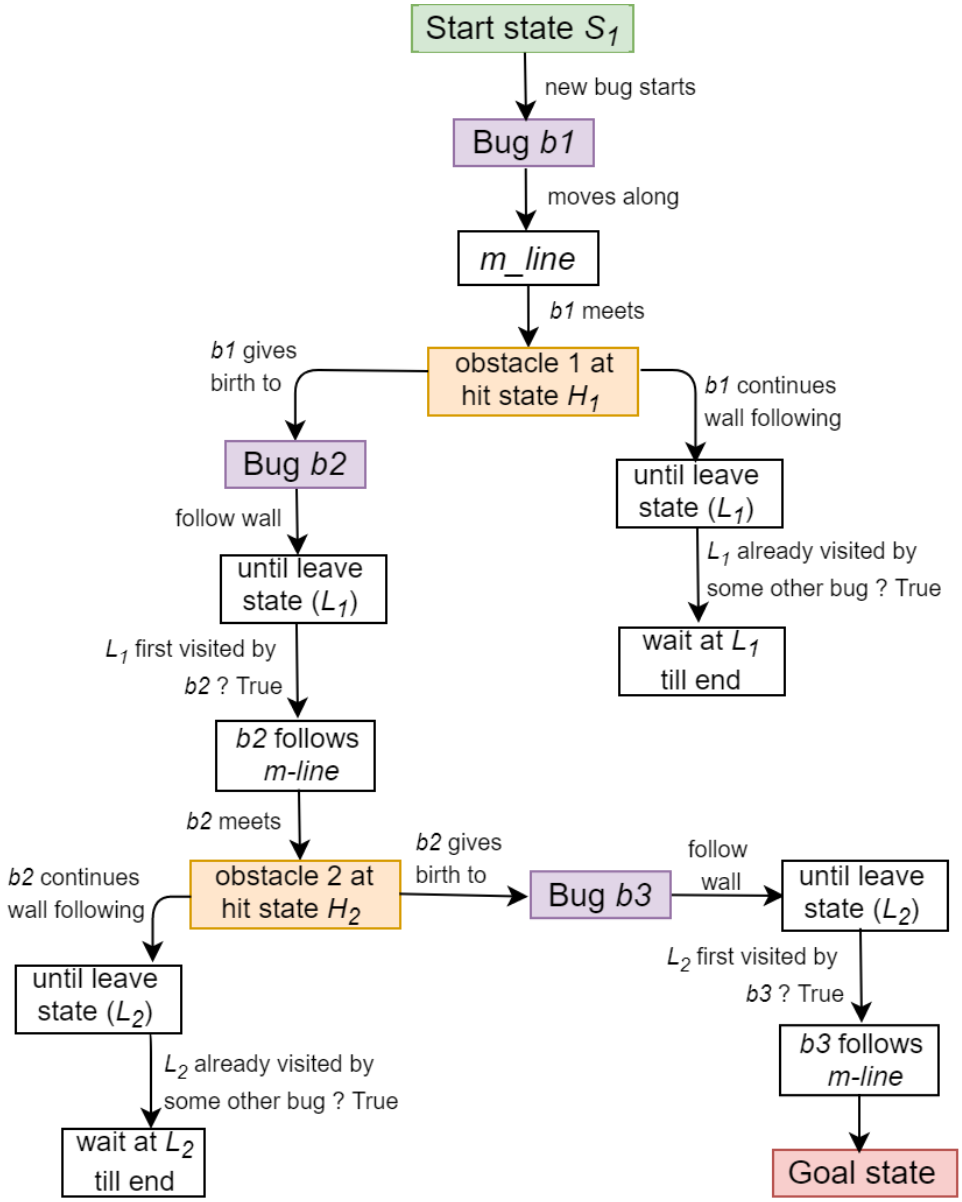


Fig. 5.6: Flow chart depicting the structure of MBPP.

5.6 MBPP CONDITIONS

Following are the set of important conditions that are applicable for MBPP algorithm.

1. **Condition C_1 : Termination conditions**
   - **C_1(a):** If a bug is following an obstacle O_i and if it meets the boundary of the map M such that any action taken is unsuccessful and will not result into any state, as the bug is not allowed to take previously visited state and it is not allowed to select a state away from the obstacle O_i wall, then that means the bug has reached a wall and there is no direct path. So, the bug should be terminated.

- ***C_1*(b):** If a bug while following an obstacle O_i lands in a state $S = L_k$ that is listed as the latest (the most recently added) leave state in the bugs *leave_list*, i.e. *leave_list* = $[L_1, L_2, \dots, L_K]$, then that bug has completed a loop without finding goal state. This bug is not helpful and hence has to be terminated.
- ***C_1*(c):** If while following an obstacle O_i, a bug ends up in a state S that is already in the hit list of any of the bugs i.e. $S \in anyof(hit\_list)_j \; \forall$ j = 1...n, where n is the number of bugs, then that bug can be terminated.

2. ***C_2*** : **Leaving conditions**
   - ***C_2*(a):** If a bug while following an obstacle O_i lands in a state S that is in *m_line*, then leave the wall following behaviour and take greedy steps thereafter.
   - ***C_2*(b):** If *repetition* = *True*, do line of sight check between current state and next state in the *v_list* of the bug, if visible then leave wall following behaviour.
3. ***C_3*** : **Waiting condition**
   - If a bug while following an obstacle O_i moves to a state L_k such that condition ***C_1*(b)** is False and $L_k \in anyof(hit\_list)_j \; \forall$ j = 1...n, where n is the number of bugs, then make the bug wait till other bugs end their iteration.
4. ***C_4*** : **Action selection conditions**
   - MBPP assumes MDP (Markov Decision Process) properties; since the next state S_{new} – to be chosen from a given state S is not dependent on the history of states $\{S_g, S_1, \dots\dots\}$. If the bug is not following any wall or hasn't hit any obstacle O_i yet, and is moving in *m_line*, then, action selection will be greedy, given by $a = argmax_{neighb(S)}[rr]$ i.e. bug moves to the neighbouring state which has the largest *rr* value. But, action selection for the bug which has just taken birth from its parent bug, (because of hitting an obstacle) is not greedy at this instant and the new bug will take a state on the other side of its parent bug - no matter what this state's *rr* value is. At every action selection step, bug is not allowed to visit from where it came from, by having a note of its previous two visited states. If the bug is following an obstacle and it has already taken at least one step along the wall, then action selection is again greedy as given by $a = argmax_{neighb(S)}[rr]$.

5.7 PROPERTIES OF MBPP

Following are the properties exhibited by MBPP with formal proofs.

Lemma 1: ***MBPP is correct***

Proof: MBPP is a planning and re-planning algorithm. MBPP initially finds set of states representing the corners of the obstacles, including the start and goal states. After finding the desired set of states, MBPP employs path smoothing and line of sight techniques to check if the path is feasible or not. By definition of feasibility we mean navigable and obstacle free. If the path is clear, then MBPP terminates outputting the found path. But if it finds any occupancy during the path, then re-planning part of MBPP starts and the algorithm propagates the search along the boundary of the newly identified obstacles, in order to find the feasibility and corner states of that obstacle. Once the corners are found, then again MBPP performs check for optimality as well as feasibility until it finds a path that is shorter and unblocked (navigable). This proves the correctness of MBPP algorithm.

Lemma 2: ***MBPP is complete***

Proof: MBPP propagates the search in the direction of target and in case of obstacle collision; it splits the search and exploits in both possible directions thus guaranteeing the path finding behaviour if a path exists. If P is empty i.e. the set of all possible paths to the goal is NIL, then all the bugs in MBPP runs over walls and boundaries and hence gets terminated, reporting no path status. Thus MBPP is complete and terminates in all cases. In fact, MBPP finds shortest possible realistic path for a given environment with or without obstacles, rather than restricting the path to grid edges as is the usual case of A*.

Lemma 3: ***No two bugs travel the same path in the same direction***

Proof: Bugs after leaving the hit state move in either direction and hence there is a possibility that they meet in opposite direction, but they can never move in the same direction. For the bug to move in the same direction, it must have the same leave state L as contained by some other bug. But according to condition C_3, when the bug meets the state already left by some other bug, then that bug is made idle. Hence, no two bugs can move in the same direction.

Lemma 4: ***MBPP does not end in infinite looping and infinite bug generation***

Proof: Generation of bugs and looping occurs whenever a bug meets any of the already hit states and is allowed to proceed. Since MBPP uses condition C_1(c), bugs are terminated whenever they meet already hit state and so there is no possibility of repeatedly visiting already visited hit states. Also, infinite bug generation occurs when we allow the bug to proceed further, even when the bug while its travel meets its own recent leave state as noted in respective *leave_list*. The reason is obvious. Any leave state is a state in *m_line* and so when the bug which has the root from say L_k revisits L_k, then it assumes the same path, hit the wall and generate one more bug unnecessarily making an infinite loop. But MBPP which utilizes condition C_1(b) terminates this bug and hence eliminate the danger of infinite bug generation.

5.8 MBPP PSEUDOCODE

Algorithm 2 given in Appendix A.2 shows pseudocode for the main MBPP program which uses two functions viz. line of sight and path smoothing. Pseudocode for line of sight check (denoted as *LOS()*) is given in Algorithm 4, while pseudocode for the path smoothing part of MBPP (denoted as *Pat_smooth()*) is given under Algorithm 5.

It has to be noted that MBPP algorithm plans a realistic path by checking line of sight between the given two states using two straight lines that join the two direct visible corners of the state cells such that the two lines are the farthest possible lines from the center of travel. For both the set of corners, line of sight check will be performed. Simulation results for MBPP is provided in the following chapter.

CHAPTER 6

EXPERIMENTAL RESULTS

This chapter discusses the results obtained using the proposed methods for map building and path planning. First, the map building results for several images in 2PP followed by 3PP will be presented. Then, the results of MBPP algorithm implemented on the deduced maps of the example single images will be dealt. Six error metrics have been considered in evaluating the efficiency of the map building method and the same have been discussed through error box plots. For MBPP, the simulation results of the algorithm were compared with the simulation results of standard A* algorithm.

6.1 MAP BUILDING EXPERIMENTAL RESULTS

As part of experiments, several images consisting mostly of real world objects like boxes, chairs, tables etc. were captured using ordinary monocular camera. The algorithm was also tested on images with some of the objects being partially occluded. Assuming that bounding boxes can be inscribed around all the objects, box shaped objects were used in most of the captured scenes to assist in fast and easy recognition of the corners, edges, and also vanishing points. Nevertheless, we also present results without considering bounding boxes, for both 2PP and 3PP image examples. Figure 6.1 shows one such experimental environment depicting a scene in 2PP.

Fig. 6.1: An example image in 2PP.

In this thesis work, we considered that the object shapes were given in priori for the sake of evaluation of the proposed methods. This is accounting to the fact that several computer vision algorithms are available for finding edges, separating floor and objects, and also for finding vanishing points. Experiments to evaluate the influence of skew and distortion (of the camera) on floor map construction were not done. That's because these parameters in most of today's cameras (especially mobile phones) are negligible. Nonetheless, the experiential images were taken using different cameras to evaluate the errors in map building. Regardless of the camera chosen and despite the negligence of the two properties, the results have proven to be quite satisfactory.

Note that all the length measurements were in centimetres in this thesis. Results for 2PP and 3PP cases were dealt separately with all the necessary plots. Proposed method was implemented using *Mathematica*™. A generalized pseudocode is provided in Appendix A.1 under Algorithm 1, which is applicable for both 2PP and 3PP cases, considering camera roll (θ_{roll}) also. To validate the method, the generated maps were compared with their corresponding ground truths. Ground truth (measurements) includes:

- Height of the camera position and its tilt (pitch and roll).
- Dimensions of objects – *l,b,h* if rectangle parallelepiped, or bounding size of objects, including its height, if shape of the objects are anything else other than rectangle parallelepiped.
- Distance of the objects' nearest base corner to the camera centre projection on the ground, measured in x and y direction of the assumed world coordinate system.
- Angle made by the edges of the bounded objects with the x-axis of the world coordinate system.

To check the error bounds, a scale factor was estimated manually for each experimental image such that the object measurements (with their relative positions) taken physically at the site will be scaled up/down to the constructed map scale. Scale factor S_k estimation is given in eq. (6.1), where, L_1 and B_1 are the dimensions (length, breadth) of the reference object as measured from experiments (model dimension value). Let L_2, B_2 are the actual dimensions of the reference object.

$$S_k = \frac{1}{2}\left(\frac{L_2}{L_1} + \frac{B_2}{B_1}\right) \tag{6.1}$$

Station point has been considered as the origin of a reference coordinate system while building comparison floor maps. For quantifying the errors, actual world dimensions have to be scaled up/down to that of the modelled world dimensions. Note that the scaling factor S_k is decided using the reference object (first object) only, regardless of other object dimensions. Based on the reference S_k, all other objects will be brought to the same factor.

6.1.1 Error Metrics

Figure 6.2 shows the notations used to evaluate the errors in order to validate the proposed method. Six error metrics were considered to strengthen the comparisons with the ground truths. They are listed below.

1. Percentage error in centroid shift = $\left(\frac{d_{cm}}{d_a}\right) * 100\%$
2. Percentage error between distance of centroid from camera centre/view-point of the actual scaled object and centroid distance of the modelled object = $\left(\frac{d_m - d_a}{d_a}\right) * 100\%$
3. Percentage of area coincidence/overlap of modelled object with area of the actual object scaled = $\left(\frac{A_{Cm}}{A_a}\right) * 100\%$
4. Percentage error between area of actual scaled object and area of object from model = $\left(\frac{A_m - A_a}{A_a}\right) * 100\%$
5. Error in angular pose of centroid w.r.t. horizontal = $(\theta_m - \theta_a)$ in degrees
6. Error in skew angle (angle of tilt of object) = $(\beta_m - \beta_a)$ in degrees

where,

$d_m =$ Euclidian distance of centroid of the modelled object from view point

$d_a =$ Euclidian distance of centroid of the actual object from the view point, positioned and dimensioned by scaling factor S_k

$d_{cm} =$ Euclidian distance between the centroids of the actual object (C_a) and the modelled object (C_m)

$A_m =$ Area of the object from the model (given by $L_m . B_m$)

$A_a =$ Actual Area of the object found using ground truths, scaled by S_k (given by $L_a . B_a$)

$A_{cm} =$ Area of the coincident zone between the modelled object and the actual object, positioned w.r.t. view point

θ_m = Orientation of the centroid of the modelled object (angular pose) w.r.t horizontal

θ_a = Orientation of the centroid of the actual object (angular pose) w.r.t horizontal

β_m = Orientation of the modelled object's side (skewness) w.r.t horizontal

β_a = Orientation of the actual object's side (skewness) w.r.t horizontal

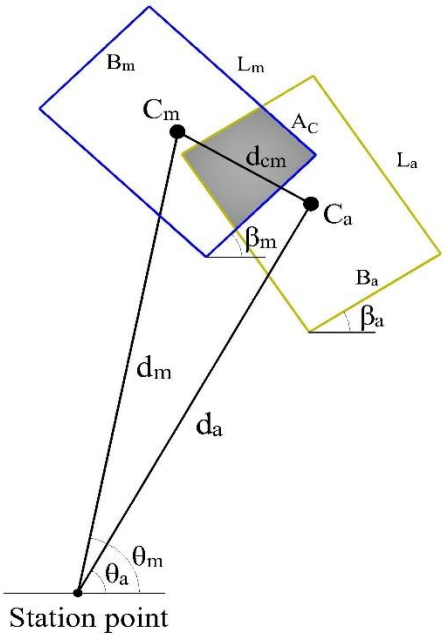


Fig. 6.2: Notations used for finding errors.

6.1.2 Results for 2PP Images

Experiments have been carried out with the help of several images in 2PP. Results of the 2D and 3D map construction for some of the images are shown in Figures 6.3−6.7. In all the images (Figures 6.3−6.7), the actual captured scene is shown in (a), the constructed 2D free space map for the given scene is shown in (b) where ground truth objects (yellow coloured rectangles) were overlaid on top of the modelled objects (blue coloured polygons) and (c) represents the 3D representation of the given scene depicting the view of the scene, as seen from the camera (the camera centre being shown as yellow coloured blob). Another 3D view of the reconstruction is shown in (d), for better visualization of how the objects are lying relative to one another and also w.r.t. the cyan coloured camera centre.

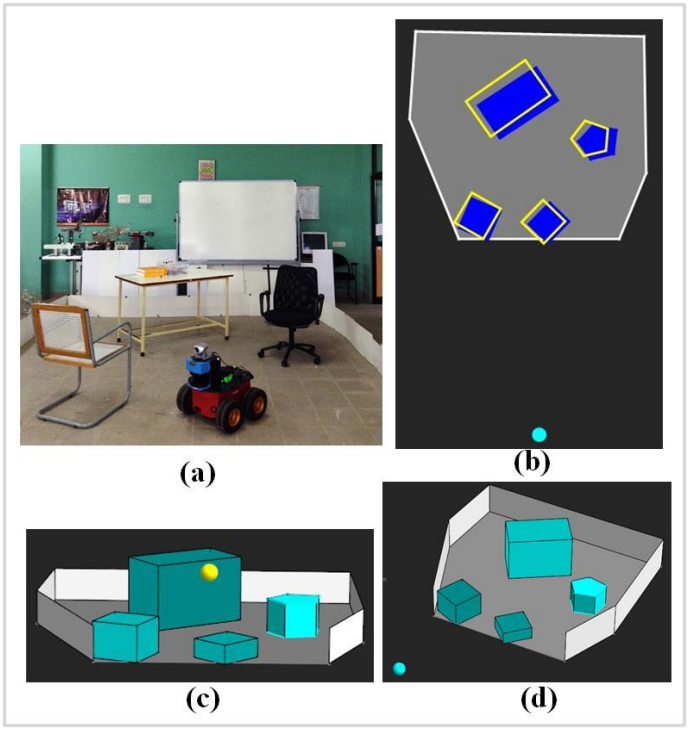


Fig. 6.3: 2PP Experimental result – 1. (a) Actual image in 2PP consisting of a robot, chairs and a table. (b) Constructed Floor map (2D map). Our method was able to construct other shapes (like the pentagonal chair) apart from the regular cuboid construction. (c) 3D reconstruction (d) Another clear 3D view

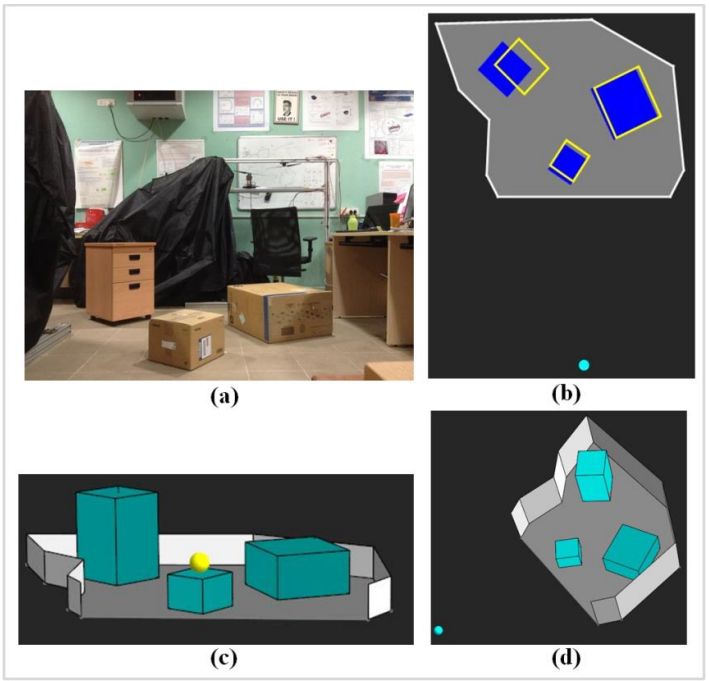


Fig. 6.4: 2PP Experimental result – 2. (a) Actual image in 2PP (b) Constructed floor map (c) 3D reconstruction (d) Another clear 3D view

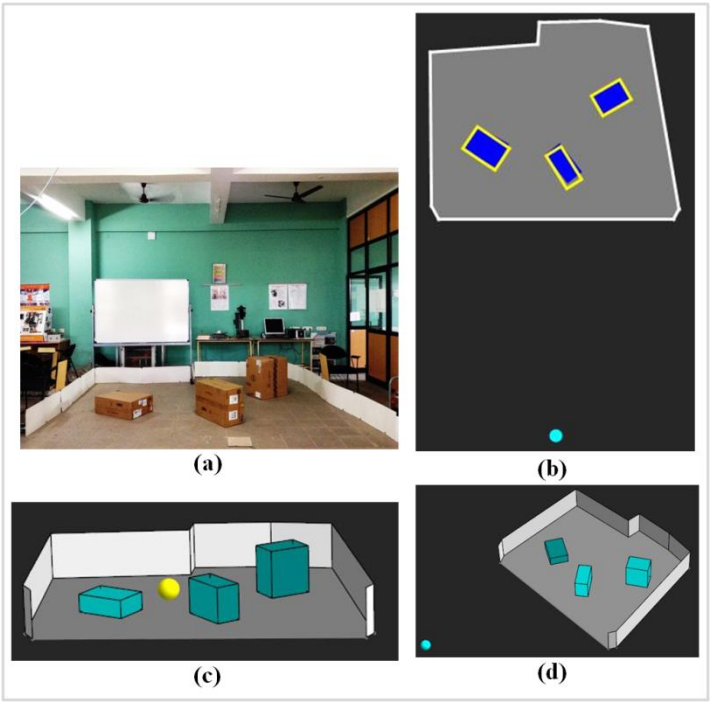


Fig. 6.5: 2PP Experimental result – 3. (a) Actual image in 2PP (b) Constructed floor map (c) 3D reconstruction (d) Another clear 3D view

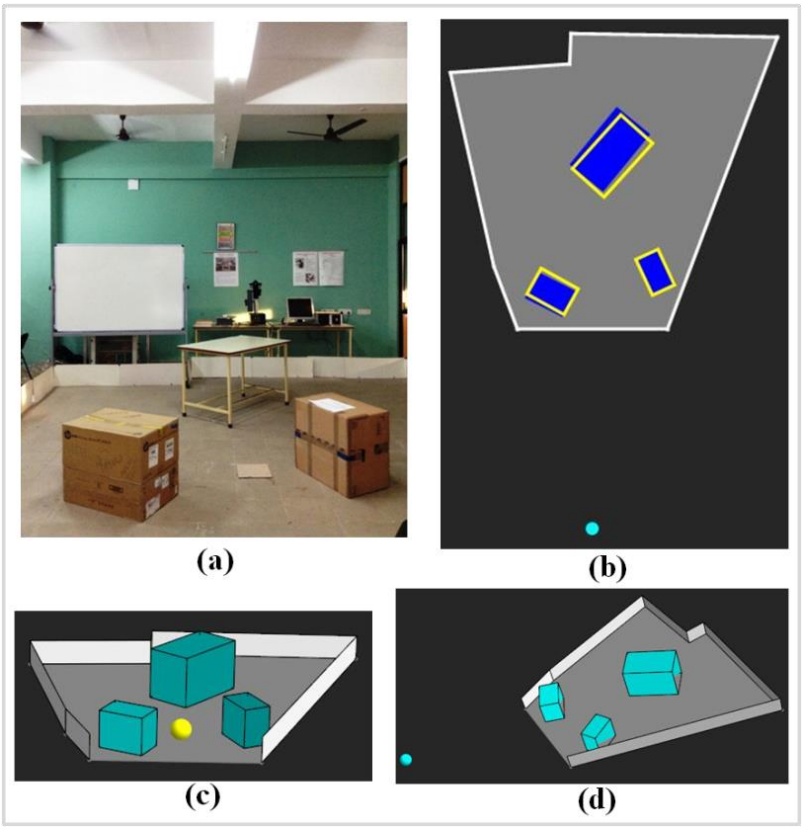


Fig. 6.6: 2PP Experimental result – 4. (a) Actual image in 2PP (b) Constructed floor map (c) 3D reconstruction (d) Another clear 3D view

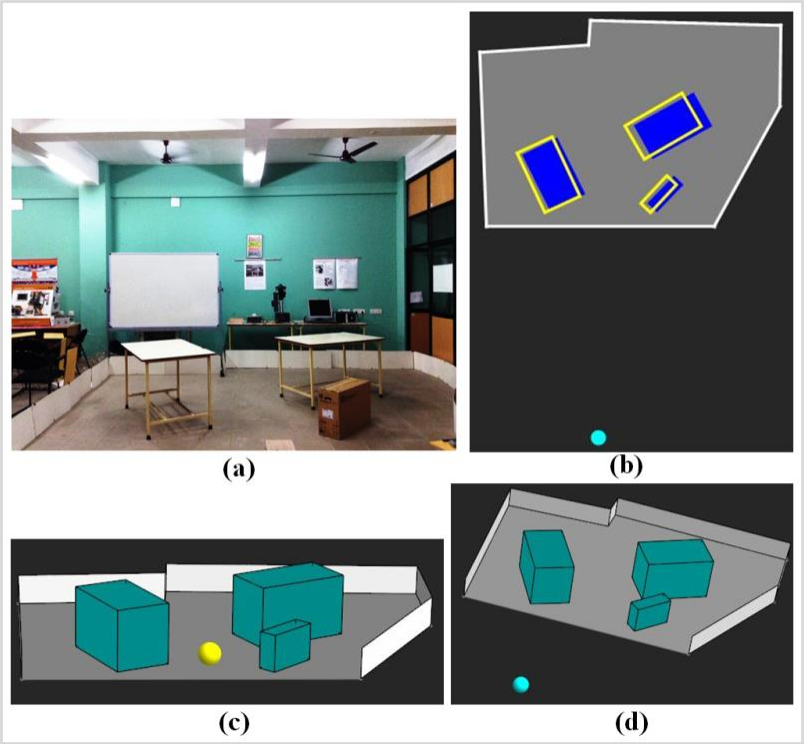


Fig. 6.7: 2PP Experimental result – 5. (a) Actual image in 2PP with one partially occluded table in the scene. (b) Constructed floor map (c) 3D reconstruction (d) Another clear 3D view

Following set of graphs (Figures 6.8−6.13) illustrates the error results for 2PP images, considering nine example image set. All the error plots are done in the form of box plots to visualize the error ranges for each example image. Note that in each experiment there are multiple objects. Figure 6.8 shows the boxplot of centroid shift error (percentage shift of centroid) for each experiment considering several objects.

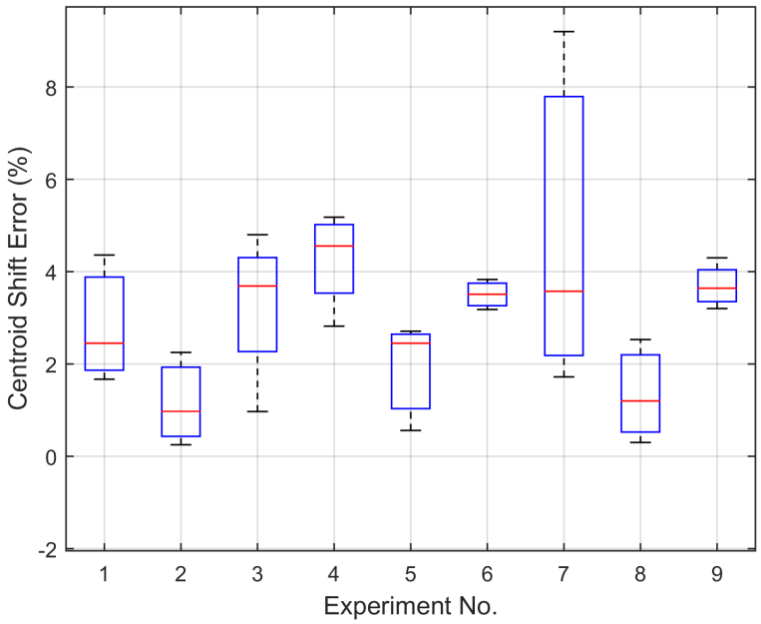


Fig. 6.8: Box plot of percentage deviation of centroids.

For example, experiment 1 consists of three objects. Worst case error in centroid shift of the modelled object compared with the actual centroid position of the scaled object is within ±9.5%. Figure 6.9 shows the boxplot results showing the error ranges of distance to centroid of the modelled object from the view point in comparison with distance to centroid of the actual ground truth object positons, scaled by S_k. Boxplot results shows that the errors are within the range of ±5%.

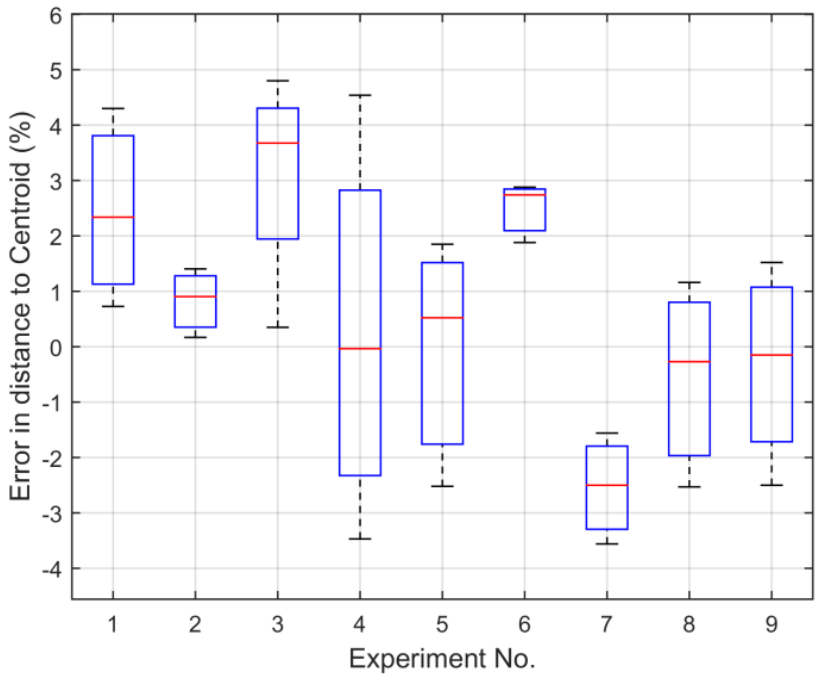


Fig. 6.9: Box plot of percentage reductions in distances to centroids.

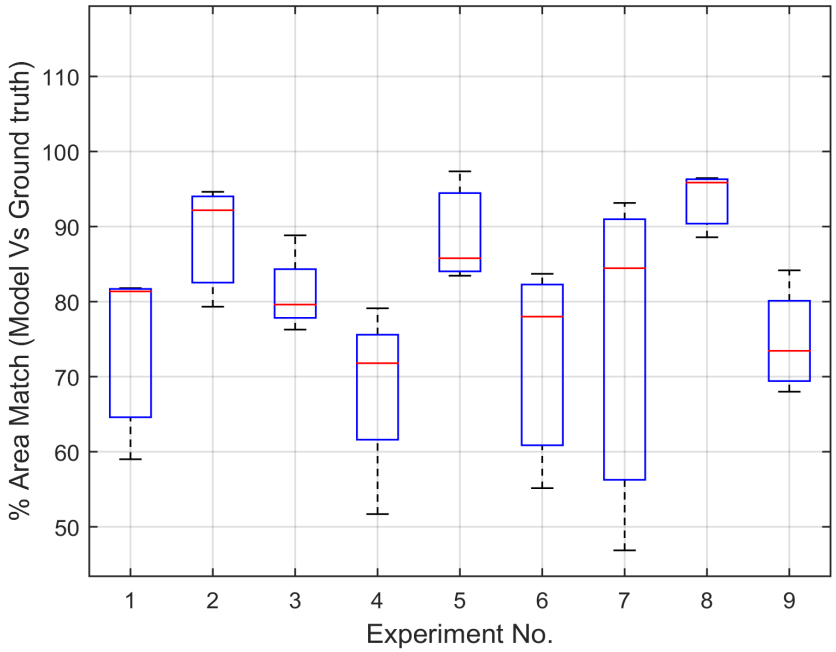


Fig. 6.10: Box plot of percentage area of overlap.

Results of the boxplot shown in Fig. 6.10 is very interesting. It shows the percentage area overlap between the modelled objects and the actual ground truths. In most cases, the worst case mismatch was around 55% while the best case match showing the correct overlap was around 95%, which is very good approximation from the model.

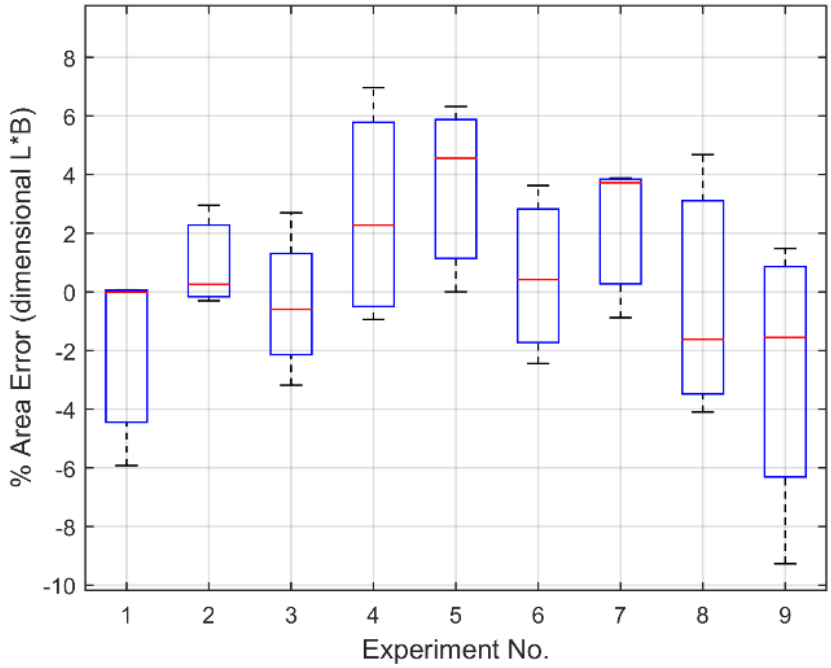


Fig. 6.11: Box plot of percentage error in area measurement.

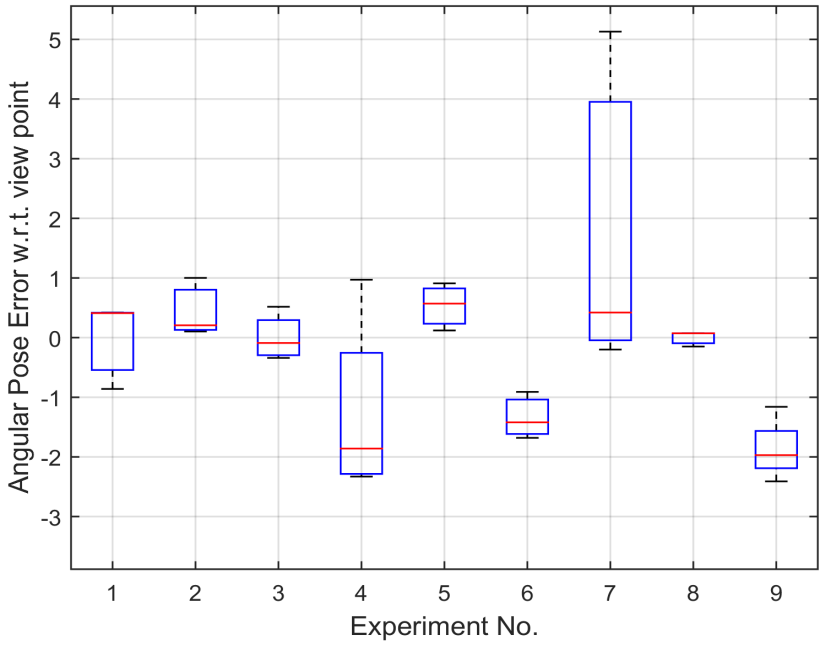


Fig. 6.12: Box plot of percentage error in angular pose of the centroids.

Figures 6.11−6.13 show the boxplots for percentage error in numerical area measurement (considering dimensions), percentage error in angular pose of the

centroids relative to world coordinate system and finally percentage error in skew angles (orientation between actual and modelled object angle w.r.t. reference coordinate system). Their error ranges are within ±9%, ±6% and ±20% respectively.

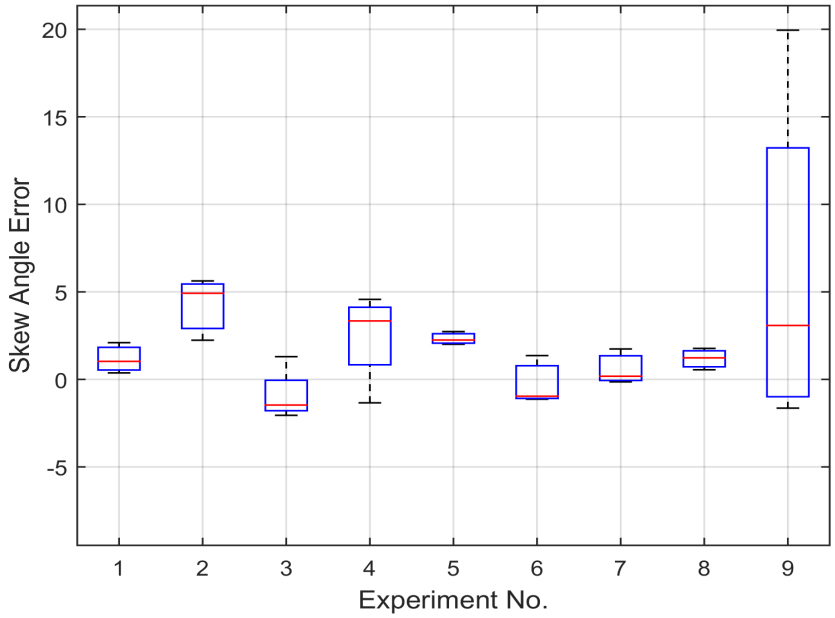


Fig. 6.13: Box plot of percentage error in skew angles.

It's worth mentioning here that the differences in measurements arise because of several factors. The possible sources of error are:

1. Human errors occurred while taking physical measurements of the objects and their distances using error-prone (erroneous) tools.
2. It's never accurate to measure the exact position of the camera centre and so there might be errors in height and distance measurements relative to the camera/view point.
3. During experiments and as time progresses there are chances that the camera got tilted from its previous orientation which would have resulted in some pitch angle and also in change of height.
4. Similarly, vanishing point detection is quite challenging owing to the chances of inaccurate edge detection. Even a small variation in vanishing point positions would result in drastic errors in ground height, object dimensions, angles (θ_1, θ_2 and θ_3) and also in distance d_p measuring the distance of the new image plane from the original image plane. This might be the first reason because of which percentage errors for some of the object's parameters reached around 5%, especially for ground height and d estimations.

5. Distortion due to cone of view of the camera. From the experimental results, it was observed that the farther the objects are (along the boundaries of the imaginary camera cone), more was the distortion.

6.1.3 Results for 3PP Images

Just like in 2PP case, as part of 3PP experiments also, several scenarios with different objects had been considered. Some of the images included unclear-edged tables as shown in Fig. 6.14. The Camera was given a pitch angle of up to 45 degrees (since most of the CCD camera won't have much tilt) and the height of the camera was varied from 0.2 meters to 2 meters from the ground level. In other words, experiments had been carried out on real world room-sized scenarios as well as on small sized table-top setups. In all the cases, results turned out to be interesting, with minimal distortion errors. Similar to 2PP case, non-parallelepiped bounding objects in one of the example images for 3PP were considered. Interesting part of the experiments included capturing images with partially occluded objects - with non-clear view of edges and also corners. In all the experiments, accurate results with acceptable error ranges were observed. Figures 6.14 to 6.17 show some of the experimental results.

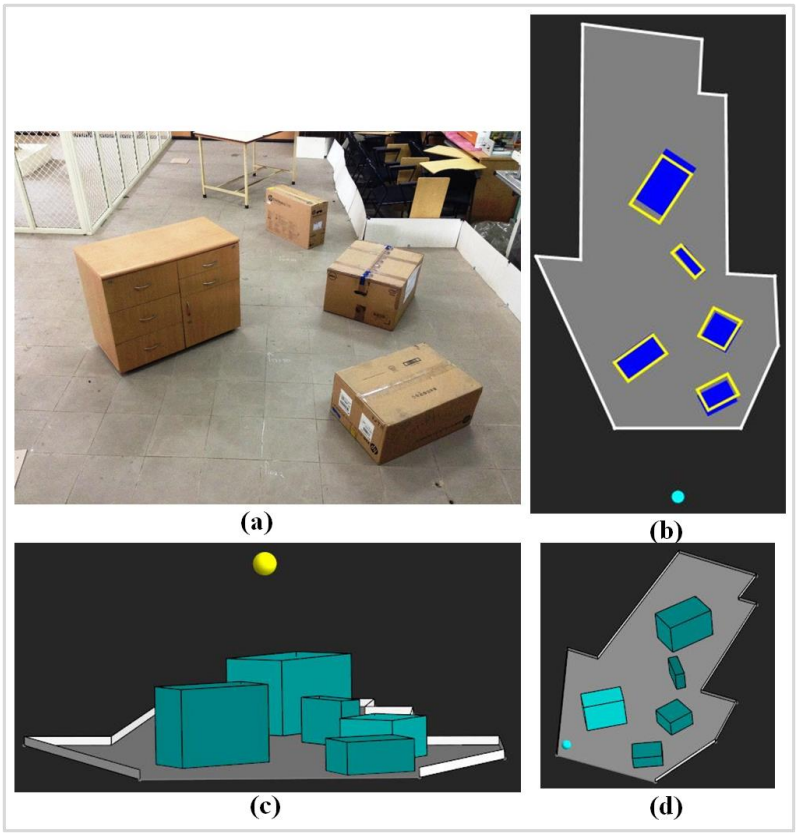


Fig. 6.14: 3PP experimental result – 1. (a) Actual image with five objects in 3PP (b) Constructed floor map (c) 3D reconstruction (d) Another clear 3D view

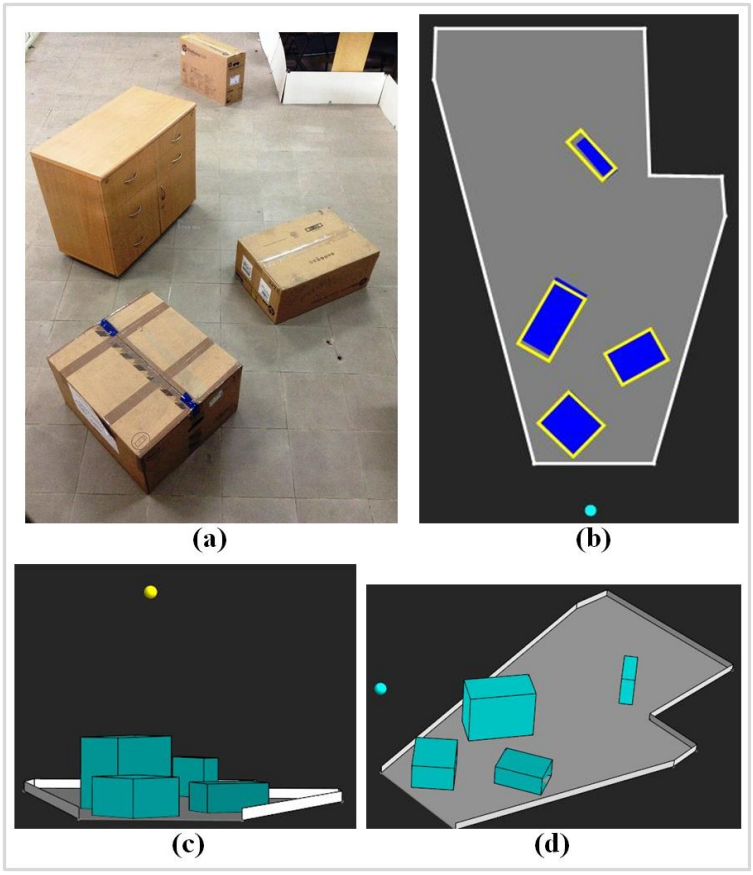


Fig. 6.15: 3PP experimental result – 2. (a) Actual image with five objects in 3PP (b) Constructed floor map (c) 3D reconstruction (d) Another clear 3D view

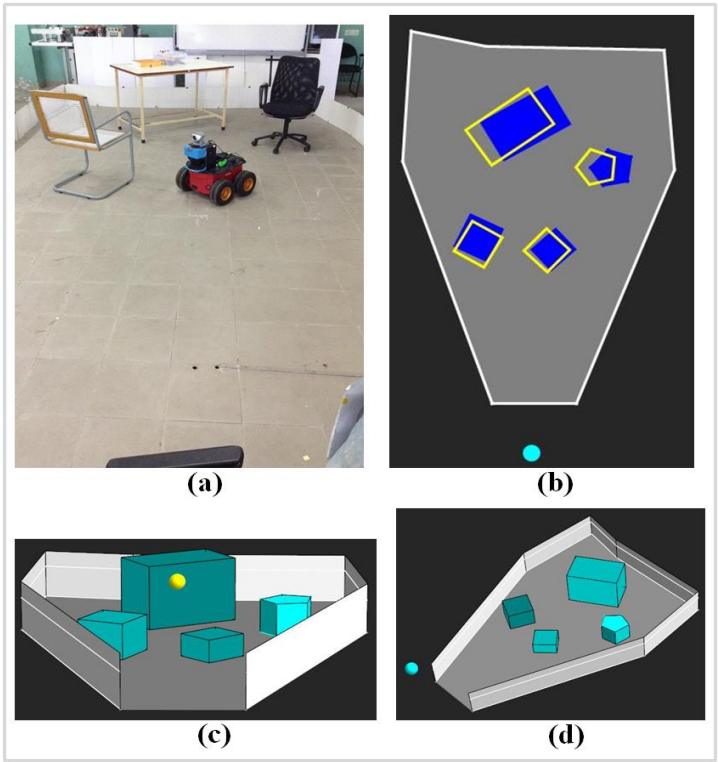


Fig. 6.16: 3PP experimental result – 3. (a) Actual image in 3PP including a Robot, chairs and a table (b) Constructed floor map (c) 3D reconstruction (d) Another clear 3D view

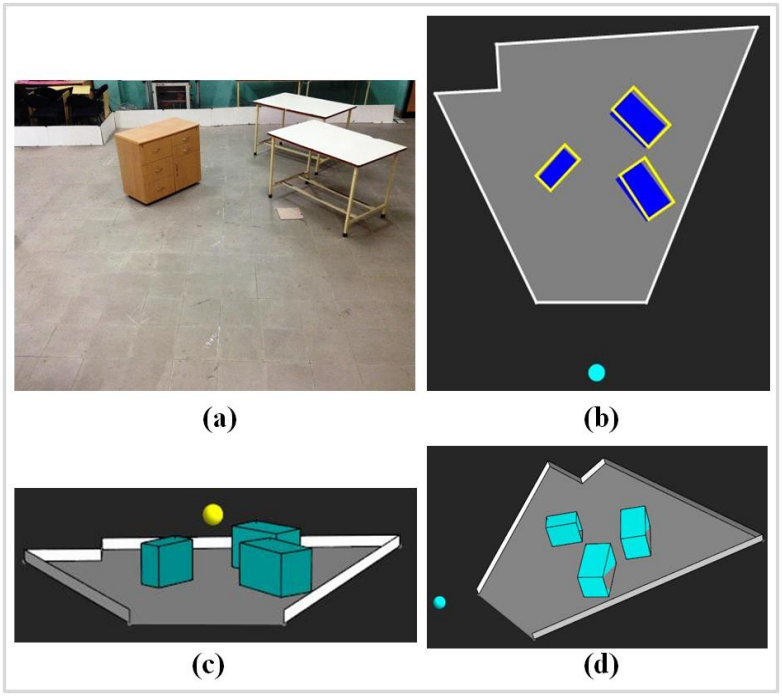


Fig. 6.17: 3PP experimental result – 4. (a) Actual image in 3PP, with one table being occluded by the other table - covering important features (corners and edges). (b) Constructed floor map (c) 3D reconstruction (d) Another clear 3D view

Following set of graphs (Figures 6.18−6.23) illustrates the error results for 3PP images, considering eight sets of example images. For example, experiment 1 consists of five objects.

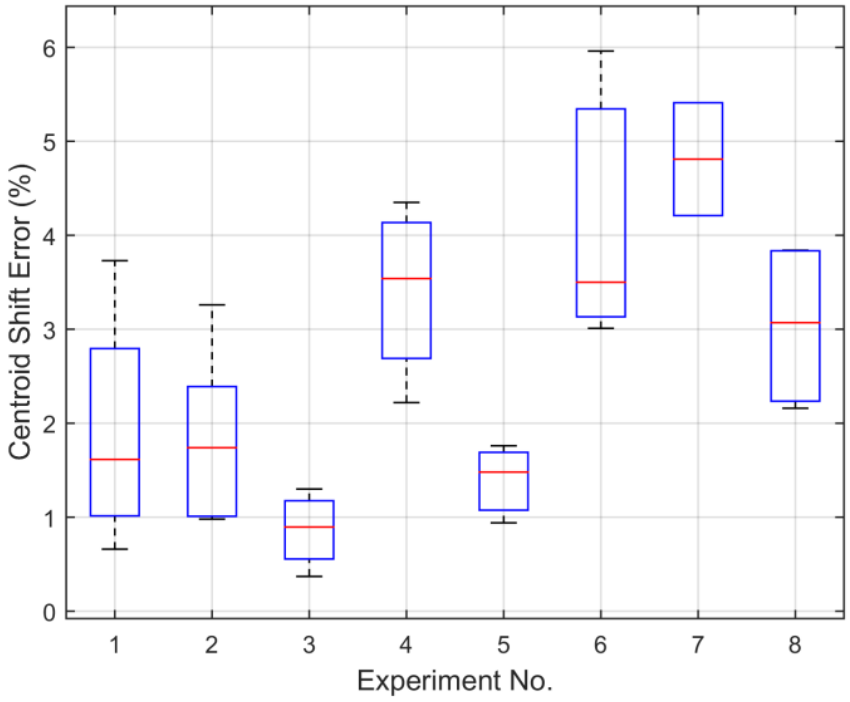


Fig. 6.18: Box plot of percentage deviation of centroids.

Worst case error in centroid shift of the modelled object compared with the actual centroid position of the scaled object is within +6%. Figure 6.19 shows the percentage reductions in distances to centroids of the modelled objects from their actual positions. Boxplot results show that the errors are within the range of ±3%. Results in Fig. 6.20 shows the percentage area overlap between the modelled object and the actual ground truth object.

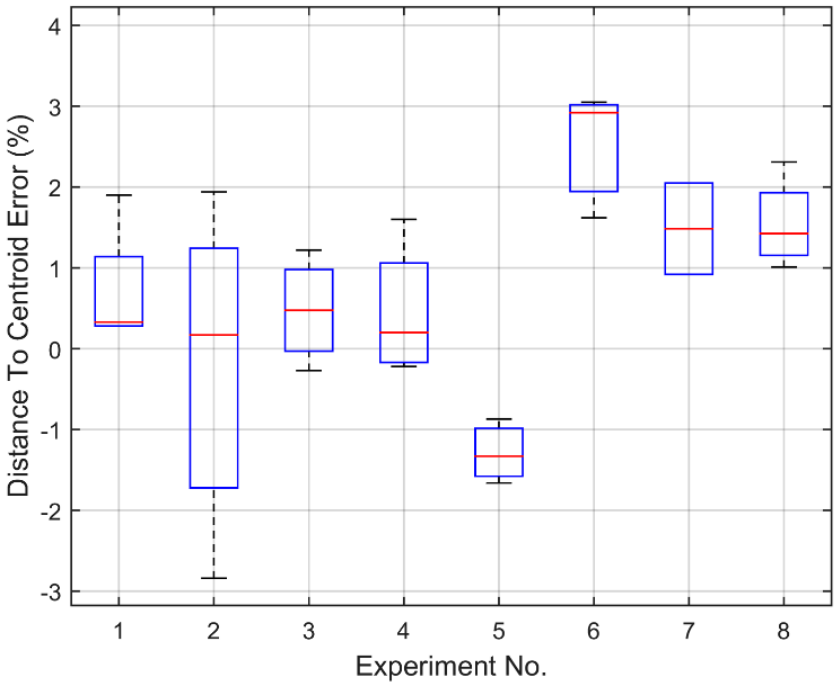


Fig. 6.19: Box plot of percentage reductions in distances to centroids.

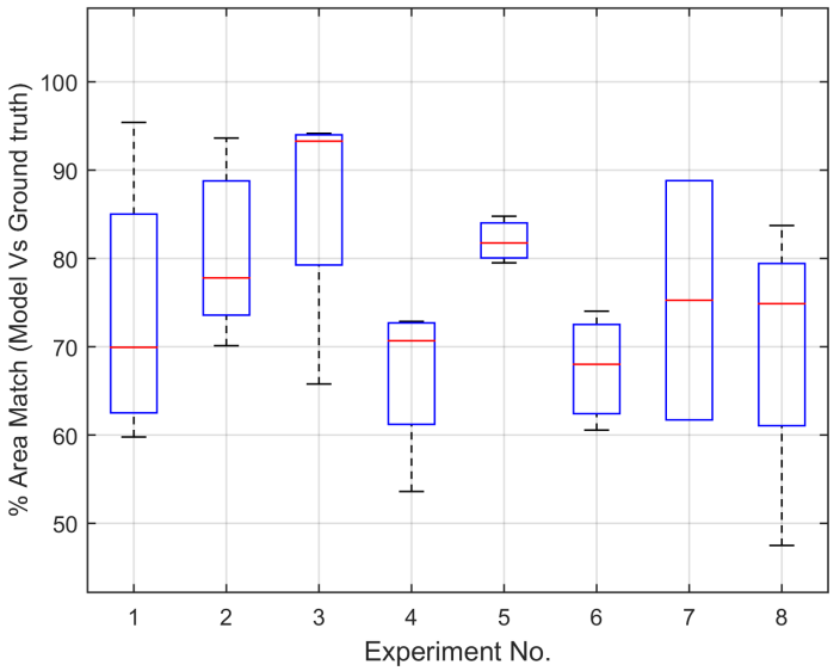


Fig. 6.20: Box plot of percentage area of overlap.

The areas off overlap directly defines the space occupied by the actual object. In most examples, just like we noted in 2PP case, the worst case mismatch was around 55% while the best case match showing the correct overlap was around 95% which is very good approximation from the model. Figures 6.21−6.23 show the boxplot for percentage error in numerical area measurement (considering dimensions), percentage error in angular pose of the centroids relative to world coordinate system and finally percentage error in skew angles (orientation between actual and modelled object w.r.t. reference coordinate system). Their error ranges are within ±25%, ±4% and ±20% respectively.

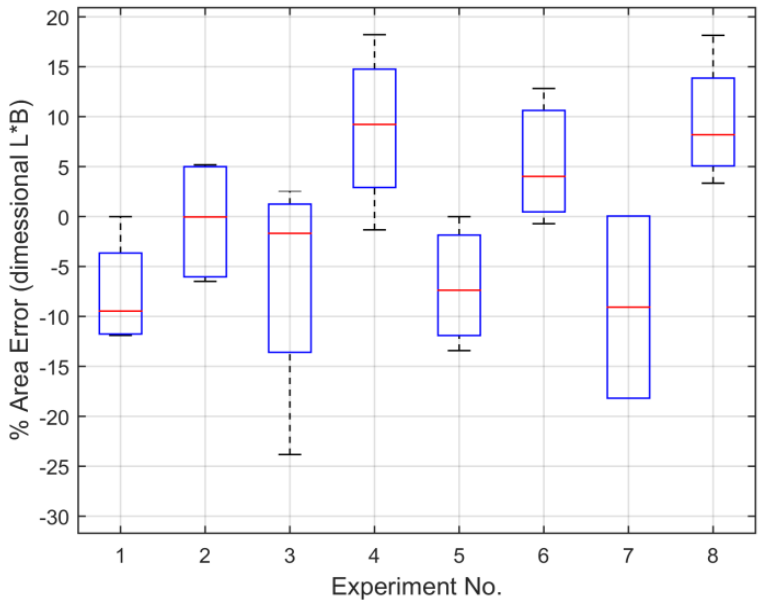


Fig. 6.21: Box plot of percentage error in area measurement.

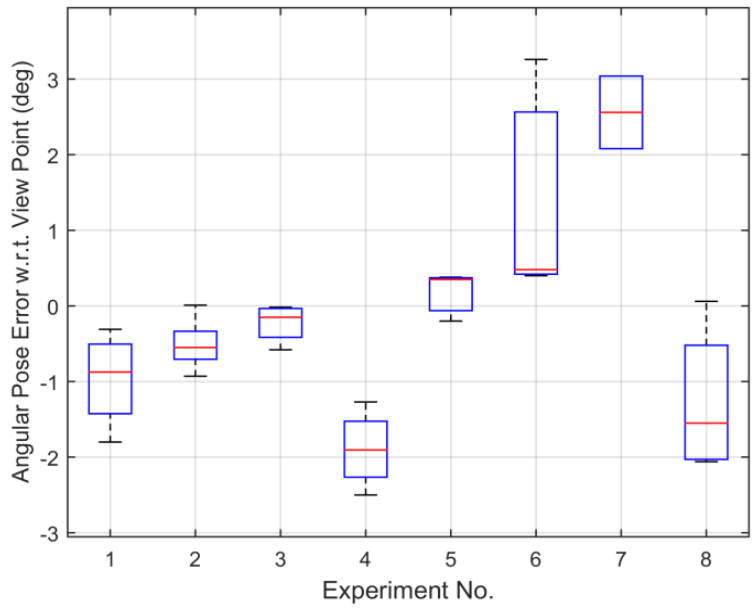


Fig. 6.22: Box plot of percentage error in angular pose of the centroids.

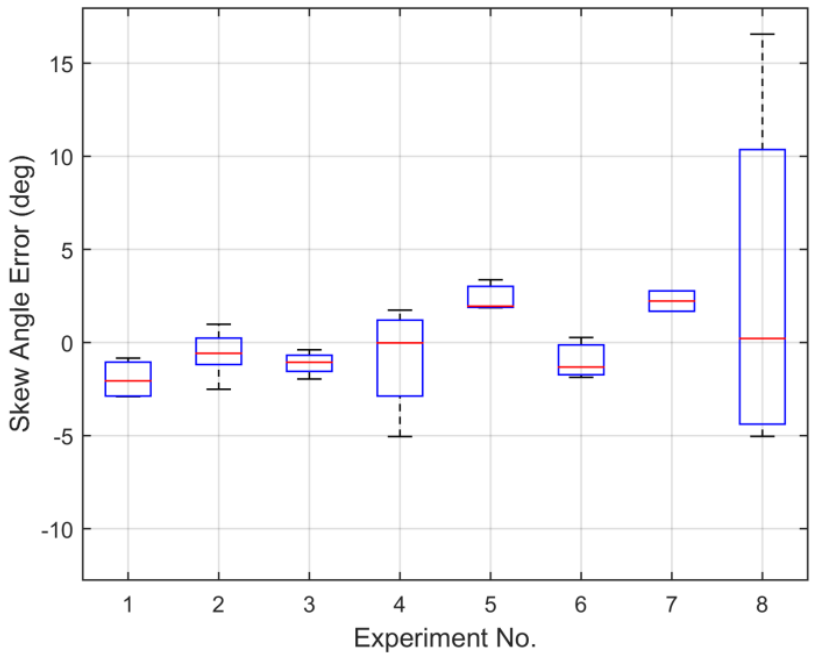


Fig. 6.23: Box plot of percentage error in skew angles.

6.2 MBPP SIMULATION RESULTS

MBPP algorithm has been coded in Python. For comparison purpose the formal post-smoothed A* (A* PS) algorithm (Hart et al., 1968) has been considered with some changes for realistic path planning, which was also coded in Python. A* is known for its optimal path finding capability and the algorithm is complete, meaning it finds path if one exists, else terminate reporting that. Post smoothing for A* is an attempt to minimize unnecessary heading changes as is usually observed in standard A* algorithm. Further, for better evaluation purposes, optimized A* PS has also been optimized in some best possible ways.

Figure 6.24 shows an implementation of MBPP algorithm on a constructed 2D map, obtained from the 3PP single image, shown in Fig. 6.14. The input to the MBPP algorithm is an occupancy grid map, start position of the robot and the goal state as shown in Fig. 6.24(b). MBPP finds all feasible paths from the start to the goal state as shown in Fig. 6.24(c). The algorithm finally outputs the best feasible path that turns out to be optimal, as shown in Fig. 6.24(d).

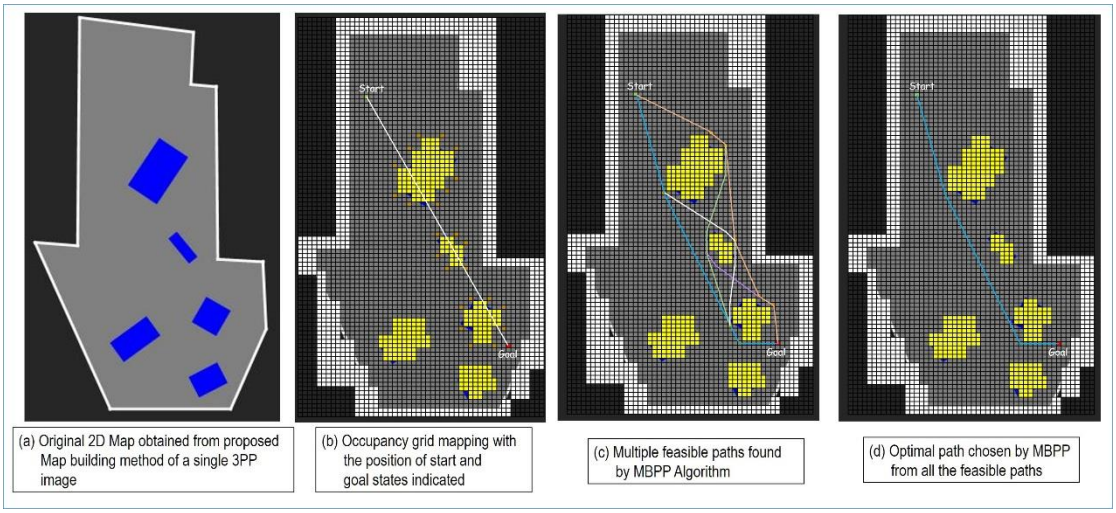
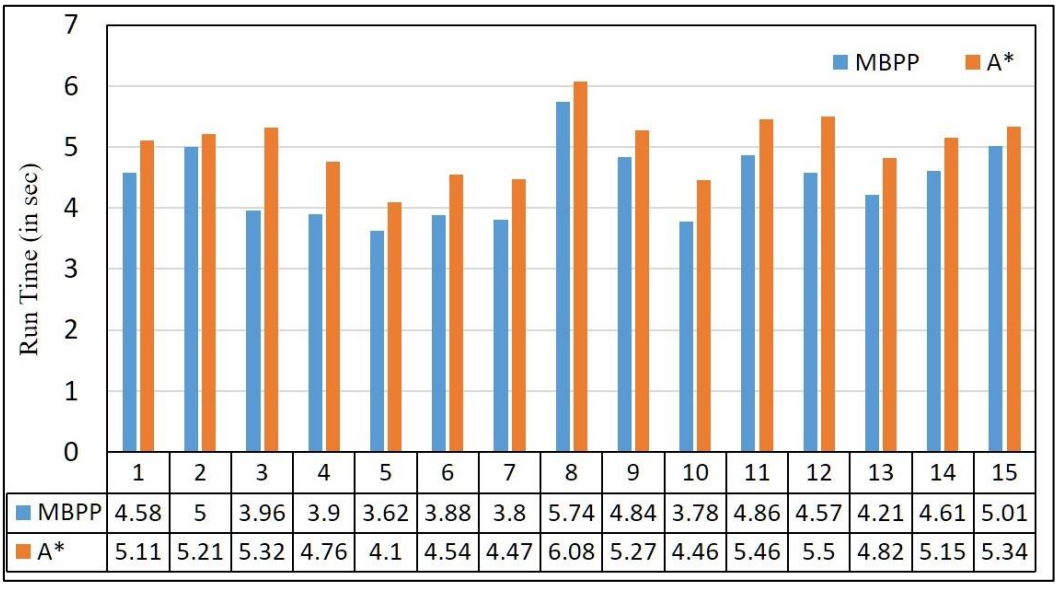


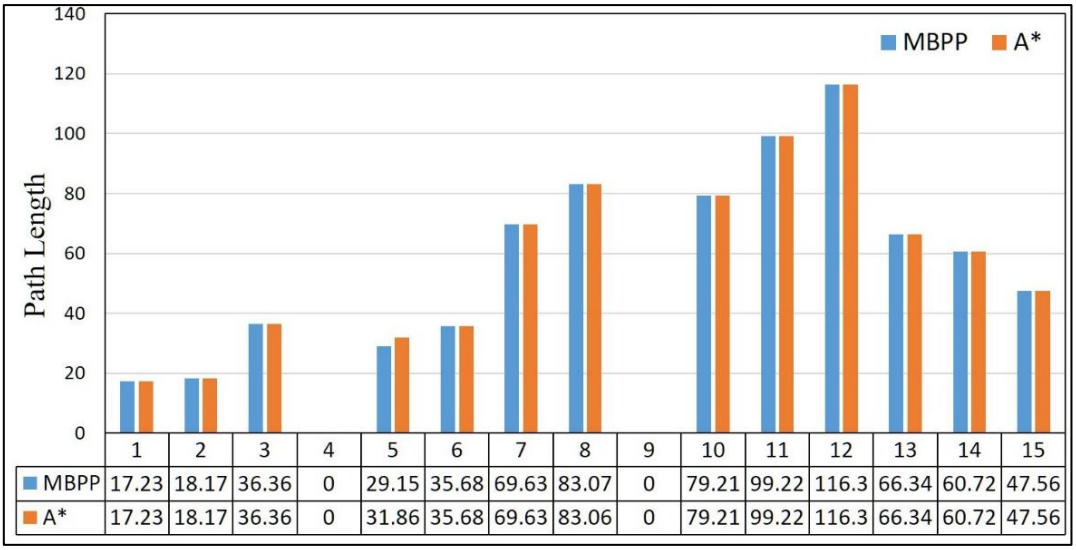
Fig. 6.24: MBPP implementation on a 3PP map. (a) Original 2D map (b) Occupancy grid mapping and implementation of MBPP (c) Possible paths (d) Optimal path chosen by MBPP algorithm

Apart from considering the map results, like the example shown in Fig. 6.24, several other environments had been considered as part of the experiments. Environments considered include real life home maps, work and office maps, game maps, including environments that are free (with no obstacles) and environments with no path (blocked way). For each world considered, ten trials were run for both MBPP and A* PS and the simulation results (path length and run time) were taken for all the trials. Figures 6.25 and 6.26 shows the mean runtime and path length results for fifteen such maps.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MBPP	4.58	5	3.96	3.9	3.62	3.88	3.8	5.74	4.84	3.78	4.86	4.57	4.21	4.61	5.01
A*	5.11	5.21	5.32	4.76	4.1	4.54	4.47	6.08	5.27	4.46	5.46	5.5	4.82	5.15	5.34

Fig. 6.25: MBPP Vs A* (runtime).



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MBPP	17.23	18.17	36.36	0	29.15	35.68	69.63	83.07	0	79.21	99.22	116.3	66.34	60.72	47.56
A*	17.23	18.17	36.36	0	31.86	35.68	69.63	83.06	0	79.21	99.22	116.3	66.34	60.72	47.56

Fig. 6.26: MBPP Vs A* (path length).

As can be seen, path lengths found by MBPP is same as that found by A* PS in almost all the cases. But for some environments, MBPP found shorter paths compared to A* PS. Runtime results for MBPP are better (lesser computation time) than A* PS in all the example cases. Thus, MBPP is shown to have advantage over the most used A* algorithm with post smoothing.

The experimental results provided above for map building confirms that the proposed generalized method is suitable for building free space maps that are well applicable for real time robot navigation tasks. Because the errors are less, the generated map from a single image can thus be fed directly to the robot for path planning purpose. The results of the proposed path planning algorithm are optimal in all the cases and have the added advantage of giving all feasible paths.

CHAPTER 7

SUMMARY AND CONCLUSIONS

7.1 SUMMARY

Methods for building scaled 2D and 3D maps using single 2PP/3PP image were introduced in the first part of this thesis work. Proposed methods are based on the geometry of images using the concept of vanishing point. Although the methods employ construction of top views using geometrical projections, mathematical relations derived in this paper would ease the process of finding the required parameters from the images. The derived relations are minimal in number, are simple, and are easy to implement. The common approach proposed for both the cases (2PP and 3PP) uses the concept of simple geometrical shifting of all objects such that they measure same ground depth as measured by a reference object in the same image; thus, demonstrating robustness of the approach. The process of camera resectioning (also called camera calibration) for finding the camera's intrinsic and extrinsic properties are thus eliminated using simple mathematical relations developed in this paper.

The proposed methods are likely to be best fit for applications where camera calibration process needs to be eliminated or when calibration is not at all possible. Given an image, the methods can literally find the objects' relative placement in the world, provided that the world is captured in perspective, with provision for establishing vanishing points. Although the methods are initially developed for robotic applications to implement planning algorithms to plan a path for navigation tasks, due to the flexibility of the approach, it can also be implemented for the more general single view metrology problems.

3D reconstruction was shown possible using the methods proposed in this work. The idea/method will be useful in many applications like surveillance, image understanding, motion planning, outdoor autonomous vehicle navigations, architecture, augmented reality, virtual reality etc. Theorem 1 proved that 2PP is a special case of 3PP, thus, the mathematical procedure stated for 3PP can be considered as a generalized one, helping in a higher chance of consideration for future applications. A new path planning algorithm (MBPP) had been introduced in this thesis which was implemented on the constructed maps.

7.2 CONCLUSIONS

The following list provides the conclusions deduced from the research on single image metrology for map building and the new path planning algorithm as well.

1. It is possible to generate 2D and 3D maps using single image, captured using monocular cameras. Free space map construction is possible for images either in 2PP or 3PP images. A generalized algorithm has been developed.
2. The proposed method for map construction eliminates the need for camera calibration and use of costly stereo vision cameras for map reconstruction.
3. Experimental results showed that the generated maps are matching well with the ground truths. Images taken with cameras that are positioned at higher heights (more than 2 meters) showed lesser errors and the area coincidence percent of the modelled objects compared with the actual scaled objects' positions showed very good overlap. So, the methods are very well applicable for scenarios like CCD cameras.
4. Table top experiments with small objects (positions and angles accurately measured) and with very less height of the cameras (as low as 20 cm) also gave good results, thus helping in supporting the hypothesis that the proposed methods are applicable for all sizes of the scenes.
5. Observations from most of the experiments showed that construction of the objects which are near to the camera exhibited lesser mismatch with minimal errors, relative to the objects that are positioned farther from the camera. Farthest objects showed some mismatch (within the ranges mentioned in the plots) due to some reasons. One possibility could be distortions (optical, perspective or even image distortions), which appears more prominent for farther objects than nearest objects.
6. Careful selection of vanishing points resulted in very accurate map with minimal errors in area as well as angular poses. Percentage match went up to 96%. But careless selection resulted in mismatch up to 50%. In any case, the deviations in centroid shift are not that high and are within the error ranges as shown in the plots for both the cases.
7. MBPP algorithm proposed here was found to be better than the existing algorithms for path planning.

7.3 FUTURE SCOPE

Although the methods are initially developed for robotic applications of using the built map to implement planning algorithms to plan a path for navigation tasks, because of flexibility of the approach and by observing the results from the experiments, it can clearly be stated that the methods can best be implemented for the more general single view metrology problems. 3D reconstruction was also shown possible using the methods proposed in this work. We therefore suggest the methods for several applications that includes: surveillance, motion planning, outdoor autonomous vehicle navigations, Architecture, image understanding, Augmented reality, Virtual reality, Mixed or even Modulated reality etc. As part of the experiments it was assumed that the position of vanishing points and objects' corners were known and supplied to the algorithm as raw inputs. Incorporating a method for automatic detection of vanishing points and objects' corners along with the developed map building methods would make the whole process a complete package. This could be potential future work in making the proposed algorithm much more user friendly. Further, the combination of data-driven methods (especially for programmatical evaluation of raw inputs to supply to the proposed single image geometric metrology) could be foreseen as an interesting extension of this work.

As part of MBPP algorithm, the initial focus of the work was to develop an algorithm that could possibly be employed for both offline and online cases with minor changes such that whenever robot encounters changes in the given environment, it could switch from offline mode to online mode, in order to plan near optimal paths. While in this paper, we experimented and proved the applicability of MBPP algorithm for offline environments, the adaptability and usage of this algorithm for online mode is left for future research. Optimization of bug travel lengths, minimization of bug generations and smoothing of paths at all heading changes (by incorporating Bezier or spline curves) are three more concerns for future works.

REFERENCES

1. **Antich, J., Ortiz, A., Minguez, J.**: A Bug Inspired Algorithm for Efficient Anytime Path Planning. In IEEE/RSJ International Conference Robots and Systems (IROS), pp 5407-5413 (2009)

2. **Bresenham, J.**: Algorithm for Computer Control of a Digital Plotter. IBM Systems Journal, Vol. 4, no. 1, pp 25-30 (1965)

3. **Byres, K., Henle, J.**: Where the Camera Was. This Magazine, 77:4, pp 251-259 (2004)

4. **Caprile, B., Torre, V.**: Using Vanishing Points for Camera Calibration. Int. J. Comp. Vis., 4, pp 127-140 (1990)

5. **Chaudhuri, S., Rajagopalan, A.**: Depth from Defocus: A Real Aperture Imaging Approach. ISBN: 978-1-4612-1490-8, Springer Verlag (1999), doi: 10.1007/978-1-4612-1490-8

6. **Ching, F.D.K., Juroszek, S.P.**: Design Drawing. 2nd Edition, Wiley Publications, Washington (2010) ISBN: 978-0-470-53369-7

7. **Cipolla, R., Drummond, T., Robertson, D.**: Camera Calibration from Vanishing Points in Image of Architectural Scenes. In: Proc. British Mach. Vis. Conf., 38.1-38.10 (1999), doi:10.5244/C.13.38

8. **Coughlan, J.M., Yuille, A.L.**: Manhattan World: Compass Direction from a Single Image by Bayesian Inference. In Int. Conf. on Comp. Vis., pp 941–947 (1999), doi: 10.1109/ICCV.1999.790349

9. **Crannell, A.**: Where the Camera Was, Take Two. Mathematics Magazine, Published by Mathe. Assoc. of America 79(4), pp 306-308 (2006)

10. **Criminisi, A., Reid, I.D., Zisserman, A.**: Single View Metrology. Int. J. Comp. Vis., 40(2), pp 123-148 (2000)

11. **Daniel, K., Nash, A., Koenig, S., Felner, A.**: Theta*: Any-Angle Path Planning on Grids. Journal of Artificial Intelligence Research, Vol. 39, pp 533-579 (2010)

12. **Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.**: MonoSLAM: Real-Time Single Camera SLAM. IEEE Trans. Pattern Anal. Mach. Intell., 29(6), pp 1052-1067 (2007)

13. **Delage, E., Lee, H., Ng, A.**: A Dynamic Bayesian Network Model for Autonomous 3D Reconstruction from a Single Indoor Image. In Proc. IEEE Conf. on Comp. Vis. & Patt. Recog., pp 2418–2428 (2006), doi: 10.1109/CVPR.2006.23

14. **Delage, E., Lee, H., Ng, A.**: Automatic Single-Image 3D Reconstructions of Indoor Manhattan World Scenes. In Int. Symp. of Robo. Res., vol. 28, pp 305-321 (2005), doi: 10.1007/978-3-540-48113-3_28

15. **Durou, J.D.**: Numerical Methods for Shape-from-shading: A New Survey with Benchmarks. J. of Comp. Vis. and Img. Understanding, 109(1), pp 22-43 (2008)

16. **Eggar, E.H.**: Pinhole Cameras, Perspective, and Projective Geometry. The American Mathe. Monthly, Pub. Math. Assoc. of America, 105(7), pp 618-630 (1998)

17. **Einhorn, E., Schroter, C., Gross, H.M.**: Monocular Scene Reconstruction for Reliable Obstacle Detection and Robot Navigation, in: Proc. 4th European Conf. on Mobile Robots (ECMR), pp 7-12 (2009)

18. **Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W.**: 3-D Mapping with an RGB-D Camera. IEEE Robotics and Automation Society. Vol. 20, issue 1, pp 177-187 (2013), doi: 10.1109/TRO.2013.2279412

19. **Ferguson, D., Stentz, A.**: Using Interpolation to Improve Path Planning: The field D* algorithm. Journal of Field Robotics, Vol. 23, issue 2, pp 79-101 (2006)

20. **Gozali, F.K.**: Two-Point Perspective 3D Modeling from a Single Image: A Tour into the Picture Experience (2006)

21. **Gupta, A., Satkin, S., Efros, A. A., Hebert, M.**: From 3D Scene Geometry to Human Workspace. In Proc. Of Comp. Vis. & Patt. Recog., pp 1961–1968 (2011)

22. **Han, F., Zhu, S.C.**: Bayesian Reconstruction of 3D Shapes and Scenes from a Single Image. In Proc. Int. Work. on Higher-Level Knowl. in 3D Model. and Motion Anal. (2003), ISBN:0-7695-2049-9

23. **Hart, R.**: Finding the Center of Projection of a Perspective Projection (2009), rnhart. http://bit.ly/2axHcEK

24. **Hart, P., Nilsson, N., Raphael, B.**: A formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, Vol. 4, no. 2, pp 100-107 (1968)

25. **Hartley, R.I., Sturm, P.**: Triangulation. Comput. Vis. and Image Understanding, 68(2), pp 146-157 (1997)

26. **He, B.W., Li, Y.F.**: A Novel Method for Camera Calibration using Vanishing Points. Int. Conf. Mechatronics and Mach. Vis. in Practice, pp 44-47 (2007), doi: 10.1109/MMVIP.2007.4430712

27. **Hebert, M., Lin, J., Satkin, S.**: Data-Driven Scene Understanding from 3D Models. In British Mach. Vis. Conf. (2012), doi: 10.5244/C.26.128

28. **Hedau, V., Hoiem, D., Forsyth, D.**: Recovering the Spatial Layout of Cluttered Rooms. In Proc. Int. Conf. on Comp. Vis., pp 1849-1856 (2009)

29. **Hedau, V., Hoiem, D., Forsyth, D.**: Recovering Free Space of Indoor Scenes from a Single Image. IEEE Conf. Comp. Vis. & Patt. Recog., pp 2807-2814 (2012), doi: 10.1109/CVPR.2012.6248005

30. **Hedau, V., Hoiem, D., Forsyth., D.**: Thinking Inside the Box: Using appearance models and context based on room geometry. In Europ. Conf. Conmp. Vis., pp 224-237 (2010)

31. **Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D.**: RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. The 12th Int. Symp. on Exp. Robotics (2014), ISBN:978-3-642-28572-1, DOI: 10.1007/978-3-642-28572-1_ 33, 477-491

32. **Hoiem, D., Efros, A., Hebert, M.**: Recovering Surface Layout from an Image, Int. J. Comp. Vis., 75(1), pp 151-172 (2007)

33. **Hoiem, D., Efros, A.A., Hebert, M.**: Geometric Context from a Single Image. In: Int. Conf. on Compt. Vis., pp 654-661 (2005)

34. **Hoiem, D., Efros, A.A., Hebert, M**: Automatic photo pop-up. In Proc. ACM SIGGRAPH, 24(3), pp 577-584 (2005), doi: 10.1145/1073204.1073232

35. **Hoiem, D., Efros, A.A., Hebert, M.**: Putting Objects in Perspective. Int J. Comp. Vis., 80(3), (2008), doi:10.1007/s11263-008-0137-5

36. **Horn, B.K.P.: Shape From Shading**: A Method for Obtaining the Shape of a Smooth Opaque Object From One View. Technical Report, 232, MIT AI Lab, Cambridge (1970)

37. **Horry, Y., Anjyo, K., Arai, K.**: Tour into the Picture: Using a Spidery Mesh Interface to Make Animation from a Single Image. In: Proc. ACM SIGGRAPH Conf. Comput. Graphics and interactive techniques, pp. 647-658 (1997), doi: 10.1145/258734.258854

38. **Jang, J., Rossignac, J.R.**: Determination of the Correct Eye Position for Viewing Perspective Images of 3D Scenes. Georgia Institute of Technology (2001).

39. **Kamencay, P., Breznan, M., Jarina, R., Lukac, P.**: Improved Depth Map Estimation from Stereo Images Based on Hybrid Method. J. Radio engineering, Vol.21, pp 70-78 (2012).

40. **Kim, T., Seo, Y., Hong, K.**: Physics-based 3D Position Analysis of a Soccer Ball from Monocular Image Sequences. In: Proc. Int. Conf. Comp. Vis., pp 721-726 (1998)

41. **Koenig, S., Likhachev, M.**: Improved Fast Re-planning for Robot Navigation in Unknown Terrain. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2002a).

42. **Koenig, S., Likhachev, M.**: Incremental A*. Advances in Neural Information Processing Systems, MIT Press (2002b).

43. **Kogecka, J., Zhang, W.**: Efficient Computation of Vanishing Points. Proc. IEEE Int. Conf. Robotics and Aut., (2002), doi: 10.1109/ROBOT.2002.1013365

44. **Kosecka, J., Zhang, W.**: Extraction, Matching, and Pose Recovery Based on Dominant Rectangular Structures. J. Comp. Vis. & Underst., 100(3), pp 274-293 (2005)

45. **Kosecka, J., Zhang, W.**: Video compass. In Proc. European. Conf. Comp. Vis. Springer-Verlag (2002) ISBN:3-540-43748-7

46. **Lee, D., Gupta, A., Hebert, M., Kanade, T.**: Estimating Spatial Layout of Rooms using Volumetric Reasoning about Objects and Surfaces. In Proc. Adv. in Neur. Inf. Process. Sys. (NIPS) 23, pp 1288–1296 (2010)

47. **Lee, D., Hebert, M., Kanade, T.**: Geometric Reasoning for Single Image Structure Recovery. In IEEE Conf. Comp. Vis. & Patt. Recog., pp 2136-2143 (2009), doi: 10.1109/CVPR.2009.5206872

48. **Li, B., Peng, K., Ying, X., Zha, H.**: Simultaneous Vanishing Point Detection and Camera Calibration from Single Images. In: Advances in Visual Computing, Lect. Notes in Comp. Sci., Vol. 6454, pp 151-160 (2010), Springer Berlin, Heidelberg

49. **Liebowitz, D., Criminisi, A., Zisserman, A.**: Creating Architectural Models from Images. In Proc. EuroGraphics, 18(3), pp. 39-50 (1999)

50. **Lumelsky, V., Stepanov, A.**: Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape. Algorithmica, Springer-Verlag New York Inc, (1987)

51. **Mauldin, J.H.**: Perspective Design: Advanced Graphic and Mathematical Approaches. 2nd Edition, Van Nostrand Reinhold Co., New York (1985) ISBN: 978-0-442-26408-6

52. **Ng, J., Braunl, T.**: Performance Comparison of Bug Navigation Algorithms. Journal of Intelligent and Robotics Systems, Vol. 50, issue 1, pp 73-84 (2007)

53. **Pero, L., Guan, J., Brau, E., Schlecht, J., Barnard, K.**: Sampling Bedrooms. In Proc. IEEE Conf. on Comp. Vis. & Pat. Recog., pp. 2009–2016 (2011)

54. **Pero, L., Bowdish, J., Fried, D., Kermgard, B., Hartley, E., Barnard, K.**: Bayesian Geometric Modeling of Indoor Scenes. IEEE Conf. Comp. Vis. and Pattern Recog., pp 2719-2726 (2012), doi: 10.1109/CVPR.2012.6247994

55. **Proesmans, M., Tuytelaars, T., and Van Gool, L.J.**: Monocular Image Measurements. Technical Report, Improofs - M12T21/1P, K.U.Leuven (1998)

56. **Robert W.G.**: Creative Perspective. Thames and Hudson, London (1975)

57. **Rother, C.**: A New Approach for Vanishing Point Detection in Architectural Environments. J. Image and Vis. Computing, 20(9-10), pp 647-655 (2002)

58. **Saez, J.M., Escolano, F.**: A Global 3d Map-Building Approach Using Stereo Vision. In Proc. IEEE Int. Conf. on Robotics and Automation (ICRA'04). Vol.2, pp 1197-1202 (2004), doi: 10.1109/ROBOT.2004.1307987

59. **Saini, V., Gade, S., Prasad, M., Chatterjee, S.**: The 3-Point Method: A Fast, Accurate and Robust Solution to Vanishing Point Estimation. Int. Conf. Comp. Graphics, Visual. and Compt. Vis., 21, pp 151-158 (2013)

60. **Saxena, A., Chung, S.H., Ng, A.Y.**: 3-D Depth Reconstruction from a Single Still Image. Int. J. Comp. Vis., 76(1), pp 53-69 (2008)

61. **Scharstein, D., Szeliski R.**: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. Int. J. Comput. Vis. 47(1), pp 7-42 (2002)

62. **Stentz, A.**: Optimal and Efficient Path Planning for Partially-Known Environments. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Vol. 4, pp 3310-3317 (1994)

63. **Thrun, S.**: Robotic Mapping: A Survey; Book - Exploring Artificial Intelligence in the New Millennium, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (2002) ISBN:1-55860-811-7.

64. **Yap, P., Burch, N., Hilte, R., Schaeffer, J.**: Any Angle Path Planning for Computer Games. Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (2011)

65. **Yu, S., Zhang, H., Malik., J.**: Inferring spatial layout from a single image via depth-ordered grouping. In Proc. IEEE Conf. on Comp. Vis. & Patt. Recog., pp 1–7 (2008), doi: 10.1109/CVPRW.2008.4562977

66. **Zhang, Z.**: A Flexible New Technique for Camera Calibration. IEEE Trans. Pattern Analy. Mach. Intell., 22(11), pp 1330-1334 (2000)

67. **Zhang, R., Tsai, P.S., Cryer, J.E., Shah, M.**: Shape from Shading: A Survey. IEEE Trans. on Pattern Anal. and Machine Intell., 21(8), pp 690-706 (1999)

68. **Zhu, X., Cao, Q., Yokoi, H., Jiang, Y.**: Large Scale Indoor 3D Mapping Using RGB-D Sensor. In Lecture Notes in Computer Science, vol 9834. Springer, Cham, Intelligent Robotics and Applications. ICIRA (2016), doi: 10.1007/978-3-319-43506-0_27

69. **Zhuo, S., Sim, T.**: defocus map estimation from a Single Image. J. Patt. Recog., Vol. 44 Issue 9, pp 1852–1858 (2011), doi: 10.1016/j.patcog.2011.03.009.

APPENDICES

A PSEUDOCODES

A.1 Algorithm 1: Generalized algorithm for Single Image Metrology of 2PP and 3PP

A.1(a): Main Algorithm

Algorithm 1: Generalized Algorithm for Single Image Metrology of 2PP and 3PP Images

1. **Main ()**
2. **{**
3. Initialize *n* = No. of objects (of interest) in the image
4. Bottom left corner of the Image frame as origin for the reference coordinate system in *(x, y)*
5. *imgcor* = coordinates of top right corner point of the image frame
6. **Inputs:**
7. ***p*** = vector of base nearest corners of all *n* objects $= \{(x_1, y_1)^P, (x_2, y_2)^P, \ \ldots\ldots (x_n, y_n)^P\}$
8. ***q*** = vector of base right most corners of all *n* objects $= \{(x_1, y_1)^q, (x_2, y_2)^q, \ \ldots\ldots (x_n, y_n)^q\}$
9. ***r*** = vector of base left most corners for all *n* objects $= \{(x_1, y_1)^r, (x_2, y_2)^r, \ \ldots\ldots (x_n, y_n)^r\}$
10. **v** = vector of top nearest corners of all *n* objects $= \{(x_1, y_1)^v, (x_2, y_2)^v, \ \ldots\ldots (x_n, y_n)^v\}$
11. ***s*** = vector of top left corners of all *n* objects $= \{(x_1, y_1)^s, (x_2, y_2)^s, \ \ldots\ldots (x_n, y_n)^s\}$
12. $\chi = \begin{cases} 2 \ for \ 2PP \ case \\ 3 \ for \ 3PP \ case \end{cases}$
13. ***Repeat*** for all ***k***
    $vp[k] = \{(x_1, y_1)_1^k, \ldots\ldots (x_1, y_1)_\dagger^k\}$
14. ***until (k** = 1 to n)*
15. ***box_topview**(i=1)*
16. **if** *(χ == 2)* **then**
    a. *imcenter = imgcor/2*
    b. *ht*[1] = *EuclideanDistance(v*[1], *p*[1]*)*
    c. *d* = perpendicular distance from view point *vpoint*[1] to the line joining *VP*'s; *vp*[1][1] and *vp*[1][2]
    d. **for all** (*i = 2 to n*) **do**
       ht[i] = *euclid*(*v*[i], *p*[i])
       h_d = perpendicular distance between parallel lines through *p*[i] and *p*[1], both at an angle θ_{roll} to the horizontal
       h2 = perpendicular distance from base point p[i] to the line joining *VP*'s; *vp*[1][1] and *vp*[1][2]
       $\mathrm{dp}[i] = \frac{\mathrm{h_d}.\mathrm{d}}{\mathrm{h_2}}$
       ***box_topview*(i)**
17. **if** *(χ == 3)* **then**
    a. imcenter = Point of intersection of altitudes of the triangle formed by three *VP*'s; *vp*[i], where i = 1, 2, 3
    b. $d = \sqrt{OG.OC}$
    c. $\theta = \tan^{-1}\left(\frac{d}{OC}\right)$

d. **for all** *i = 1 to n* **do**

*k*1 = perpendicular distance from base point p[i] to the line joining *VP*'s; *vp*[1][1] and *vp*[1][2]

$h[i] = k1.\cos\theta$

*k*2 = perpendicular distance between parallel lines through *imcenter* and $p[i]$, both at an angle θ_{roll} to the horizontal

$\beta[i] = tan^{-1}\left(\frac{k2}{d}\right)$

i. **if** *(i == 1)* **then**

$b[1] = h[i]/\,tan(\theta + \beta[1])$

campos = point on a line through *vpoint*[1] perpendicular to line joining *vp*[1][1] and *vp*[1][2] at a distance *b*[1] from it towards *vpoint*

ke[1] = perpendicular distance between parallel lines through *v*[1] and *p*[1], both at an angle θ_{roll} to the horizontal

*k*2 = perpendicular distance between parallel lines through *imcenter* and *v*[1], both at an angle θ_{roll} to the horizontal

if $imcenter^y < v[1]^y$ **then** *k*3 = –*k*3

$\gamma[1] = tan^{-1}\left(\frac{k3}{d}\right)$

$ht[1] = ke[1] * \cos(\theta) + ke[1] * \sin(\theta) * \tan(\theta + \gamma[1])$

ii. **else**

$qpe[i] = (h[1] - h[i])/\,tan(\theta + \beta[1])$

$pq[i] = (h[1] - h[i])/\,tan\,\theta$

bnew[i] = *b*[1] + *pq*[i] + *qpe*[i]

box_topview(*i*)

$ke[i] = (h[1] - h[i]) * \cos(\beta[i])/\sin(\theta + \beta[i])$

k2= perpendicular distance between parallel lines through *v*[i] and $p[i]$, both at an angle θ_{roll} to the horizontal

*k*3 = perpendicular distance between parallel lines through *imcenter* and *v*[i], both at an angle θ_{roll} to the horizontal

if $imcenter^y < v[i]^y$ **then** *k*3 = –*k*3

$\gamma[i] = tan^{-1}\left(\frac{k3}{d}\right)$

$L[i] = k2 + ke[i] * \tan(\beta[i]) - ke[1] * \tan(\gamma[i])$

$ht[i] = L[i] * \cos(\theta) + L[i] * \sin(\theta) * \tan(\theta + \gamma[i])$

iii. **show** ***box***[i] in a reference coordinate map for all (i *= 1 to n*)

18. **if** *(χ == 2)* **then**

show *vpoint* in the above coordinate image

19. **elseif** *(χ == 3)*

show *campos* in the above coordinate image

20. **}**

21. **End of Main()**

A.1(b): Function : *box_topview()*

Function : *box_topview(i)*

1. ***box_topview(i)***
2. ***{***
3. *vpoint*[i] = Point of intersection of vertical line through O inclined at (90^0–θ_{roll}) AND vanishing circle through *vp*[i][1] and *vp*[i][2]
4. Γ_1= slope/angle between line joining *vp*[i][2]and *vpoint*[i] with horizontal
5. Γ_2= slope/angle between line joining *vp*[i][1] and *vpoint*[i] with horizontal
6. $P[i]_1$ = Point of intersection of line through *p*[i] making angle (90^0–θ_{roll}), AND horizon line joining *vp*[i][1] and *vp*[i][2]
7. $Q[i]_1$ = Point of intersection of line through *q*[i] making angle (90^0–θ_{roll}), AND horizon line joining *vp*[i][1] and *vp*[i][2]
8. $R[i]_1$ = Point of intersection of line through *r*[i] making angle (90^0–θ_{roll}), AND horizon line joining *vp*[i][1] and *vp*[i][2]
9. **if** ($i == 1$) **then**
    a. $Pe[i]_1 = P[i]_1$
10. **else**
    a. **if** ($\theta_{roll} == 0$) **then**
        if ($\chi == 2$) **then**
            $Pe[i]^y = d_p[i] + vp[i]^y$
        elseif ($\chi == 3$)
            $Pe[i]^y = campos + bnew[i]$
        Pe[i] = point of intersection of line through *vpoint*[i], $P[i]_1$ with horizontal line $y = Pe[i]^y$
    b. **else**
        if ($\chi == 2$) **then**
            $dummy = \{P\{i\}_1^x - d_p[i].\sin(\theta_{roll}), P\{i\}_1^y + d_p[i].\cos(\theta_{roll})\}$
        elseif ($\chi == 3$)
            $dummy = \{campos^x - bnew[i].\sin(\theta_{roll}),$
            $campos^y + bnew[i].\cos(\theta_{roll})\}$
        Pe[i] = point of intersection of line through *dummy* having slope θ_{roll} AND line through *vpoint*[i] and $P[i]_1$
11. *Qe* [i] = point of intersection of line through *dummy* having slope θ_{roll} AND line through *vpoint*[i] and $Q[i]_1$
12. *Re* [i] = point of intersection of line through *dummy* having slope θ_{roll} AND line through *vpoint*[i] and $R[i]_1$
13. *box*[i] = rectangle formed using *Pe*[i], *Qe*[i] and *Re*[i]
14. ***Return*** *box*[i], *vpoint*[i]
15. }

A.2 **Algorithm 2:** Multi-Bug Path Planning Algorithm

A.2(a): Main Algorithm

Algorithm 2 MBPP Algorithm pseudocode

```
1: Main ()
2: Initialize S_S, repetiotion = True
3: while repetiotion == True do
4:     Initialize:
       bug =1, rr, m_line, S_g, v_list = [S_S], hit_list = [], leave_list = [S_S]
5:     repeat ∀ bugs
6:       Snew = action(S, bug); satisfying C_4
7:       if Snew == S_g then
8:             v_list.append(Snew)
9:             Terminate this bug
10:      else if O_i NOT in neigh(Snew) then
11:            Continue
12:      else
13:            if bug hits Oi first time then
14:                Generate new bug ∀p ∈ (bug, bug+1)
15:                hit_list[p].append(Snew)
16:                v_list[bug +1] = v_list[bug]
17:            else
18:                if condition C_4 True then
19:                    Terminate the bug
20:                else if not LOS(v_list, Snew) then
21:                    v_list.append(S_n) where S_n ∈ neighb(S,Snew), S_n is visible to latest
                       state in v_list
22:                else if condition C_3 True then
23:                    leave_list.append(Snew); Add bug to wait_list
24:                else if condition C_2 True then
25:                    Stop wall following
26:                else Continue
27:            S = Snew
28:    until No bug is left for iteration
29:    n = no. of live bugs
30:    if S_g not in all(v_list[i]) ∀ i ∈ (1,…n) then
31:          Return no path found
32:          break
33:    else if S_g not in any(v_list[i]) ∀ i ∈ (1,…n) then
34:          Append states to waiting bugs based on their leave_list
35:    else Continue
36:  for all bugs do
37:          dist = Sum of euclidian distance between successive elements of v_list[bug]
38:  select_path = {v_list: v_list has least dist}
39:  for all successive states in select_path do
40:          LOS and Path_smooth check
41:          if NOT visible then
42:              S = Pre-state to obstacle identified state
43:              repetiotion = True
44:              remove select_list from v_list
45:          else
46:              repetiotion = False
47:              return shortest path is found
```

A.2(b): Function : *line_of_sight Algorithm*

Algorithm 2 LOS(S_1,S_2)

Initialize LOS = True
(x_1,y_1) are S_1 state coordinates; (x_2,y_2) are S_2 state coordinates
$R_d = (x_2 - x_1)$; $C_d = (y_2 - y_1)$; $x_k = x_1$; $y_k = y_1$
$diff = abs(R_d) - abs(C_d)$; $n = abs(R_d) + abs(C_d)$
x_{inc} = 1 if ($x_2 > x_1$) else −1
y_{inc} = 1 if ($y_2 > y_1$) else −1
for all $i \forall (1,n)$ **do**
    if diff > 0 **then**
        $x_i = x_1 + x_{inc}$; $diff = abs(C_d)*2$; $x_j = (x_1 + x_i)/2$
        if (x_j,y_k) occupied **then**
            return False; note = (x_j,y_k)
        else $x_j = x_i$; $x_k = x_j$
    else
        $y_i = y_1 + y_{inc}$; $diff -= abs(R_d)*2$; $x_j = (y_1 + y_i)/2$
        if (x_k,y_j) occupied **then**
            return False; note = (x_k,y_j)
        else $y_1 = y_i$; $y_k = y_j$
if $i = n$; *return* True

A.2(c): Function : *path_smooth Algorithm*

Algorithm 3 Path_smooth(*select_list*)

K = *select_list*
$S = K[1]$; $j = 1$; $S_E = K[len(K) + 1]$
while $S \neq S_E$ **do**
    if $LOS(S, K[j+1])$ **then**
        Remove states between (S, $K[j]$) from K
        $S = K[j]$
    $j = j + 1$
return(K)

B PROCEEDURE FOR 2PP TOP VIEW CONSTRUCTION

In this section, we brief the basic steps usually involved in finding view point of the camera in 2PP (Hart, 2009) as well as the procedure for constructing top view of the object (Ching, 36), given a perspective image with two vanishing points. For understanding and evaluation of the proposed method, we considered images with rectangular cuboid shaped objects. Given an image with vanishing points determined or marked on it, we can construct a vanishing circle with diameter equal to the distance between the two vanishing points.

Figure B.1 shows an object in perspective along with the terminology used in this thesis. This configuration/layout will be used as reference for explanation throughout this section. The two vanishing points (VP_1 and VP_2) lie on a horizontal straight line termed '*horizon*' as mentioned in section 2.1.

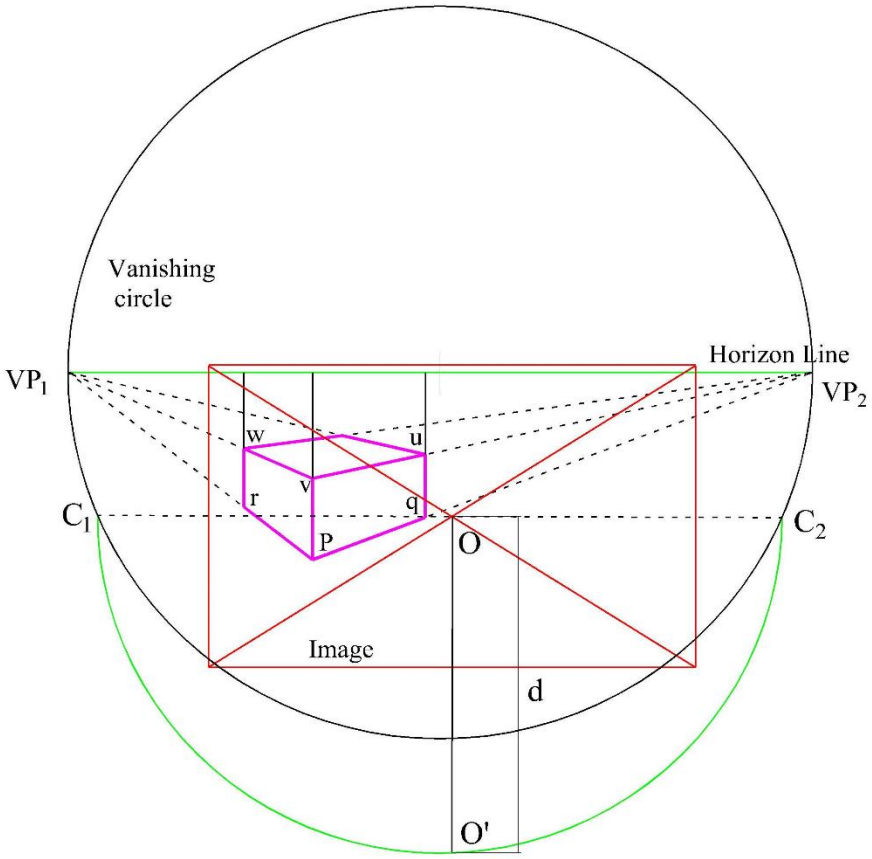


Fig. B.1: Procedure for station point determination.

The first step is to find the position of the view point on the vanishing circle. It has to be noted that the vanishing circle represents the trajectory (circle) along which viewing was done for the given configuration such that viewing from any position along the circle will result in two fixed vanishing points VP_1 and VP_2. The only difference is that the top view will change in terms of dimensions and orientation. A simple method has been discussed in (Hart, 2009) for finding the position of the view point. The method basically requires the image centre of the given image, procedure of which is outlined below:

1. Find the intersection of the diagonals of the given image; image being in rectangular or square shape. This intersection represents the image centre denoted as *O* in Fig. B.1.

2. Draw a chord (horizontal line) that meets the vanishing circle on either side, denoted by C_1C_2 in Fig. B.1. Construct a semicircle with this chord length as diameter.
3. Project O vertically until it meets this semicircle at say O'. This represents the view point of the camera (or a person viewing) with the viewing direction being perpendicular to the horizon line in top view.
4. The vertical distance d between the image centre O and this newly drawn semicircle represents the distance of the camera/viewer to be stationed perpendicular from O away from the paper (considering the paper as our image plane).

We now see the procedure for constructing top view of a single object given that we have found the view point (or projection point) O'. The horizon line VP_1–VP_2 is usually considered as the image plane in top view. Position the view point O' from the image plane such that the perpendicular distance from the image plane is d and lies vertically along the line through O, the image centre. This will usually lie on the vanishing circle introduced as O' earlier. Join O' with VP_1 and VP_2. It should be noted that $O'VP_1$ and $O'VP_2$ are perpendicular to each other. In actual, these two lines indicate the sides of the object's perpendicular surfaces and extension of these surfaces to infinity which meets at vanishing points on either side (in top view).

According to *Thales theorem* in mathematics, for any point on the circle, joining the diameter points with it will always make 90° angle. Our next step will be finding the dimensions of the object which are scaled by a scaling factor (say S_k) to the actual dimensions of the object in the real world. Dimensions are obtained by constructing top view from the given perspective image of the object. This is done in usual way as is done in arts/architecture. Figure B.2 illustrates the process.

The procedure is outlined as follows (MacEvoy, 2015):

1. Project the vertical edges of the bounding box onto the horizon line; i.e., Project the vertices r, p, q vertically until it meets horizon line at r_h, p_h and s_h respectively.
2. Join O' with vanishing points (VP_1 and VP_2). Note the angles θ_1 and θ_2 made by $O'VP_1$ and $O'VP_2$ with the horizontal at O'. Also, draw construction lines from O' to r_h and s_h. Extend the lines beyond r_h and s_h.
3. Now, with one vertex of the object at p_h, draw two lines such that they make same angles θ_1 and θ_2 with the horizon. The reason is simple, when stationed at O', object

will make θ_1 and θ_2 with the horizontal and hence the top view should also be aligned to the same angles with the image plane.

4. Mark the intersections of these two lines with the construction lines drawn earlier (from *O'* passing through r_h and s_h). Mark these intersections as r_p and s_p as shown in Fig. B.2.
5. Construct a parallelogram with $p_h r_p$ and $p_h s_p$ as sides. For the considered example in Fig. B.2, construction is shown as a rectangle (since it is a box) in red colour. The dimensions of the rectangular box (top view) can thus be obtained.

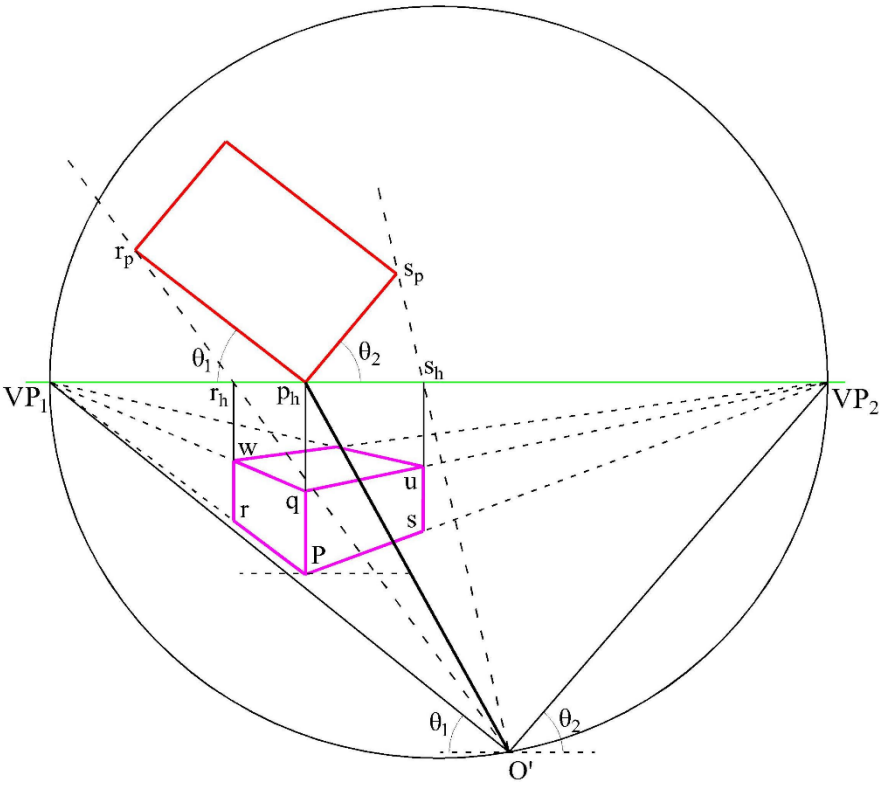


Fig. B.2: 2PP object top view construction procedure.

At this juncture, it's worth mentioning the important properties that we got so far:

1. Scaled dimensions of the object given by $p_h s_p$ and $p_h r_p$.
2. Orientations of the object w.r.t. the camera view position given by θ_1, θ_2 and θ_3.
3. Distance of the object measured between the object's nearest corner p_h to the camera position.
4. Scaled height of the object given by the measure *pq* as shown in Fig. B.2.
5. Scaled height of the camera axis from the ground plane that is measured by the distance pp_h.

PUBLICATIONS

A. Conferences

1. **Bhanu Chander V, Asokan T and Ravindran B**, "A new Multi-Bug Path Planning algorithm for robot navigation in known environments," IEEE Int. Conf. (TENCON), Singapore, 2016, pp. 3363-3367. (doi: 10.1109/TENCON.2016. 7848676)
2. **Bhanu Chander V, Asokan T and Ravindran B**, "Recovering Free Space from a Single Two-Point Perspective Image for Mobile Robot Navigation for Indoor Applications", iNaCOMM'17, 3rd Int. Conf. Machines & Mechanisms, Mumbai, 2017.

B. Journal

1. **Bhanu Chander, V., Asokan, T., Ravindran, B.**, "Geometrical Methods for 2D and 3D Map Reconstruction Using Single View Metrology of Two-Point and Three-Point Perspective Images", International Journal of Machine Graphics & Vision, Poland. (under review)