https://ntrs.nasa.gov/search.jsp?R=19980002903 2020-06-16T00:36:43+00:00Z

Detection of Obstacles in Monocular Image Sequences

Final Technical Report for NASA Co-Operative Research Agreement
Number NCC 2-916
"A Vision-Based Obstacle Detection System for Aircraft Navigation"
Period of Grant: August 1, 1995 to July 31, 1997

Submitted to
NASA Ames Research Center
Technical Officer: Albert Ahumada
Flight Management and Human Factors Division
Mail Stop 262-2
Moffett Field, CA 94035

by

Rangachar Kasturi and Octavia Camps
Principal Investigators
Departments of Computer Science and Engineering and Electrical Engineering
The Pennsylvania State University
University Park, PA 16802
Tel: (814) 863-4254 Fax: (814) 865-3176
E-Mail: {kasturi,camps}@cse.psu.edu

Graduate Students:
Sadashiva Devadiga
Tarak Gandhi

November 26, 1997

Detection of Obstacles in Monocular Image Sequences

Preface

This research was initiated as a part of the Synthetic Vision System (SVS) project for the development of a High Speed Civil Transport (HSCT) aircraft at NASA Ames Research Center. Primary goal of the research reported here is to develop image analysis algorithms to detect the runways/taxiways and obstacles in monocular image sequences obtained from two types of sensors, a Passive Millimeter Wave (PMMW) sensor and a video camera. The report is divided into two parts. The first part of this report aims at developing a model-based approach for detecting runways/taxiways and objects on the runway from a sequence of images obtained from a moving PMMW sensor. An approximate runway model and the position information of the camera provided by the Global Positioning System (GPS) is used to define the region of interest to search for image features corresponding to the runway markers. Once the runway is identified, a histogram-based thresholding is used to detect obstacles on and outside the runway. The second part addresses the problem of detection of objects in monocular image sequences obtained from an on-board video camera. A recursive motion-based segmentation algorithm based on planar motion model is used. The background motion due to camera is identified as the dominant motion and the outliers which do not follow this motion are identified as obstacles.

Abstract

The ability to detect and locate runways/taxiways and obstacles in images captured using on-board sensors is an essential first step in the automation of low-altitude flight, landing, takeoff, and taxiing phase of aircraft navigation. Automation of these functions under different weather and lighting situations, can be facilitated by using sensors of different modalities. An aircraft-based Synthetic Vision System (SVS), with sensors of different modalities mounted on-board, complements the current ground-based systems in functions such as detection and prevention of potential runway collisions, airport surface navigation, and landing and takeoff in all weather conditions.

In this report, we address the problem of detection of objects in monocular image sequences obtained from two types of sensors, a Passive Millimeter Wave (PMMW) sensor and a video camera mounted on-board a landing aircraft. Since the sensors differ in their spatial resolution, and the quality of the images obtained using these sensors is not the same, different approaches are used for detecting obstacles depending on the sensor type. These approaches are described separately in two parts of this report.

The goal of the first part of the report is to develop a method for detecting runways/taxiways and objects on the runway in a sequence of images obtained from a moving PMMW sensor. Since the sensor resolution is low and the image quality is very poor, we propose a model-based approach for detecting runways/taxiways. We use the approximate runway model and the position information of the camera provided by the Global Positioning System (GPS) to define regions of interest in the image plane to search for the image features corresponding to the runway markers. Once the runway region is identified, we use histogram-based thresholding to detect obstacles on the runway and regions outside the runway. This algorithm is tested using image sequences simulated from a single real PMMW image.

The camera position information provided by the GPS is not accurate. An alternative is to estimate the camera position using image-based features, such as points and lines. The accuracy of such estimates, however, depends on the resolution of the sensor and the position of the sensor in 3-D. We propose an analytical model to compute the accuracy of the estimated camera position and develop equations for accuracy in terms of sensor resolution and the sensor position in 3-D. Analytical results obtained using this model for three different sensors at six different positions of the sensor are presented. We also propose a new algorithmic approach for computing the error in the estimated camera pose due to image plane quantization. Such theoretical analysis is useful in deciding the required camera resolution for use in the design of SVS for navigation.

The second part of this report addresses the problem of detection of objects in monocular image sequences obtained from a video camera mounted on-board a landing aircraft. First, the optical flow is computed at corner-like feature points detected on the image using a corner detector. We assume that the runway is either planar or is piecewise planar and propose a recursive motion-based segmentation algorithm based on the planar motion model for segmenting flow vectors into separate regions. Model parameters are recovered for each of the region. The background motion due to camera is identified as the dominant motion and the outliers are identified as due to the obstacles. Various stages of the algorithm were tested using both synthetic and real image sequences obtained from actual flight test.

Line features are considered to be prominent and more reliable than point features. In this report we have explored the use of line features for two purposes: (1) estimating the runway plane parameter using runway markers, and (2) tracking and estimating the position and velocity of the ground-based obstacles using object lines. The algorithm was tested using real image sequences. Since the angle between the projecting planes (i.e., the plane containing the runway marker and the corresponding line segment in the image plane) for a given line segment in two frames was too small to resolve, good estimate for the plane parameter could not be obtained. Though line features were tracked, end points could not be properly located and hence good estimate for the motion of the object could not be obtained.

Table of Contents

1 Introduction

The ability to detect and locate runways/taxiways and obstacles in images captured using on-board sensors is an essential first step in the automation of low altitude flight, landing, takeoff and taxiing phase of navigation of aircrafts. At present the autopilot system navigates the aircraft up to a certain height (depending on the autopilot system installed on the aircraft and the capability of the ground systems such as Instrument Landing System (ILS), Microwave Landing System (MLS) etc.) above the runway during landing after which the human pilot assumes control of the aircraft. At this point, the pilot makes decisions about landing depending on the external visibility. The landing is often aborted or delayed if the visibility is poor. Crew members rely on a sophisticated combination and fusion of information from multiple sensors in order to accomplish their mission under conditions of darkness, reduced visibility and bad weather. Hence, there is a strong need for an aircraft-based Synthetic Vision System (SVS) to complement the ground-based systems in various functions such as detection and prevention of potential runway collisions, airport surface navigation, and landing and takeoff in all weather conditions [17, 107].

Implementing an SVS would upgrade the capabilities of over 1100 US airports to CAT IIIa[1] or better, which are currently equipped with CAT II[2] or CAT I[3] capability [24, 107].

Another motivating factor for the development of a SVS is the design of a supersonic

[1]The minimum vertical visibility ceiling is 12-35 feet with runway visual range of 100-300 meters. A fully automatic landing system with automatic flare and a failure survival autopilot system with a probability of catastrophic failure less than 10^{-7} per hour is required. Pilot assumes control at touchdown.

[2]The minimum required vertical visibility ceiling is 100 feet and runway visual range is 400 meters. A fail passive autopilot is required. Pilot takes over landing at a height of 100 feet.

[3]There is sufficient vertical visibility at a height of 200 feet with a runway visual range of at least 800 meters for the pilot to carry out a safe landing under manual control. A simplex autopilot system is acceptable. Pilot assumes control at a height of 200 feet.

commercial aircraft. It is argued that a High Speed Civil Transport (HSCT) without a Concorde-like drooped nose could result in significant reduction in the gross takeoff weight[107, 116]. Without a drooped nose, the cockpit has limited external visibility and the pilot has to depend on an SVS which derives information from combination of sensors.

The SVS is envisioned to be equipped with a Passive Milli-Meter Wave (PMMW) sensor in addition to the infra-red and video cameras. A PMMW sensor is designed to be operated at lower frequencies (e.g., 94 GHz), at which point the energy attenuation due to fog is known to be at a minimum, thus providing the ability to see through fog [132]. Outputs of these sensors will be integrated with aircraft position and orientation information (yaw, pitch, and roll) provided by the Global Positioning System (GPS) and the Inertial Navigation System (INS), and an airport geometry database to detect the runways/taxiways and obstacles, locate the aircraft within the airport, determine potential conflicts, issue advisories, and sound cockpit alarms.

In this research, we address the problem of detection of ground-based obstacles in monocular image sequences obtained from on-board sensors for use in an SVS. Any object in the sensor's field of view is considered to be an obstacle. Most of the algorithms available in the literature for obstacle detection are developed and evaluated for applications in robotics, road vehicle navigation, and intelligent vehicle and highway systems (IVHS). These applications use high resolution and high quality images obtained using a video camera. Stereo images are often used to estimate the distance to an obstacle. PMMW sensor images, however, are of low resolution and poor quality. The use of stereo is not currently recommended for SVS due to the difficulty with having such a system with a large enough base distance to estimate the 3D position accurately. Hence, there is a strong need to design and evaluate computer vision algorithms for use in SVS for obstacle detection using monocular image sequences.

1.1 Overview

This report is about detection of objects in monocular image sequences obtained from a moving sensor. The initial goal of this research was to design a computer vision system to analyze image sequences obtained from a Passive Millimeter Wave imaging system mounted on-board the aircraft to detect runways/taxiways and objects on the runway. Although PMMW sensors have good response in fog, their spatial resolution and radiometric contrast are very low, and the quality of the images obtained using these sensors is poor. However, we have additional data in the form of the airport model and approximate position and orientation of aircraft, to guide our system to locate objects in the low resolution image.

The position of the aircraft is available from the GPS, and the orientation of the aircraft is provided by the INS. The data from these instruments are known only up to a certain accuracy. For example, GPS data are updated once every second and it is likely that a few such updates could be missed, potentially causing camera position data to be as much as a few hundred feet off. Knowledge of the camera motion information from the GPS/INS is the key to detecting objects and estimating their position and velocity. Any inaccuracy in this information could result in an error in the estimated position and velocity of the objects.

An alternate approach to obtain an improved estimate of camera position is to use objects with known world coordinates and their position in the image plane (e.g., intersections of runway/taxiways, corners of buildings, etc.). This requires an analytical study of the relationship among the camera parameters, the resolution of the images, and the distances between the aircraft and objects. We propose an analytical model to compute the accuracy of the estimated camera position and develop equations for accuracy in terms of sensor resolution and sensor position in 3-D. We also propose a new algorithmic approach for computing error in the estimated camera pose due to plane quantization. Such theoretical analysis are useful in deciding the required camera resolution for use in the design of SVS for navigation.

Initial experiments were conducted using a test image obtained from a single pixel camera located at a fixed point in space and then mechanically scanning it to obtain a 50×150 pixel image. Using the camera position information and the airport model, a sequence of 30 frames was simulated for this experiment. Although results obtained using our algorithms on these images were encouraging, further research using PMMW sensor could not be continued for the following reasons:

- A practical camera with an array of pixels could not be developed,
- It is not clear at this time whether such a sensor could be safely mounted on-board an aircraft, and
- Camera pose estimation from image-based features using low resolution PMMW sensor images was not better than the position information provided by the GPS.

Hence, the remaining part of this research is focused on detecting independently moving objects in monocular images obtained by a moving video camera. In this work, we assume that the objects were moving on a background which is either planar or is piecewise planar. We describe a new recursive motion-based segmentation algorithm for segmenting images into regions corresponding to independently moving objects and estimating their motion parameters.

Apparent motion of brightness patterns in images, called the optical flow, is computed first. Since the camera is moving, background (i.e., the runway) appears to be moving in the image. Using knowledge of camera motion, flow vectors resulting from the runway are grouped based on the hypothesis that they are resulting from a single planar surface in motion Planar surface motion model parameters for the runway is computed using a least square model fitting approach. Flow vectors violating the planar motion model are detected as outliers and are identified as obstacles.

Line features are considered to be prominent and more reliable than point features. Two types of line features are detected in image sequences obtained from a camera mounted onboard a landing aircraft - - line features due to the runway markers and line features due to the 3-D objects in the scene. Runway plane parameters are computed by matching at least two line features in two frames. Using the known camera motion information and the estimated plane parameters, line features in the image corresponding to the 3-D object in motion are tracked. A recursive Kalman filter-based approach is used to track the line features and estimate the position and velocity of the object in 3-D assuming that the object is moving on the planar runway.

This research addresses many difficult problems in computer vision, such as motion-based segmentation, feature tracking, dynamic scene analysis, etc. In the past, researchers solved the problem of motion-based segmentation using the Hough method, image registration using digital warping and other statistical methods. The digital warping approach requires knowledge of the warping parameters. It is mainly used to compensate for the dominant motion; all features violating the dominant motion are detected as outliers or obstacles. This results in warping noise in addition to the digitization noise. The Hough method can be used to segment and estimate multiple motions simultaneously, although it is computationally expensive and its success and accuracy are mainly dependent on knowledge of the range and resolution of the parameter space.

Unlike the Hough-based approach, our approach does not require prior knowledge of the range of parameter values for the motion model. Instead, parameter values are directly estimated. This estimation is refined by removing the outliers at every iteration. Using our algorithm, basic segmentation can be done without knowledge of the motion parameters, as opposed to the warping technique, which requires the camera motion parameters to compute the warping matrix. To deal with the computation of multiple motions, we propose incorporation of split and merge technique to the motion-based segmentation algorithm.

The algorithms developed in this work have been tested using both real and synthetic images. The synthetic images were generated by the program developed for this report, as well as a simulation software developed at the NASA Ames Research Center. Several real image sequences used in this work were obtained by sensors mounted on-board a landing aircraft, and were provided by the NASA Ames Research Center and NASA Langley Research Center.

1.2 Review of the Literature on Obstacle Detection Systems for Navigation

In recent years, considerable effort has been put into evaluating the feasibility of computer vision algorithms for these tasks. This section gives a brief overview of research work done at NASA and other institutions in the area of obstacle detection for navigation.

The detection of runway lines has been attempted in [55, 77] using properties specific to the runway (e.g., the use of parallel lines). Work described in [30] is based on a model of the runway where the camera image is compared with the expected runway image obtained by generating an image using the known position of the camera and the airport runway model database. On-board INS and GPS can provide reasonably good position estimation during flight, but they are not considered accurate enough to permit automatic landing [101]. Image-based features such as points, lines, etc., are used to determine the position of a sensor in [4, 61, 70]. A number of authors have attempted to determine the position of a sensor from the captured image [4, 61, 69].

Most approaches have been based on some model of environment that is expected by the sensor [5, 60, 73]. The position of the sensor is computed from the actual sensor input and some perceived difference between the actual and the expected sensor input. The choice of a reasonable model for generating the runway scene, as well as the choice of a cost function defining the goodness of fit, are crucial to the accuracy and speed of convergence of such an algorithm. In [102] three different models - - namely edge-based model, area-based model, and texture-based model - - are used to compare the viewed scene with an expected scene. A cost function quantifying the matching of the expected image with the camera image is minimized to obtain more accurate camera position.

In principle, the problem of obstacle detection with a moving camera can be completely solved if either the range map is supplied for all points in front of the camera, or the image acquired using the sensor is segmented and recognized into constituent parts (i.e., sky, runways, buildings, etc.). A Kalman filter-based recursive estimation procedure for estimating range to feature points in the image is explained in [95, 108, 110, 111]. Image regions of size 11 $\times$ 11 pixels with high variance are detected as features. The initial range of these features is computed using motion stereo. The feature positions in the subsequent frames are predicted, and the estimates of ranges to the feature points are refined using a Kalman filter. The algorithm is tested and the results are reported for both indoor images and images acquired from the actual flight test. In this work, all obstacles are assumed to be stationary. The algorithm was modified to handle stereo images in [97]. Motion-based and stereo-based algorithms for passive range estimation are compared in [109].

The range estimation procedure described in [118] uses a simple incremental weighted least squares method for estimating the position of stationary objects using known camera state parameters. This algorithm extends the epipolar plane image analysis described in [14, 8] to general camera motion by assuming the camera motion to be piecewise linear. In [120] an optical flow-based approach is used for detecting independently moving objects. Image regions corresponding to the independently moving objects are segmented from the background by applying the constraint filtering originally proposed by Nelson [87] on the optical flow computed from the initial few frames of the sequence. These detected regions are tracked over subsequent frames using a model-based tracking algorithm described in [56]. Position and velocity of the moving objects in the world coordinate is estimated using an extended Kalman filter.

The obstacle detection algorithm described in [115, 44] assumes that the obstacle-free runway is a planar surface and computes the residual flow by compensating for the camera motion using image warping. The algorithm is tested using images obtained during a helicopter flight.

1.3 Organization

This section provided a brief introduction to the research problem and objectives addressed in this report. Even though this report is based on the main theme of object detection in monocular image sequences, the total research dealt with processing images obtained from two different sensors. Detecting obstacles in images obtained from sensors of different modalities require different processing techniques. The next two sections explain our work using PMMW sensor images. Subsequent sections describe the work done using video image sequences.

Section 2 describes the analytical model for computing the error in the position and orientation of the camera estimated using image-based features. An analytical model is described for an on-board sensor in terms of its position and orientation, and other sensor internal parameters. Equations for error in camera parameters are also derived. Three sensors - - the Passive Millimeter Wave (PMMW) sensor, the Forward Looking Infrared (FLIR) sensor, and the High Definition Television (HDTV) sensor - - are evaluated and quantitative results are presented.

A model-based approach for detecting objects in a PMMW sensor image is described in Section 3. Results were obtained using an image sequence obtained from a single pixel camera provided by the NASA Langley Research Center. Since a practical PMMW sensor array was not completely developed by NASA's commercial partner, we could not continue using this sensor modality. The remaining sections of this report use video images.

Section 4 describes our motion-based segmentation algorithm for segmenting images into regions corresponding to different motions and estimating their motion parameters.

An algorithm for computing the optical flow from image sequences is described. A planar motion model and a linear algorithm for recovering the model parameters from optical flow vectors are presented. A recursive algorithm for estimating single and multiple independent motions is described. The results obtained using both synthetic and real images are presented.

Algorithm for estimating the plane parameters using line features is described in Section 5. Details are given for a Kalman filter-based approach for tracking line features in the image sequence and estimating the position and velocity of 3-D objects moving on a planar runway. The feasibility of these methods was tested using real image sequences; the results are presented in this section. Section 6 contains a summary of the entire research and suggests future research topics.

2 Sensor Sensitivity Evaluation and Calibration

A Synthetic Vision System for enhancing the pilot's ability to navigate and control the aircraft during landing and taxiing operations uses an on-board airport database and images acquired by external sensors such as millimeter wave, infrared and low light TV cameras. Additional navigation information needed by the system is provided by the Inertial Navigation System (INS) and Global Positioning System (GPS). The data from these navigation instruments are known only to a certain accuracy (depending upon the type of the instruments used), and are updated once every second. Since data from on-board instruments are more accurate than the GPS-based data, they are useful for obtaining more accurate camera position. An alternative approach is to use the 3-D location information of certain landmarks such as intersection of runways/taxiways, corners of buildings and their corresponding positions in the image to obtain a better estimate of the camera position. This requires an analytical study of the relationship among the camera positional parameters (i.e. position and orientation in 3-D), the sensor characteristics, and the relative distance between the sensor and objects.

2.1 Sensor Positional Sensitivity Evaluation

Imaging is a process of mapping a large 3-D scene onto a small 2-D plane. This 2-D image plane is quantized in both spatial directions to facilitate image processing by digital computers. The limited resolution of the sensor requires that the entire scene be mapped to an $M \times N$ array of points called pixels. Such an array being too small to adequately represent the scene introduces significant amount of error into computations involving the locations of image points and features. Another natural outcome of the imaging process is the loss of depth information resulting from mapping of an infinite number of points on a line-of-sight onto a single point in the image plane. As a result, reconstruction of a scene by computing the depth of scene points using image-based features is a challenging problem. The inverse process, known as the camera calibration problem, is to compute the camera position by triangulation between known scene points in the world and their corresponding positions in the image plane. Nearly all of the methods for solving this problem require only one monocular image with special marks which can be man-made point features or line features [42, 46, 54, 64, 69, 113]. The accuracy of such computational approaches, however, depends upon the camera geometry and the quantization of the image plane.

Most analysis for obtaining three-dimensional position information of the scene points are based on the triangulation system [13, 76]. To compute the depth to a scene point, triangulation must be performed using its projection onto two images, captured either simultaneously, by two cameras separated by a base distance (as in the case of *binocular stereo*), or separately, by a single camera at two different positions (as in the case of *motion stereo*). This requires computation of features in one image and a correspondence in the other image. Assuming correct matches have already been found, the next step is to recover the 3-D information using triangulation. The accuracy of such a computation depends on the spatial quantization of the image plane.

Equations for calculating error in distance measurement between an object and a stereo vision system established in [76] present a worst case error analysis, and show that the percentage error in distance measurement is inversely proportional to the number of pixels occupying the shift between the two images and directly proportional to the object distance. Blostien and Huang [13] have developed equations to determine the probability that a certain estimate is within a specified tolerance given the camera geometry of a stereo setup. Error equations developed in [31, 32] for determining the optimum line width for visual navigation of an autonomous mobile robot give the percentage of error for any sensor geometry, line width, and error conditions. Even though their analysis looks somewhat similar to our work, the basis for their analysis was measurement of line width.

In this research, we estimated the aircraft position by tracking known static scene points in the image plane. Since the accuracy of this estimation depends on various sensor parameters, we wanted to develop an analytical model that could relate the accuracy of the estimated sensor position to the sensor positional parameters and attributes of the captured image. The sensor positional parameters include range, cross range, altitude and pitch, roll, and yaw angles. Sensor imaging attributes include the number of pixels in the image and the optical angular view (measured in degrees).

We assume a pin-hole camera and, therefore, ignore camera lens distortion and optical non-linearities. We also assume that the problem of feature correspondence was solved using some correspondence algorithm and that the correspondence was accurate to a single pixel. Hence, the analysis does not depend on the type of algorithm used for establishing feature correspondence. Any error in computation was basically due to spatial quantization of the image plane. In this analysis, we projected the image plane onto the ground, and the pixel (p, q) in the image plane was modeled as a patch on the ground plane (ground area represented by the pixel). We defined the sensitivity or the error in computation as the minimum change in a camera parameter that would move a fixed ground point to the next pixel in the image plane.

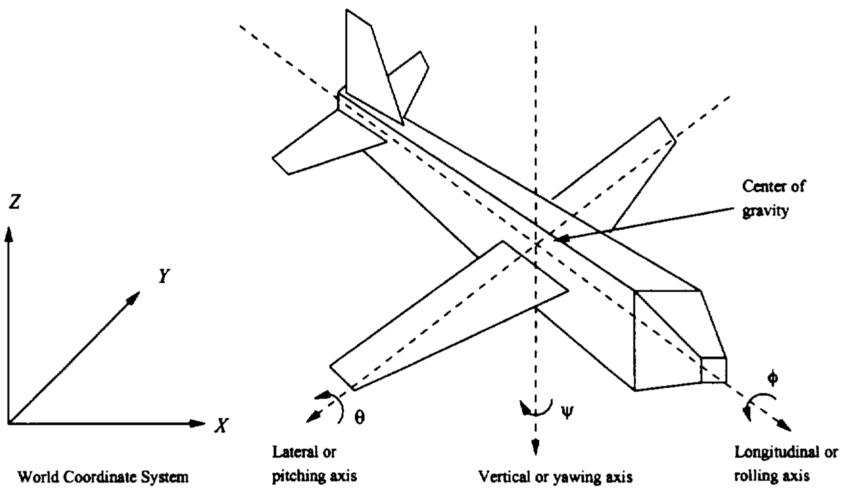


Figure 1: Airplane body axes

2.1.1 Imaging Geometry

Throughout the analysis, for convenience, we assume that the sensor is located at the aircraft's center of gravity. Hence, we can use the terms sensor position and aircraft position interchangeably. We also neglect the effect of curvature of the earth.

The system of reference axis that forms the basis of the system of notations used to describe the position of the sensor is shown in Fig. 1. The figure shows an aircraft with three mutually perpendicular axes (pitch, roll, and yaw) passing through the aircraft's center of gravity. The position of the aircraft is defined with respect to the three mutually perpendicular world coordinate axes X, Y, and Z. The image plane is assumed to be perpendicular to the rolling axis, with its vertical and horizontal axes coinciding with the yawing and the pitching axes of the airplane, respectively.

Fig. 2 shows an imaging situation during landing, where the aircraft is at (X_c, Y_c, Z_c), with pitch angle θ, zero yaw, and zero roll angle. Let $\alpha = 90 - \theta$. The field of view of the camera is determined by two viewing angles: $\Delta\alpha$ defined in the same plane as θ, and $\Delta\beta$ at right angles to $\Delta\alpha$. ($\Delta\alpha$ determines the vertical extent of the image and $\Delta\beta$ its horizontal extent.) Even though the image obtained by the sensor is always a rectangle, the ground area captured by the sensor is a trapezoid $ABCD$ whose side length and area depend on $\Delta\alpha$, $\Delta\beta$, and various other sensor parameters like position, orientation, etc. Note that a pixel in the image plane corresponds to a patch on the ground plane. We refer to this as a pixel-patch (See Fig. 3).

2.1.2 Sensitivity Analysis

Consider a point feature which has been detected at some pixel (p, q). Let the actual world coordinates of this feature be $(P, Q, 0)$. Since a pixel represents a patch on the ground, the camera could change in its position by a certain amount while still retaining the image of the feature at the same pixel (p, q). Hence, a camera pose estimation by passive triangulation will always give the same camera pose for nearby camera positions, unless the change in the camera position is large enough for the feature to be observed in the neighboring pixel. We define this minimum change in camera displacement as the sensitivity of the camera. Note that this is a measure of accuracy of camera position estimate and is a function of the camera position, image size in number of pixels, angular resolution, and the pixel location (p, q) in the image plane.

Let N_x and N_y represent the number of pixels in the vertical and horizontal directions, respectively. The pixels are numbered $-\frac{N_x}{2}, \ldots 0, \ldots \frac{N_x}{2} - 1$ in the vertical direction and $-\frac{N_y}{2}, \ldots 0, \ldots \frac{N_y}{2} - 1$ in the horizontal direction. The rolling axis of the plane is assumed to pass through the bottom right corner of the patch on the ground plane, which corresponds

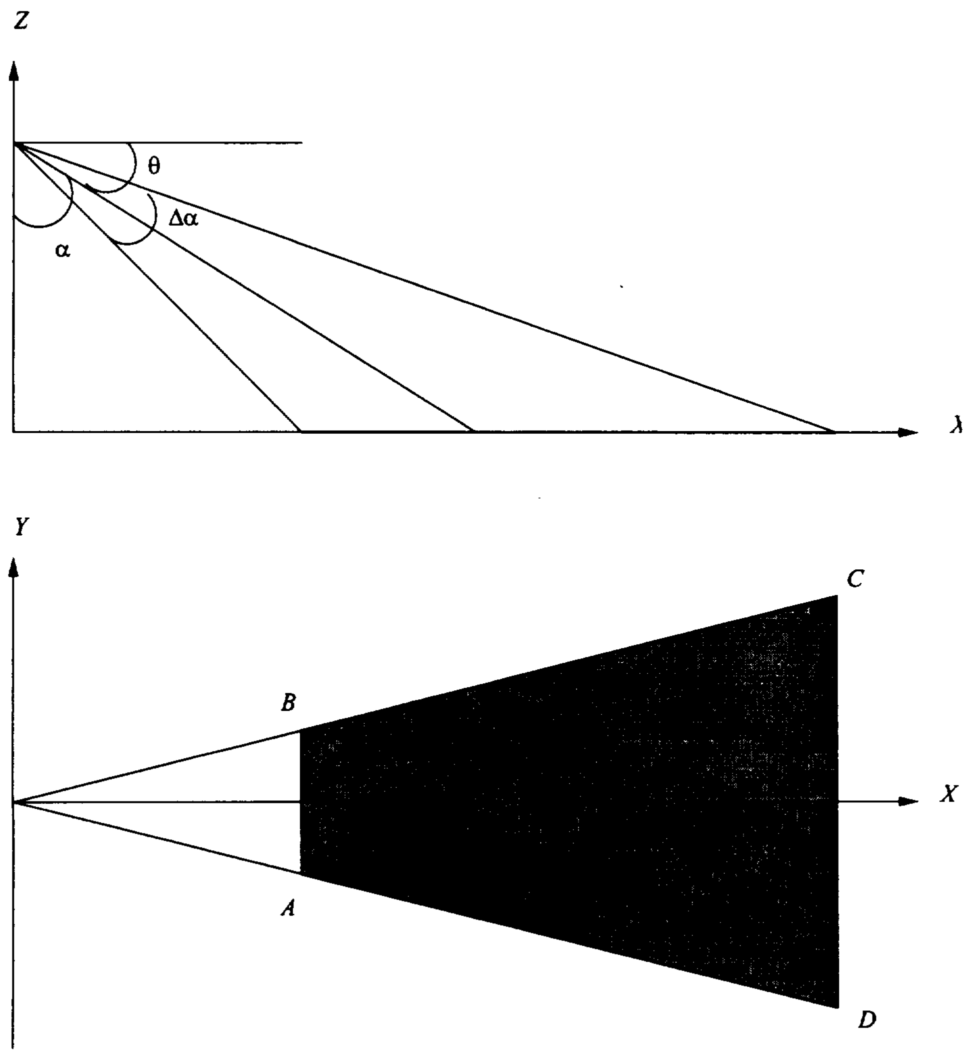


Figure 2: Image obtained by the sensor is projected towards the ground

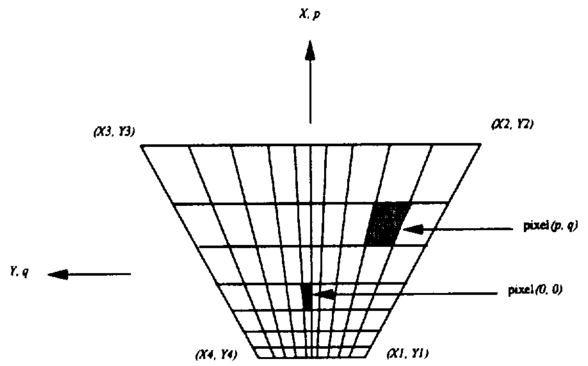


Figure 3: Ground area covered by the sensor. Each small trapezoid corresponds to a pixel in the actual image

to the center pixel in the image plane. Other pixels are referenced in a similar manner. The coordinates of the reference corner of the ground area covered by a pixel (p, q) can be estimated by the following relations:

$$\begin{aligned} X &= X_c + Z_c \tan(\alpha + p\frac{\Delta\alpha}{N_x}) \\ Y &= Y_c + \frac{Z_c}{\cos(\alpha + p\frac{\Delta\alpha}{N_x})} \tan(q\frac{\Delta\beta}{N_y}) \end{aligned} \tag{1}$$

For a non-zero rolling angle ϕ, the ground coordinates (X', Y') which correspond to a pixel (p, q) in the image plane are obtained with replacing (p, q) in the equation by (p', q'), where

$$\begin{aligned} p' &= p\cos\phi - q\sin\phi \\ q' &= p\sin\phi + q\cos\phi \end{aligned} \tag{2}$$

Since a pixel-patch is referenced by the bottom right corner of the pixel, the other three corners become the reference of its three neighboring pixel patches, as shown in Fig. 4.

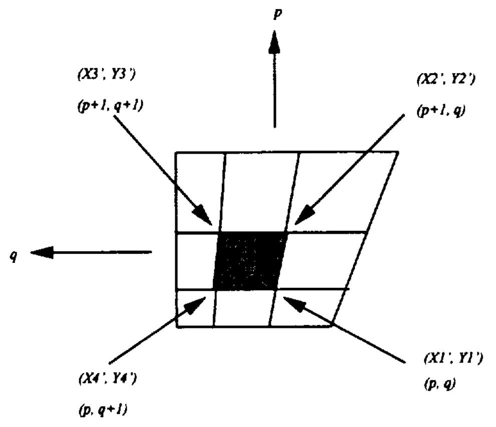


Figure 4: A pixel (p, q) projected towards the ground

Thus, the four corners of this pixel-patch (X_i', Y_i'), $i = 1, 2, 3, 4$ are obtained by using Eq. 1, where (p, q) are replaced by (p', q'), such that

$$\begin{aligned} p_i' &= p_i \cos\phi - q_i \sin\phi \\ q_i' &= p_i \sin\phi + q_i \cos\phi \end{aligned} \tag{3}$$

where $(p_1, q_1) = (p, q), (p_2, q_2) = (p+1, q), (p_3, q_3) = (p+1, q+1)$, and $(p_4, q_4) = (p, q+1)$.

Eq. 1 gives the relationship between the camera parameters $(X_c, Y_c, Z_c, \theta, \phi)$ and the ground point corresponding to a pixel (p, q). We are now interested in computing the sensitivity of the imagery sensor. This is defined as the minimum change in a camera parameter that would move a fixed ground point to the next pixel in the image plane. We obtain this by taking the partial derivative of X_1' and Y_1' with respect to the corresponding parameter. For example,

$$D_{X_c}^X = \frac{\partial X_1'}{\partial X_c},\ D_{X_c}^Y = \frac{\partial Y_1'}{\partial X_c} \tag{4}$$

This derivation is an approximation to the amount of change in X_1' for unit change in X_c. Thus, we estimate that the amount of change in X_c in order to change X_1' to X_2' or Y_1' to Y_2' (which define the corners of adjacent pixels) as

$$S_{X_c}^X = \frac{(X_2' - X_1')}{D_{X_c}^X}, \; S_{X_c}^Y = \frac{(Y_4' - Y_1')}{D_{X_c}^Y} \tag{5}$$

Note that $S_{X_c}^Y = \infty$, as expected. Sensitivity with reference to other parameters is defined in a similar manner. These are summarized in Table 1. In general, S_j^i stands for sensitivity in the direction i due to the sensor positional parameter j computed at pixel (p, q) in the image plane.

Sensor sensitivity is a function of various sensor parameters and sensor attitudes. Since the sensor plane is inclined to the ground plane, the sensitivity varies in the vertical and horizontal directions, along the sensor plane and, hence, is a function of pixel number (p, q). Equivalently, the accuracy of estimation of sensor position using ground truth data is a function of pixel position as well as other parameters. For a given range, the estimation using features that are observed at the top half of the sensor are less accurate because of the large ground area represented by these pixels. Also, for a given p, the accuracy decreases as we move towards the border of the sensor in the horizontal direction. In summary, the accuracy of estimation is a function of sensor characteristic and the ratio of the sensor view angle to the number of pixels in the image.

SPP		**Sensor Sensitivity at** *(p, q)*	**Sensitivity at** *(0, 0)* $\phi = 0$
X_c	$S^X_{X_c}$	$\frac{2Z_c \sin(\cos\phi \frac{\Delta\alpha}{N_x})}{\cos(2\alpha + \frac{\Delta\alpha}{N_x}((2p+1)\cos\phi - 2q\sin\phi)) + 1}$	$2Z_c \frac{\sin(\frac{\Delta\alpha}{N_x})}{\cos(2\alpha + \frac{\Delta\alpha}{N_x}) + 1}$
	$S^Y_{X_c}$	∞	∞
Y_c	$S^X_{Y_c}$	∞	∞
	$S^Y_{Y_c}$	$Z_c \frac{\tan(\frac{q'_4 \Delta\beta}{N_y})}{\cos(\alpha + p'_4 \frac{\Delta\alpha}{N_x})} - Z_c \frac{\tan(\frac{q'_1 \Delta\beta}{N_y})}{\cos(\alpha + p'_1 \frac{\Delta\alpha}{N_x})}$	$2Z_c \frac{\sin(\frac{Delta\beta}{N_y})}{\cos\alpha \cos(\frac{\Delta\beta}{N_y}) + 1}$
Z_c	$S^X_{Z_c}$	$\frac{S^X_{X_c}}{\tan(\alpha + p'_1 \frac{\Delta\alpha}{N_x})}$	$2Z_c \frac{\sin(\frac{\Delta\alpha}{N_x})}{\sin(2\alpha + \frac{\Delta\alpha}{N_x})}$
	$S^Y_{Z_c}$	$S^Y_{Y_c} \frac{\cos(\alpha + p'_1 \frac{\Delta\alpha}{N_x})}{\tan(\frac{q'_1 \Delta\beta}{N_y})}$	∞
θ	S^X_θ	$S^X_{X_c} \cos^2(\alpha + p'_1 \frac{\Delta\alpha}{N_x}) \frac{1}{Z_c}$	$\frac{\sin(\frac{\Delta\alpha}{N_x}) \cos(\alpha + \frac{\Delta\alpha}{N_x})}{\cos\alpha}$
	S^Y_θ	$S^Y_{Y_c} \frac{\cos^2(\alpha + p'_1 \frac{\Delta\alpha}{N_x})}{Z_c \tan(q'_1 \frac{\Delta\beta}{N_y}) \sin(\alpha + p'_1 \frac{\Delta\alpha}{N_x})}$	∞
ϕ	S^X_ϕ	$S^X_{X_c} \frac{\cos^2(\alpha + p'_1 \frac{\Delta\alpha}{N_x})}{(Z_c \frac{\Delta\alpha}{N_x})(-p\sin\phi - q\cos\phi)}$	∞
	S^Y_ϕ	$S^Y_{Y_c} \frac{1}{Z_c(A\frac{\partial B}{\partial\phi} + B\frac{\partial A}{\partial\phi})}$	∞

$A = \frac{1}{\cos(\alpha + p'_1 \frac{\Delta\alpha}{N_x})}$; $B = \tan(q'_1 \frac{\Delta\beta}{N_y})$

$\frac{\partial A}{\partial\phi} = \frac{\Delta\alpha}{N_x} \tan(\alpha + p'_1 \frac{\Delta\alpha}{N_x})(-p\sin\phi - q\cos\phi)\cos(\alpha + p'_1 \frac{\Delta\alpha}{N_x})$

$\frac{\partial B}{\partial\phi} = \frac{\Delta\beta}{N_y}(p\cos\phi - q\sin\phi)\cos^2(q'_1 \frac{\Delta\beta}{N_y})$

$p'_1 = p_1\cos\phi - q_1\sin\phi$; $p'_4 = p_4\cos\phi - q_4\sin\phi$

$q'_1 = p_1\sin\phi + q_1\cos\phi$; $q'_4 = p_4\sin\phi + q_4\cos\phi$

$\alpha = 90 + \theta$; $(p_1, q_1) = (p, q)$; $(p_4, q_4) = (p, q + 1)$

Table 1: Sensor positional sensitivity equations

2.1.3 Quantitative Results and Discussion

The sensitivity analysis described in the previous section was applied to three different sensors at six different positions. Characteristics of the sensors are given in Table 2. Sensitivities $S^X_{X_c}$, $S^X_{Y_c}$, and $S^X_{Z_c}$, at the aim point (i.e., $p = 0, q = 0$) for various sensor positions are plotted in Figures 5, 6, and 7, respectively. In all of the above cases, pitch angle is $-3.0°$, roll angle is $0°$, and cross range is 0 feet (typical values when an aircraft is approaching the runway for landing). Note that $S^X_{Z_c}$ is larger than $S^Y_{Z_c}$ at (0, 0) and, hence, a feature would move to the next horizontal pixel before it moves to the next vertical pixel. Thus, only $S^X_{Z_c}$ is important.

As expected, the sensitivity is best for the sensor with the highest pixel resolution. Sensitivity also improves as the sensor is moved closer to the ground. It becomes poor for the features that are located at the far end of the vertical axis (top of the sensor i.e., for the objects that are located at the far end of the runway). As expected, the position and velocity of the aircraft can be computed more accurately by knowing the position of stationary objects on the ground that are closer to the aircraft.

The above results indicate that the accuracy of camera state estimation would be no better than the GPS unless a high resolution sensor is employed. Note that these results do not consider potential improvements that can be obtained by motion stereo techniques using a large number of image frames, or by using more feature points than the required minimum. In addition to being useful in sensor design, this analysis will also help us evaluate the accuracy of camera state estimation by any algorithm that uses image-based features that correspond to known scene points.

Sensor Positional Parameter			**Sensor Characteristic**		
Location	*Range in ft.*	*Altitude in ft.*	*Sensor Type*	*Pixel* $(H \times V)$	*Field of View* $(H \times V) deg$
Threshold	0.0	50.0	HDTV	1920×1035	30×24
CAT II-DH	908.1	100.0	FLIR	512×512	28×21
CAT I-DH	2816.2	200.0	MMW	80×64	27×22
Middle Marker	4500.0	288.2			
1000' Altitude	18081.1	1000.0			
Outer Marker	29040.1	1574.3			

Table 2: Sensor positional parameters and sensor characteristics

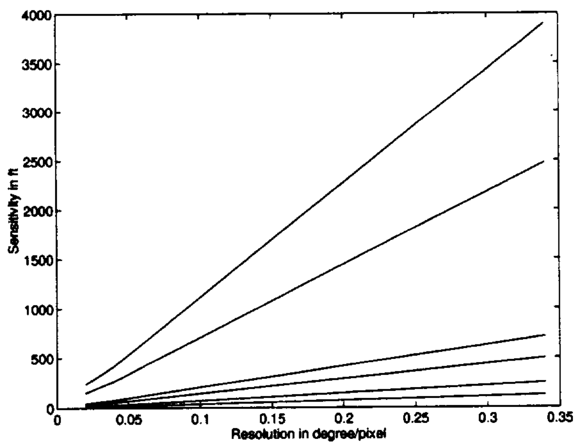


Figure 5: Sensitivity in the direction of range

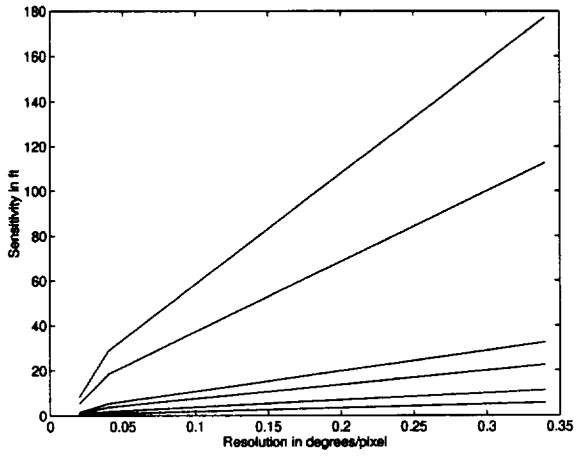


Figure 6: Sensitivity in the direction of cross range

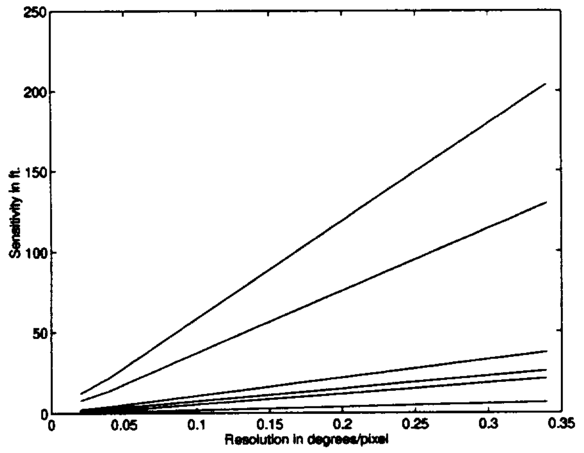


Figure 7: Sensitivity in the direction of altitude

2.2 Camera Calibration from Image-based Features

Camera calibration is the problem of determining camera parameters – including location and orientation – from images. Camera parameters can be grouped into two categories: *extrinsic parameters*, which provide information regarding the camera position and orientation with respect to a reference world coordinate system, and *intrinsic parameters*, which include focal length and scale factors in units of pixels in the image plane.

A number of methods for estimating the camera pose are available in the literature. They can be divided into two classes:

- The most common approach to estimating the camera pose has been to use point and line correspondences. These point or line features arise either from certain man-made objects placed in the scene for the purpose of calibration or from certain landmarks that already exist in the 3D scene. Estimation is done either by applying triangulation to known scene points and their corresponding image points, or by developing and solving a set of equations that relate the scene points, the corresponding image points, and the camera parameters. Earlier research on this approach was done by Wolf [130], Fischler and Bolles [39], and Ganapathy [43]. Liu et al. [70] showed that the computation of the rotation and translation vector of the camera are separable. Chou and Tsai [23] used house corners as marks and Haralick [46] used the corners of rectangles of unknown size to determine the camera view angle parameters. Chen et al.[21] proposed using a cube as a calibration object, whereas a method that uses rectangular parallelepiped with known dimensions as the calibration object is proposed in [113]. For calibration of a camera mounted on an autonomous land vehicle running on an outdoor road, Liu and Deng [66] used road boundaries as calibration objects. Fukuchi [42] used special man-made marks with certain constraints for determining the position of a robot using standard shaped marks.

- Objects with curves have been used for camera calibration, such as a plane with conic or polygon arcs as in [48] and semi-circle in [65]. Magee and Aggarwal proposed a method that uses a calibrated sphere [75].

All of the above procedures involve two steps. The first step is to locate feature points on the image plane that correspond to the known 3-D points, and the second step is to formulate and solve a set of equations that relate the scene points and the image points, thereby satisfying certain constraints. These computational approaches assume an ideal pinhole camera and model the pixels as points of insignificant dimension, largely ignoring the error introduced by the image plane quantization. More accurate calibration of the camera, or a priori knowledge about the error introduced by the image plane quantization, is quite critical in various aspects of computer vision like object recognition, scene reconstruction, robot navigation, etc. A brief review of some of the past work in error analysis of triangulation to image plane quantization is given in the next section.

2.2.1 Error due to Image Plane Quantization

Quantization of the image plane as well as the intensity levels have significant impact on the outcome of various computational approaches to solving computer vision problems. The problems are particularly important when the images are captured using low resolution sensors. The impact of quantization error on computer vision was addressed as early as

1969. A report by Hart on stereo-scopic calculations discusses sensitivity of triangulation process to pan, tilt, and quantization errors [49]. McVey and Lee developed equations for measuring the worst case error in calculating the distance between an object and a stereo vision system [76].

The navigation system reported in [31, 32] uses a single continuous strip painted on the floor marking the robot's route. For successful navigation of the robot, an important design consideration was the width of the line. That study analyzed the effects of various error conditions on the width of the line, as seen in the image plane, to determine the optimal line width on the floor.

The effect of image plane quantization on the determination of object location using stereo set-up was analyzed in [13, 85]. These works assumed that the scene point was equally likely to be everywhere within the volume formed by the lateral [13] or axial [85] stereo triangulation method, and derived the probability distribution of the errors in all three component directions.

Later in this section, we describe a new approach for determining the error in the camera state. Our approach is based on the fact that two or more lines-of-sight connecting the scene points and the corresponding image pixels meet at a single point known as the focal point. Due to finite size of the pixel, for each scene point and the corresponding pixel, we can consider four lines-of-sight passing through the four corners of each pixel. Two or more scene points and their corresponding image pixels result in a polyhedron within which the focal point is expected to lie. The volume of this polyhedron is proportional to the error in the camera pose. We propose to minimize this volume by considering more points. The selection of proper scene points – and also the number of scene points – reduces the region of uncertainty in the determination of the camera pose. This analysis can be used to determine good calibrating points, their distribution, and the number of scene points required to stay within the allowed range of computational error for a particular vision problem. In situations where it is not possible to determine the calibrating points in advance, this analysis can be used to compute the error in the estimated camera position. Our analysis can be used to design algorithm to select good feature points for calibration from available pool of feature points dynamically.

2.2.2 Analysis of Image Plane Quantization Error

Determination of camera external parameters using point-based features requires a certain minimum number of scene points $(S_1, S_2, S_3, \ldots, S_n)$ and their corresponding image points $(I_1, I_2, I_3, \ldots, I_n)$. Basic triangulation of these scene points and the corresponding image points requires all the lines-of-sight to pass through the focal point of the camera lens. Under ideal conditions (ignoring the effects of quantization of the image plane and lens distortion), for a given set of scene points and their corresponding image points, there can be only one unique position for the camera in the world coordinate system. Due to the finite size of an image pixel, however, a given scene point can be projected onto the image plane through an infinite number of points within the finite sized pixel. When any two such scene points are projected onto the image plane, the intersection of these lines of sight form a polyhedron of finite volume within which the focal point of the camera is expected to lie anywhere, as shown in Fig. 8. The approach presented here computes the range of values for camera state parameters (i.e., the 3-D coordinates of the center of the image plane in the world coordinate system and its orientation with respect to the reference axes that would

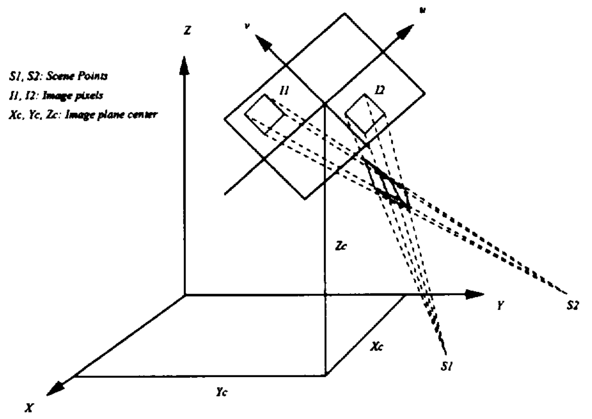


Figure 8: Lines of sight forming a polyhedron due to image plane quantization

satisfy the constraint for the valid location of the focal point).

As explained in the previous section, due to image plane quantization, basic triangulation of known scene points and their corresponding image points forms a polyhedron of finite volume within which the focal point of the lens is expected to lie. Since the camera's extrinsic parameters (position and orientation) are uncertain, it is not clear where this polyhedron is located in the world coordinate system. However, it is known that the focal point is at a perpendicular distance f (focal length) in front of the image plane along the optical axis of the camera, and that the scene point S_i corresponds to image point I_i in the image plane. Hence, there are ranges of values for each camera parameter that would satisfy the above two constraints. In the next two sections we present the mathematical treatment of the above concept, to find the range of values for each camera parameter separately.

The problem addressed here is different from the well known problem of determination of error in computing the 3-D position of the objects using a stereo system [13, 85], where the camera position is fixed and known and, as a result, the region of uncertainty for the object location is well defined by the camera geometry. However, in our problem scenario the location of the region of uncertainty is not defined a priori, since the camera position itself is uncertain. Although the problem discussed in [31, 32] is somewhat close to the problem being addressed here, their analysis is limited to a particular application. The approach taken in this work does not make any assumptions about the kind of outdoor scene points available for calibration.

Fig. 9 shows the system geometry used in this analysis. Two separate coordinate systems are shown: the coordinate (X, Y, Z) represents the world coordinate system, and the coordinate (u, v) represents the coordinate system in the image plane. (X_c, Y_c, Z_c) is the camera position in the world coordinate system and (θ, ψ, ϕ) are, respectively, the tilt, yaw, and roll angles of the camera. We follow the convention given in [45]: the term "camera position" or "sensor position" means the position of the center of the image plane, and the focal point is in front of the image plane. In the above geometry, we assume the gimbal center offset to be zero and $Z = 0$ to be the ground plane. Therefore, the height of the sensor is Z_c. Let Δu and Δv be the size of a pixel along the u and v directions, respectively, on the image plane. Let f be the focal length of the camera and $M \times N$ be the number of pixels along the u and v directions of the image plane, respectively, with the center pixel numbered as $(0, 0)$. We make the following assumptions in this analysis:

- This work analyzes the accuracy in obtaining the camera pose by triangulation of 3D scene points on a single quantized image plane. Image plane quantization is assumed to be the only source of error in computation of the camera position.

- A pinhole camera model is assumed [59], thereby ignoring camera lens distortion and

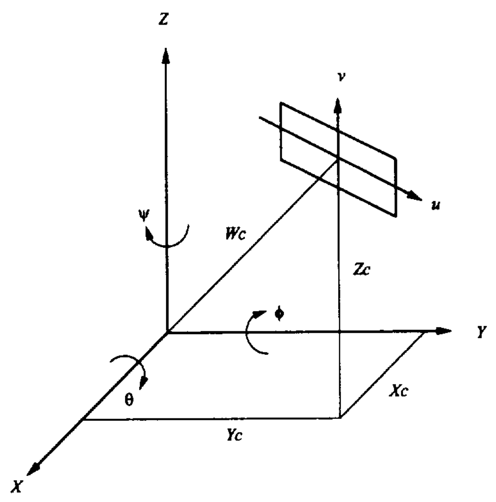


Figure 9: Camera geometry

other optical non-linearities.

- The problem of detecting the image point that corresponds to a given scene point is assumed to have been solved. The world coordinates of the scene points used in the calibration process are assumed to be accurate.

- The focal point is in front of the image plane, as is the case in any practical camera. Assuming the image plane to be in front of the lens center is not appropriate for this analysis, since this might introduce significant error in the volume of the polyhedron formed – especially when the scene points are close to the camera.

Analysis is carried out in the following four steps:

- Assume the camera position vector to be $(X_c, Y_c, Z_c, \theta, \psi, \phi)$, and develop equations for the world coordinates of the corners of the image pixels that are to be used in the calibration process.

- Form the plane equations that contain the scene point and the two adjacent corners of the pixel corresponding to the scene point.
- Compute the coordinates of the focal point in the world coordinate system for the assumed camera position vector.
- Verify that the focal point lies within the polyhedron formed by the intersection of the above planes.

The mathematical treatment of the above four steps are described in detail below.

A. Computing the world coordinates of the corners of image pixels

We assume the image plane to be at the origin of the world coordinate system, with zero tilt, roll, and yaw, as shown in Fig. 10. In this situation, the image plane coincides with the XZ plane and the world coordinates of a pixel (i, j) in the image plane are $(i\Delta u,\ 0,\ j\Delta v)$. When the camera is rotated, the coordinates of the pixel (i, j) in the world coordinate system can be computed by applying necessary point transformations [45] to the original point as given below.

$$\begin{pmatrix} X_{ij} \\ Y_{ij} \\ Z_{ij} \\ 1 \end{pmatrix} = R \begin{pmatrix} i\Delta u \\ 0 \\ j\Delta v \\ 1 \end{pmatrix} \tag{6}$$

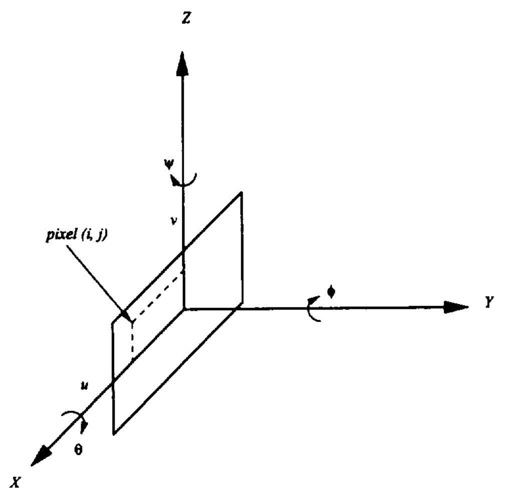


Figure 10: A point (i, j) in the image plane rotated about (X, Y, Z)

where

$$R = R_{\psi} R_{\theta} R_{\phi} = \begin{pmatrix} r_{00} & r_{01} & r_{02} & r_{03} \\ r_{10} & r_{11} & r_{12} & r_{13} \\ r_{20} & r_{21} & r_{22} & r_{23} \\ r_{30} & r_{31} & r_{32} & r_{33} \end{pmatrix} \tag{7}$$

The individual elements of the rotation vector can be computed by premultiplying the following three rotation vectors.

$$R_{\psi} = \begin{pmatrix} C_{\phi} & 0 & -S_{\phi} & 0 \\ 0 & 1 & 0 & 0 \\ S_{\phi} & 0 & C_{\phi} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} R_{\theta} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & C_{\theta} & S_{\theta} & 0 \\ 0 & -S_{\theta} & C_{\theta} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} R_{\psi} = \begin{pmatrix} C_{\psi} & S_{\psi} & 0 & 0 \\ -S_{\psi} & C_{\psi} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{8}$$

where C stands for cosine and S for sine of the respective angles.

Since we are concerned with computing the world coordinates of the corners of any pixel in the image plane, we can think of the imaging process as rotating every point in the image plane by applying proper rotation vectors, with the center of the image plane as the origin of the world coordinate system [45], and then translating them to a new position in the world coordinate system by applying proper translation vector. Therefore, for nonzero values of the camera position vector, the world coordinate of a pixel (i, j) in the image plane is given by

$$\begin{pmatrix} X_{ij} \\ Y_{ij} \\ Z_{ij} \\ 1 \end{pmatrix} = R \begin{pmatrix} i\Delta u \\ 0 \\ j\Delta v \\ 1 \end{pmatrix} + \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{9}$$

where

$$-\left\lceil \frac{M}{2} \right\rceil \leq i < \frac{M}{2}, -\left\lceil \frac{N}{2} \right\rceil \leq i < \frac{N}{2} \tag{10}$$

The 3D coordinates $(X_{ij}^k, Y_{ij}^k, Z_{ij}^k)$ of the four corners of the image pixel (i, j) in the world coordinate system can be computed by the following equations.

$$\begin{pmatrix} X_{ij}^k \\ Y_{ij}^k \\ Z_{ij}^k \\ 1 \end{pmatrix} = R \begin{pmatrix} (i + k_1)\Delta u \\ 0 \\ (j + k_2)\Delta v \\ 1 \end{pmatrix} + \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{11}$$

where $k = 0, 1, 2, 3$ stands for the four corners of the pixel (i, j). Values of k_1 and k_2 are decided by the value of k as shown in Fig. 11 .

B. Form the plane equations

The second step in the analysis is to form the plane equations that contain the scene point and the two adjacent corners of the pixel in the image plane that correspond to the scene

k	k1	k2
0	1	0
1	0	0
2	0	1
3	1	1

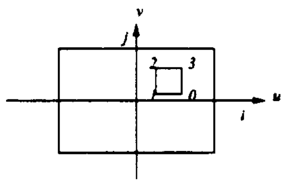


Figure 11: Pixel (i, j) in the image plane with numbers assigned to the corners

point (see Fig. 8). Let (X_p, Y_p, Z_p) be the coordinate of the scene point p whose image is formed at pixel (i_p, j_p) in the image plane. The coordinates of the four corners of the pixel (i_p, j_p) can be computed using Eq. 11 as

$$\begin{pmatrix} X_{ij}^{pk} \\ Y_{ij}^{pk} \\ Z_{ij}^{pk} \\ 1 \end{pmatrix} = R \begin{pmatrix} (i_p + k_1)\Delta u \\ 0 \\ (j_p + k_2)\Delta v \\ 1 \end{pmatrix} + \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{12}$$

Two adjacent corners of the pixel (i_p, j_p) and the scene point (X_p, Y_p, Z_p) form a plane. Every scene point and the corresponding image pixel gives rise to four such planes. Each of these planes can be represented by plane equations that contain the three points (X_p, Y_p, Z_p), $(X_{ij}^{k \bmod 4p}, Y_{ij}^{k \bmod 4p}, Z_{ij}^{k \bmod 4p})$ and $(X_{ij}^{(k+1) \bmod 4p}, Y_{ij}^{(k+1) \bmod 4p}, Z_{ij}^{(k+1) \bmod 4p})$. The equation for the plane can be written as

$$a_{ij}^{pk} X + b_{ij}^{pk} Y + c_{ij}^{pk} Z + d_{ij}^{pk} = 0 \tag{13}$$

where

$$
\begin{aligned}
a_{ij}^{pk} &= \begin{vmatrix} Y_p & Z_p & 1 \\ Y_{ij}^{kmod4p} & Z_{ij}^{kmod4p} & 1 \\ Y_{ij}^{(k+1)mod4p} & Z_{ij}^{(k+1)mod4p} & 1 \end{vmatrix} \\
b_{ij}^{pk} &= - \begin{vmatrix} X_p & Z_p & 1 \\ X_{ij}^{kmod4p} & Z_{ij}^{kmod4p} & 1 \\ X_{ij}^{(k+1)mod4p} & Z_{ij}^{(k+1)mod4p} & 1 \end{vmatrix} \\
c_{ij}^{pk} &= \begin{vmatrix} X_p & Y_p & 1 \\ X_{ij}^{kmod4p} & Y_{ij}^{kmod4p} & 1 \\ X_{ij}^{(k+1)mod4p} & Y_{ij}^{(k+1)mod4p} & 1 \end{vmatrix} \\
d_{ij}^{pk} &= - \begin{vmatrix} X_p & Y_p & Z_p \\ X_{ij}^{kmod4p} & Y_{ij}^{kmod4p} & Z_{ij}^{kmod4p} \\ X_{ij}^{(k+1)mod4p} & Y_{ij}^{(k+1)mod4p} & Z_{ij}^{(k+1)mod4p} \end{vmatrix}
\end{aligned}
\tag{14}
$$

$p = 0, 1, 2, \ldots, n$ are the scene points and $(a_{ij}^{pk}, b_{ij}^{pk}, c_{ij}^{pk}, d_{ij}^{pk})$ are the plane parameters.

C. Find the world coordinates of the focal point

The focal point of the camera in our model is at a perpendicular distance f in front of the image plane along the optical axis of the camera. The world coordinates (X_f, Y_f, Z_f) of the focal point for a given image plane position (X_c, Y_c, Z_c) can be computed (as in Step 1 of this analysis) by use of necessary transformation:

$$\begin{pmatrix} X_f \\ Y_f \\ Z_f \\ 1 \end{pmatrix} = R \begin{pmatrix} 0 \\ f \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} \tag{15}$$

D. Find the range of values for camera parameters

Four planes formed by a scene point and its corresponding pixel edges in the image plane intersect with other similar planes formed by other scene points and their corresponding image pixel edges, thus forming a common volume of a polyhedral shape. For a given camera position, the estimated focal point should lie within this volume (see Fig. 8). One way to check for this is to compute bounding surfaces of the polyhedron and then verify that the focal point lies within the polyhedron. This method is computationally expensive, since it is not easy to determine the coordinates of the vertices of the polyhedron formed by the intersection of many planes, especially when some of the planes may lie outside the polyhedron formed by the other planes. Instead, we extend the concept of solution set of linear inequalities [81] to determine if the point is within the polyhedron.

Solution set of Linear Inequalities

The solution set or graph of the inequality $ax + by + cz + d \geq 0$ consists of all the points that lie on or above the plane $ax + by + cz + d = 0$ if $c > 0$; and consists of all the points that lie on or below the plane $ax + by + cz + d = 0$ if $c < 0$. Similarly, the solution set of the inequality $ax + by + cz + d \leq 0$ consists of all the points that lie on or above the plane

$ax + by + cz + d = 0$ if $c < 0$; and consists of all the points that lie on or below the plane $ax + by + cz + d = 0$ if $c > 0$.

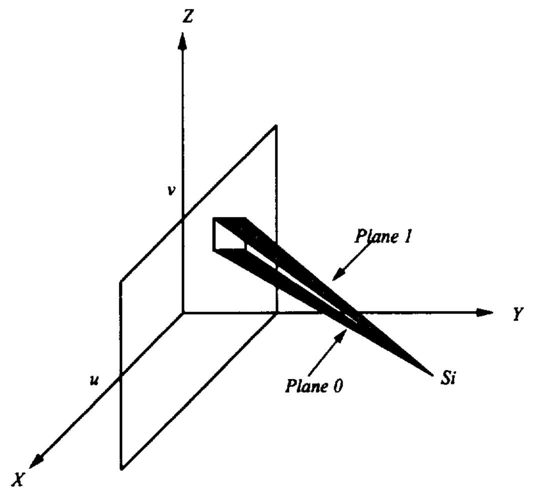


Figure 12: Two of the four planes formed by a scene point and its corresponding image pixel

Fig. 12 shows how the above concept can be used to verify whether the focal point is within the polyhedron. (For clarity, only one scene point and its corresponding image pixel is shown with only the two of the four planes (plane 0 and 2) formed by the scene point and the corresponding pixel edges.) According to the figure, the focal point has to lie below plane 2 and above plane 0. Hence, for (X_f, Y_f, Z_f) to lie within the common volume formed by the intersection of the planes, every plane has to satisfy the following inequality: for planes 0 and 2

$$\begin{aligned} a_{ij}^{pk} X_f + b_{ij}^{pk} Y_f + c_{ij}^{pk} Z_f + d_{ij}^{pk} \leq 0 \quad \text{if } c_{ij} > 0 \\ a_{ij}^{pk} X_f + b_{ij}^{pk} Y_f + c_{ij}^{pk} Z_f + d_{ij}^{pk} \geq 0 \quad \text{if } c_{ij} < 0 \end{aligned} \tag{16}$$

for planes 1 and 3

$$\begin{aligned} a_{ij}^{pk}X_f + b_{ij}^{pk}Y_f + c_{ij}^{pk}Z_f + d_{ij}^{pk} \geq 0 \quad \text{if } c_{ij} > 0 \\ a_{ij}^{pk}X_f + b_{ij}^{pk}Y_f + c_{ij}^{pk}Z_f + d_{ij}^{pk} \leq 0 \quad \text{if } c_{ij} < 0 \end{aligned} \tag{17}$$

The values of camera parameters are varied independently and separately around their true values to find the range of possible values for the camera parameters that will satisfy the above set of inequality equations formed by a set of n scene points and their corresponding image points.

2.2.3 Experimental Results

The above analytical procedure is used to determine the calibration error for a simulated camera geometry. The analytical procedure for simulating the test data is described in the following section.

2.2.4 Data Simulation

For verification of this analysis using simulated data, a camera with an $M \times N$ pixel sensor with sensor elements of size $\Delta u \times \Delta v$ is considered. Let f be the focal length of the sensor. For a given camera position vector $(X_c, Y_c, Z_c, \theta, \psi, \phi)$, a number of scene points with coordinates (X_p, Y_p, Z_p) that are computed to lie within the angular view of the camera are picked randomly. The coordinates of these scene points with reference to the camera axes are computed by applying the following transformation

$$\begin{pmatrix} u' \\ w' \\ v' \\ 1 \end{pmatrix} = R^{-1} \begin{pmatrix} 1 & 0 & 0 & -X_c \\ 0 & 1 & 0 & -Y_c \\ 0 & 0 & 1 & -Z_c \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{pmatrix} \tag{18}$$

where $R^{-1} = R_{\psi}^{-1} R_{\theta}^{-1} R_{\psi}^{-1}$. (Camera geometry is assumed to be the same as in the previous section where w is the optical or viewing axis of the camera.) (See Fig. 9.) (u', w', v') is the coordinate of a scene point with respect to the camera axes (u, w, v), and (X_p, Y_p, Z_p) is the scene point whose image coordinate is to be computed.

The position of the image point on the image plane after perspective transformation is given by

$$u'' = \frac{f \times u'}{f - w'}, \; v'' = \frac{f \times v'}{f - w'} \tag{19}$$

The coordinate of the image point in terms of pixel numbers is computed by applying proper image plane quantization as

$$i_p = \frac{u''}{\Delta u}, \; j_p = \frac{v''}{\Delta v} \tag{20}$$

The set of scene points (X_p, Y_p, Z_p) and their corresponding image coordinates (i_p, j_p) are used to compute the accuracy of the camera state estimates using the analytical procedure described in the Section 2.2.2.

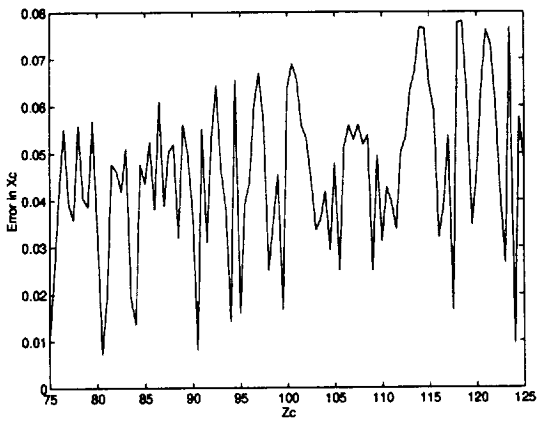
2.2.5 Results and Discussion

Three sets of tests were conducted to evaluate the effect of quantization on camera state estimation using point-based features. The objective of the first set of tests was to compute the error in various camera parameters as one of the camera parameters was continuously varied over a range of values. Figures 13(a) through 13(f) show the estimation error for each of the external parameters [i.e., $(X_c, Y_c, Z_c, \theta, \psi, \phi)$] using two fixed ground points, as the camera was moved along the Z direction with zero roll and yaw, and a tilt of 60°. A high resolution camera with a pixel width of 0.005 × 0.005 cm and focal length of 7.5 cm was considered. The coordinates of the two ground points used in the triangulation process were (10, 48, 0) and (-10, 68, 0). The error in computing the camera state increased as the camera moved away from the calibrating scene points. Similar results were obtained for different sets of calibration points and also when the camera moved in X and Y directions. The serrated nature of the error curve was also observed in the stereopsis experiment carried out by Hart [33]; similar arguments can be used to explain the process here.

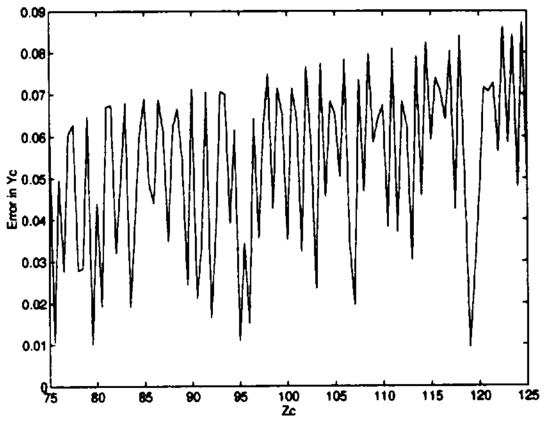
To study the variation of error with image resolution, the above set of tests were repeated for three sensors with different resolutions. The experiments were repeated for sensors with pixel size 0.0005 × 0.0005, 0.01 × 0.01 and 0.02 × 0.02. The error in computing the three positional parameters (i.e., (X_c, Y_c, Z_c)) for each of the sensors is plotted respectively in Figures 14(a) through 14(c) . In each of these cases, the camera was moved along the Z axis. Two ground points were used in the triangulation process, and the camera was assumed to have a tilt of 60°, with zero roll and yaw. As expected, the error in the camera pose estimation increased as the pixel size increased.

The objective of the third set of tests was to observe the effect of selecting more scene points for calibrating the camera. Figures 15(a) through 15(c) show the error in computing the positional parameters (X_c, Y_c, Z_c) of the camera by considering two and three calibrating points. A camera with pixel size of 0.01×0.01 was assumed with a tilt of 45°, zero roll and yaw. The coordinates of the three points used in the triangulation were (0, 46, 0), (4, 46, 0), and (-4, 46, 0). As expected, using three points improved the accuracy of estimation. Considering even more points has been found to further improve accuracy. The accuracy of computation was also found to depend upon the location of the feature points on the ground.

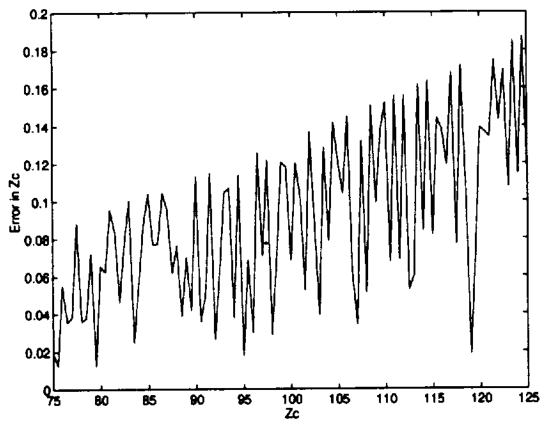
Image plane quantization – the natural outcome of an imaging process – has a significant impact on the accuracy of the outcome of all computational approaches to computer vision problems. Small errors introduced in estimating the camera pose due to image plane quantization might have significant impact on later stages of processing. The analytical approach proposed in this section lets one compute the error in each of the camera parameters separately. Given the scene points and their corresponding image points, one can decide in advance which scene points are good for estimating the camera pose and how many calibration points might be necessary to have the computed pose within the desired accuracy range. One drawback of the proposed analytical approach is that, with the present approach, it is not dynamically possible to identify good scene points or select an additional point to improve accuracy.



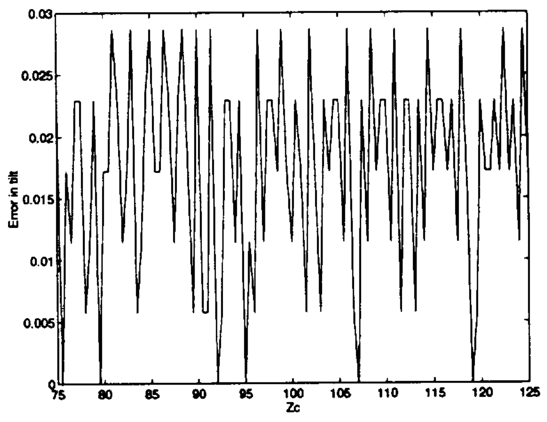
(a)



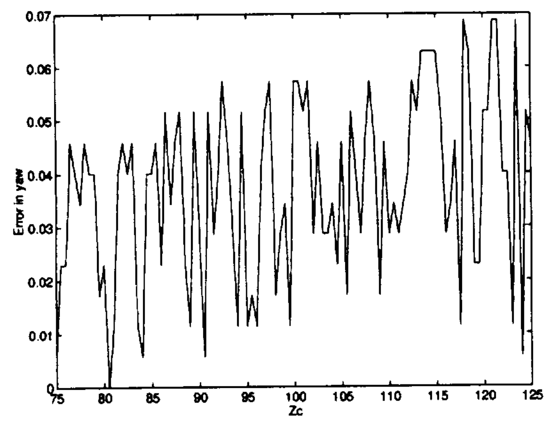
(b)



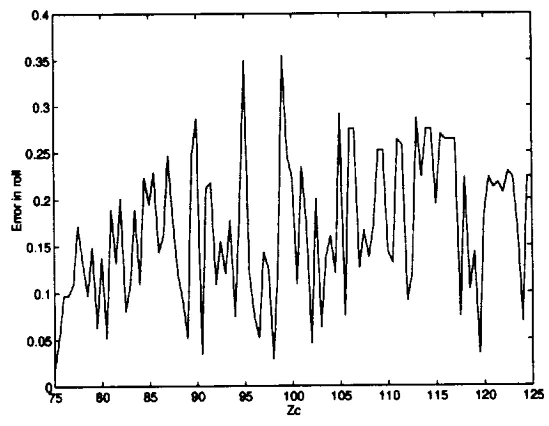
(c)



(d)

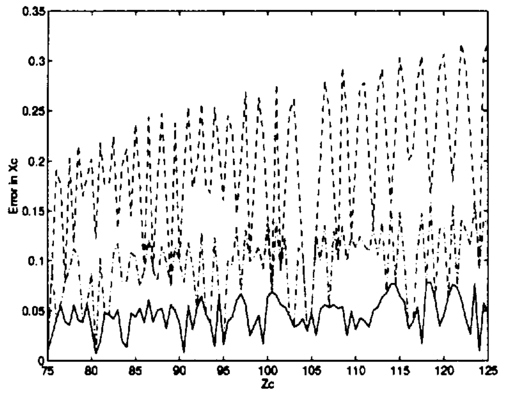


(e)

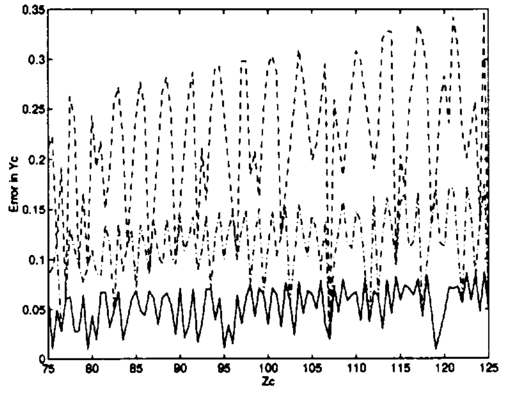


(f)

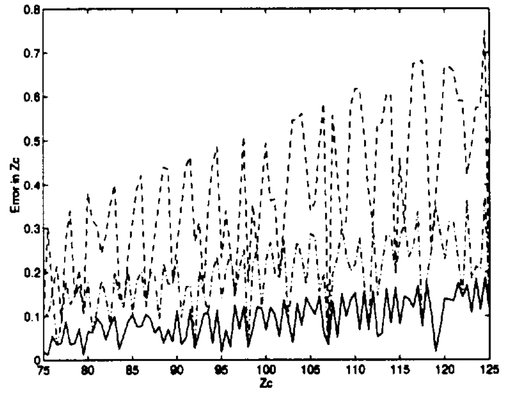
Figure 13: Error in camera parameters (resolution 0.005 cm): Error in (a) X_c (b) Y_c (c) Z_c (d) Tilt (e) Yaw (f) Roll



(a)



(b)



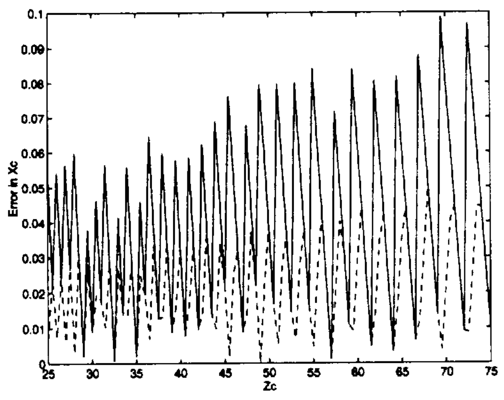
(c)

——-: Resolution of 0.005×0.005

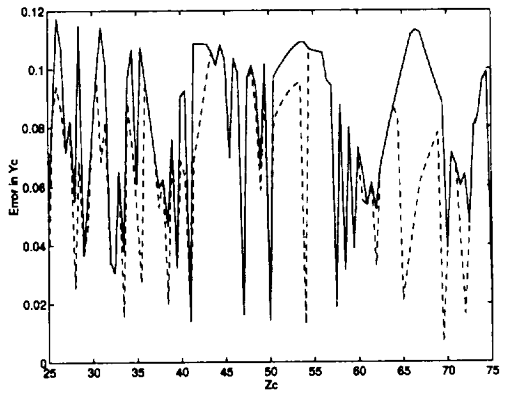
– . –: Resolution of 0.01×0.01

- - - -: Resolution of 0.02×0.02

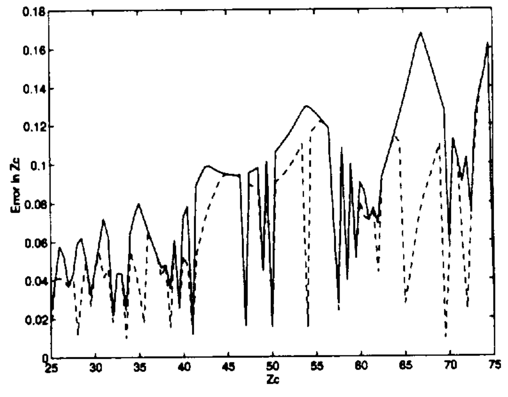
Figure 14: Error in camera parameters for sensors with different resolutions: Error in (a) X_c (b) Y_c (c) Z_c.



6a)



(b)



(c)

——: 2 calibration points

- - -: 3 calibration points

Figure 15: Error in camera parameters when using 2 and 3 calibration points: Error in (a) X_c (b) Y_c (c) Z_c.

2.3 Summary

Since the camera pose information provided by the GPS and on-board instrument is not accurate, we intend to use image-based features such as points or lines to estimate the position and orientation of the camera in 3-D. The accuracy of such estimation depends on the sensor resolution and sensor position in 3-D. A SVS is envisioned to be equipped with sensors of different modalities operating under different weather conditions and lighting situations. Since the resolution of these sensors varies, we wanted to explore whether the use of image-based features (such as points or lines) when used in a passive triangulation could provide a better estimate of the camera pose than what was provided by the GPS and other on-board instruments.

In section 2.1, we described an analytical model for computing the error in the camera pose estimation using image-based features. The sensitivity equations developed using the above model relate the error in the individual sensor positional parameters to the sensor resolution and the position of the image features in terms of the pixel number in the image plane. We evaluated three different sensors at six different positions using the equations developed based on the proposed analytical model. It is clear from our analysis that only HDTV can provide a better estimate than the GPS. The use of HDTV, however, is not currently approved or recommended. One alternative is to use the GPS information as the initial solution in a non-linear optimization algorithm to obtain an optimal solution from the image-based features obtained from the low resolution sensor images. The problem of integration of various sensor information was not explored in this research.

In section 2.2, we proposed a new method for computing the camera pose and the error in the estimated camera pose due to image plane quantization. This procedure was based on the fact that lines-of-sight formed by two or more feature points provide an uncertainty volume for the camera pose. We proposed a non-linear algorithmic approach for computing this uncertainty volume. By selecting more feature points, the uncertainty volume was reduced. Selecting good points from the available set of feature points, however, will improve the convergence of the algorithm. The earlier model provided the distribution of error in the camera pose estimation, whereas the second model provided an algorithm for estimating the camera pose and the error in the camera pose, and also minimized the error by considering more points.

If no prior knowledge about the likelihood of error in the estimate is available when selecting a point for use in camera pose estimation, then the second algorithm can be used alone in a greedy algorithm. However, this does not guarantee a global minimum in the error function. By using the first analysis, we can presort the feature points based on the sensitivity value at those pixels for a given position of the sensor, and use the features incrementally to obtain a global minimum in the error function.

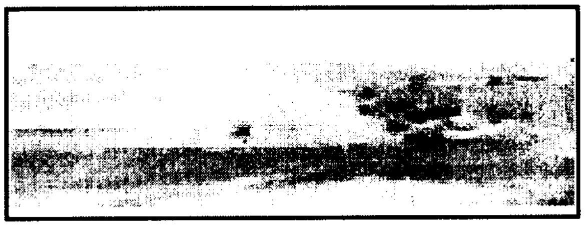


Figure 16: The input Passive Millimeter Wave image

3 Detection of Objects in Passive Millimeter Wave Images

A Synthetic Vision System (SVS) integrates the outputs of various sensors with GPS/INS information and airport geometry database to locate the aircraft within the airport, detect obstacles, determine potential conflicts, issue advisories, and sound cockpit alarms. The system selects among the outputs of different sensors depending on the visibility conditions, and processes these sensor outputs to extract informations to guide the pilot. In the previous section we analyzed the sensitivity of three sensors which could be used under three different visibility conditions(i.e, FLIR during night, HDTV during day light under normal weather conditions, and PMMW during fog). Since the resolution and quality of the images captured using these sensors vary, an algorithm for a specific purpose should be designed depending on the sensor type. In this section, we describe an algorithm for detection of both the runway and obstacles on the runway for images acquired using a PMMW sensor [117].

Since the energy attenuation in the visible spectrum due to fog is very large [132], sensors are being designed to operate at lower frequencies (e.g., 94 GHz). A lower attenuation provides the ability to see through fog. Images obtained from sensors operating at this frequency (such as the PMMW sensors) are of very low spatial resolution (Fig. 16); however, supporting information about the airport and the position, orientation, and velocity of aircraft is generally available. We used this information to guide our image analysis system. The geometric model of the airport contained positions of the runways/taxiways and buildings.The navigation instruments provided the position of the aircraft, and on-board instruments provided the orientation of the aircraft (yaw, pitch, and roll). We used this information to define regions of interest in the image where important features – such as runways/taxiways, the horizon, etc. – are likely to be present. Edges corresponding to these features of interest are detected within these regions. After delineating the regions representing runway/taxiways, we look for objects inside and outside these regions. The proposed system consists of the three functional modules, which are described below.

- *Model Transformation* allows input of the airport model and the camera state information to define regions of interest in the image plane.
- *Feature Detection and Localization* algorithm operates within these regions of interest, to detect the edges of the runway, horizon, etc., in the image.
- *Object Detection* algorithm uses simple histogram-based thresholding to detect object regions which are distinct from the homogeneous background. Two different thresholds are used to detect obstacles inside and outside the runway.

3.1 Model Transformation

Since PMMW images are low-contrast, low-resolution images, simple edge detection techniques on these images generate many noisy edge pixels, in addition to those belonging to the true edges, such as runways, sky, etc. This problem is alleviated by defining regions where the true edges are expected to occur, using knowledge about the aircraft position and a model of the airport. The main function of the module is to define a region of interest on the ground plane for each feature in the model and to perform 3-D to 2-D transformation. The module also defines a region in the image plane where the horizon line should occur.

3.1.1 Defining Regions of Interest for Runway Edges

The error between the expected location of a feature and its actual position in the image depends on several factors, most notably the accuracy of the camera position parameters used by the model transformation. Furthermore, it is evident from the analysis in section 2 that the ground area covered by a pixel is a function of the position of the pixel in the image. Thus, it is not reasonable to define the search space for each feature as a fixed number of pixels centered around the expected location in the image plane. We define the region of interest in the 3-D space and then apply transformation to get the corresponding region of interest in the image. The extent of the search space in the 3-D space is determined by the estimated error in the camera positional parameters (which are based on GPS and on-board instrument data).

The geometric model of the airport contains a sequence of 3-D coordinates for the vertices of the runway/taxiway, which form a polygon with n vertices:

$$runway = \{P_i\}\,,\ i = 1, 2, \ldots, n. \tag{21}$$

where $P_i = (X_i, Y_i, Z_i)$ is one of the vertices of the polygon. Note that $Z_i = 0$. P_iP_{i+1} specifies an edge of the polygon. The region of interest is defined as a rectangle on the ground which encloses the edge. Therefore, each edge P_iP_{i+1} of the polygon is associated with the region of interest by four points $b_i = (X_j, Y_j, Z_j)$, $j = 1, \ldots, 4$, and $Z_j = 0$.

The width of the region of interest is defined as a function of the width of the runway/taxiway, w accuracy of the GPS data, $g(g \leq 1)$ and the accuracy of the on-board instruments, $d(d \leq 1)$. Note that g and d are determined by the specification and characteristics of these instruments. This relationship is given by

$$width(w, g, d) = \frac{0.2w}{gd} \tag{22}$$

Note that the minimum width is $0.2w$ when $g = d = 1$, which corresponds to $\pm 10\%$ potential displacement of runway edge feature. To limit the search area from being a large fraction of the runway width, we limit the search width to $0.4w$ – even if $gd < 0.5$.

After defining the region of interest for each edge, 3-D to 2-D coordinate transformation is performed using perspective projection, rotation, and translation transformation matrices [45]. After perspective projection, following special cases are considered:

- The region of interest degenerates to a line in the image plane when the region is too far from the camera, and
- The region of interest in the image plane becomes very large when the edge is very close to the camera.

A minimum width is assigned in the image plane in order to provide some search space for the feature detector in the former case. A maximum width is defined in the image plane to further restrict the region in the latter. In our experiment, the minimum and maximum widths of a region of interest were set to 10 and 20 pixels, respectively.

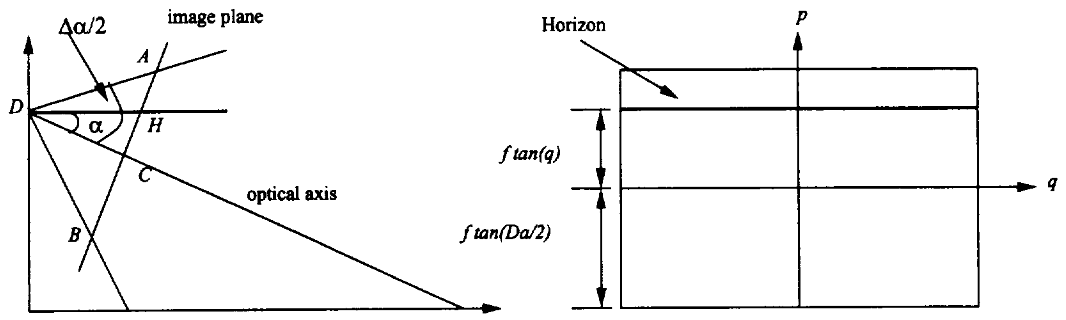


Figure 17: Horizon generation

3.1.2 Defining Search Space for Horizon Line

When the vertical angular field of view is larger than 2θ, a horizon line appears in the image (see Fig. 17). The horizon is an important clue to estimating the camera orientation, since it gives the roll angle information directly. The expected position of the horizon line is easily computed using the camera geometry shown in Fig. 17. The associated region of interest is defined to be 20 pixels wide centered around the expected horizon line in the image.

It is possible for the projection of the region of interest onto the image plane to be partially outside the image boundary. In such cases, we need to clip these regions so that the search space always remains within the confines of the image. This is done using a "polygon clip and fill" algorithm [41]. The regions of interest for both the runway and the horizon of the image sequence used in these experiments are shown in Fig. 18.

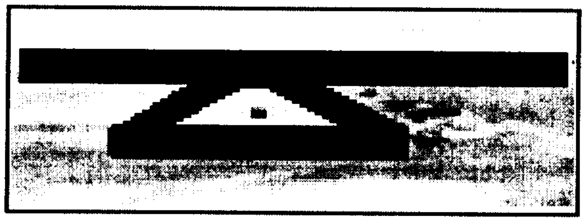


Figure 18: Region of interest

3.2 Feature Localization and Object Detection

3.2.1 Runway Localization

The system searches for the expected features within the region of interest, as defined by the previous algorithm. This will significantly reduce search time and also avoid the spurious response which is likely in such a low resolution input image. An accurate localization is necessary to estimate the motion parameters and camera pose.

A sobel edge detector is applied to the sensor image. We then select one of the four scanning directions $(-45°, 0°, 45°, 90°)$ which is approximately orthogonal to the direction of the expected edge. Along each scan line, we locate pixels with greatest edge strength. Since the runway edge is supposed to be a straight line, we fit a best line to these pixels. We also associate a measure of confidence for these detected edges based on the number of edge pixels detected along the line.

3.2.2 Object Detection

In this section, the region inside and outside the runway/taxiways are checked separately for the existence of stationary or moving objects. The image has three homogeneous regions: sky, the runway/taxiways, and the region outside the runway/taxiway. Any objects on or outside of the runway/taxiway are expected to have some deviation in gray level from their respective homogeneous background. Therefore, we use histogram-based thresholding for object detection. The thresholds which determine this deviation are set to be different for different regions.

We generate a mask image, which generates three homogeneous regions. Using this mask image, we generate the histogram of the original input image and compute its standard deviation for each region. The threshold value is determined as a function of the mean and the standard deviations. Any area which has gray level lower than the threshold is considered to be an object region. An object is assumed to have reasonable size. This size restriction on the object can be used to ignore spurious responses resulting from the thresholding. Each object is labeled based on the 4-connectivity.

3.3 Experimental Results

We tested our algorithm on a test image provided by the NASA Langley Research Center. This image was obtained using a single pixel camera located at a fixed point in space. (A camera with an array of pixels was under development when this work was carried out.) The camera was mechanically scanned to obtain a 50 × 150 pixel image. This is the image shown in Fig. 16 . We were also provided with a model of the runway which gave the 3-D world coordinates of the runway corners, locations of the buildings, etc. Using these data and the single image, we created a sequence of 30 frames to simulate the images from a moving camera. Frames 1(original), 5, 10 and 15 from this sequence are shown in Fig. 19. Edge-enhanced images corresponding to these frames are shown in Fig. 20. The regions of interest defined on these frames are shown in Fig. 21. Delineated features superimposed

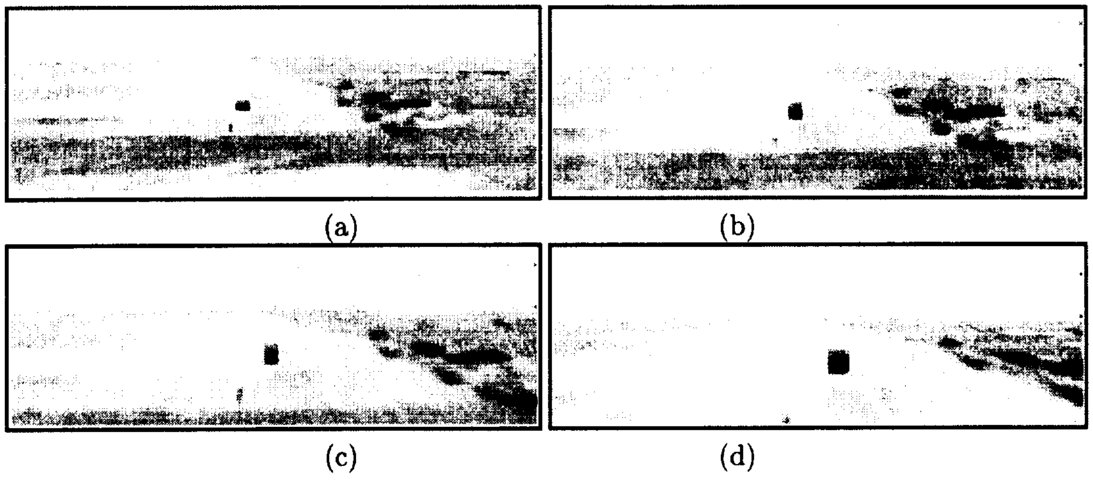


Figure 19: Input PMMW images (a)Frame 0 (b)Frame 5 (c)Frame 10 (d)Frame 15

on the image are shown in Fig. 22. Objects detected on the runway in Frame 1 and those outside the runway are shown in Fig. 23. Warning signals could be generated for each object on or near the runway.

3.4 Summary

In this section, we described a model-based system for recognition of objects in low resolution PMMW image sequences. Using knowledge of the camera motion, the runway model, and error in the parameters, the regions of interest for each of the runway edges are defined. We applied least square fit on strong edge pixels detected along the direction approximately orthogonal to the expected edge, to localize the runway feature within the search region. We tested our algorithm on a sequence of 30 frames and good results were obtained. The performance of the algorithm depend on the correctness of the search region and the robustness of the feature localization approach. The correctness of the search region depends on the knowledge of error in the camera pose and the runway model which, in turn, depends on

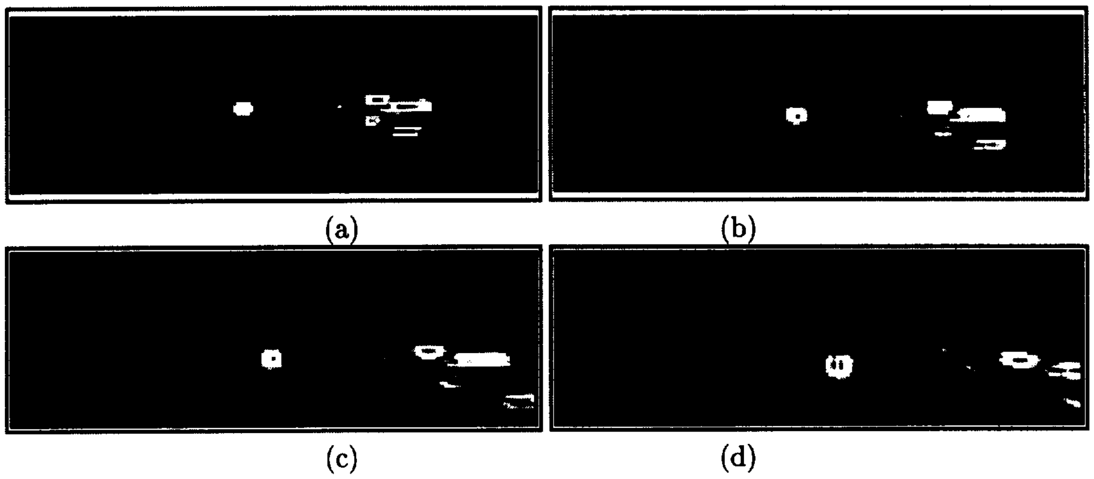


Figure 20: Edge images (a)Frame 0 (b)Frame 5 (c)Frame 10 (d)Frame 15

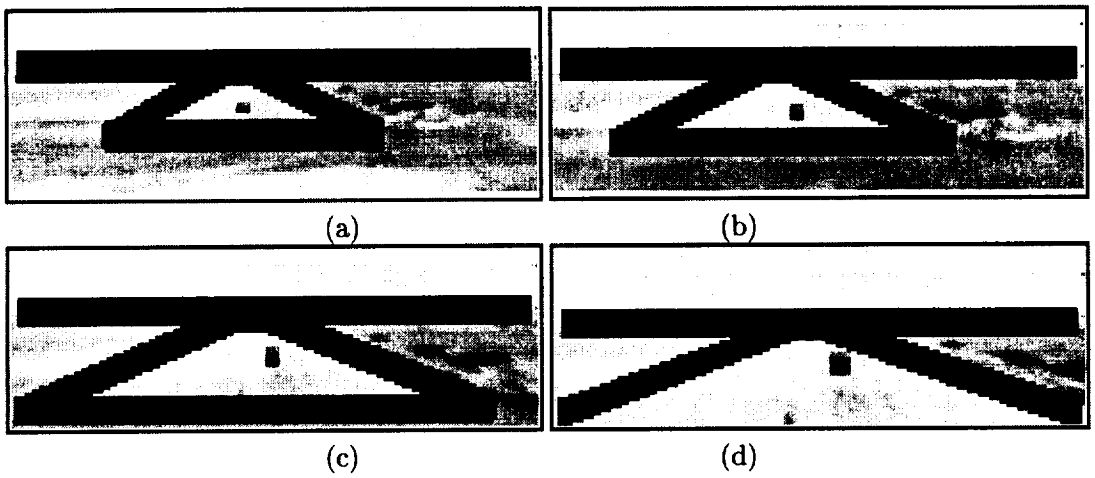


Figure 21: Region of interest images (a)Frame 0 (b)Frame 5 (c)Frame 10 (d)Frame 15

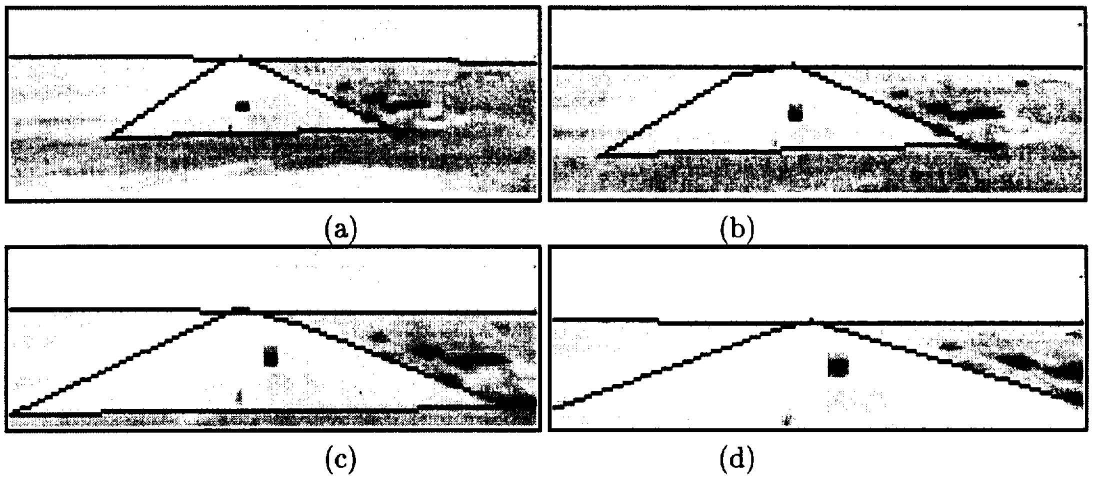


Figure 22: Detected runway (a)Frame 0 (b)Frame 5 (c)Frame 10 (d)Frame 15

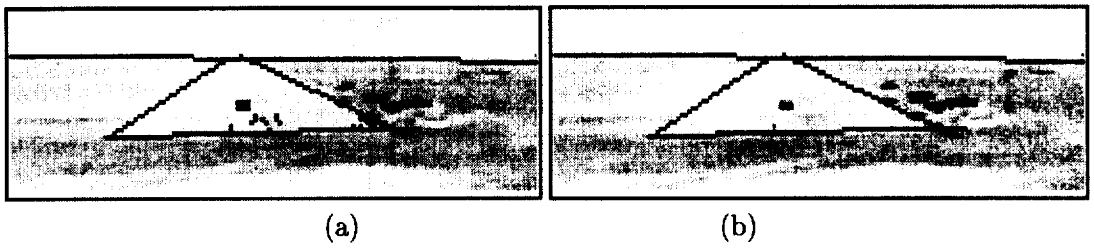


Figure 23: Detected object (a)Inside the runway (b)Outside the runway

the accuracy of the position estimate given by the GPS and INS. However, the robustness of the feature localization procedure depends on performance of the edge detector and the actual localization method. To make the method more robust, one can employ one or more of the following additional processing steps, however, at the cost of additional computation time: (1) consider strong edge pixels in each of the eight directions for least square fitting, (2) use a weighting function based on the gradient strength, (3) detect and remove outliers by computing the deviation of the edge pixels from the fitted runway edge.

A modified version of the runway detection algorithm is described in [119]. It was tested using a video image sequence obtained from a sensor mounted on-board a landing aircraft. Our results suggest that a model-based approach for detection of runways performs well, in cases of both PMMW and video sensors, and can be used to aid pilots in identifying the runway during landing.

The obstacle detection algorithm used in this work is based on the assumption that the regions are homogeneous. This assumption is valid for the PMMW image used in this experiment, and has also produced good results in all images of the sequence. On the other hand, it is a very weak assumption. It is not valid for images of different modality, especially in cases of video images. We have also investigated the real time implementation feasibility of various algorithms described in this section using available dedicated image processing boards from three separate vendors (Data-cube, Data-translation, and Imaging Technology) [29]. Further research on the feasibility of a PMMW sensor in the design of a SVS could not be continued for this study due to the unavailability of the sensor array.

4 Multiple Motion Segmentation and Estimation

A key to obstacle detection in Passive Millimeter Wave (PMMW) images is the homogeneity property of the different image regions that are identified by using the camera position information and the runway model. Since this assumption is not valid in the case of video images, a different method needs to be used to detect obstacles.

An important cue for detecting obstacles in a sequence of images obtained from a moving camera is the apparent motion of the brightness pattern in the image. Due to the camera motion, though, the whole scene appears to be moving, with image regions due to the obstacles moving differently than the background. Depending on the height of the obstacles and the 3-D motion of the obstacles relative to the camera, the apparent motion of the corresponding image regions will be different.

In this section, we describe a new segmentation algorithm for detecting obstacles in video image sequences using motion as the main cue. We assume that the background is planar or is piecewise planar and use a recursive motion-based segmentation algorithm to segment image into regions corresponding to different moving objects. Initially, to detect motion, optical flow is computed at corner-like feature points in the image. Optical flow vectors are then grouped into regions, and motion parameters are computed for each region using the new recursive motion-based segmentation and estimation algorithm.

A brief review of the past research in the area of motion detection, segmentation, and estimation is given in the following section, followed by an introduction to the proposed approach. The details of the individual stages of the proposed motion segmentation and estimation method are described in detail in Sections 4.3 and 4.4. The results obtained using both simulated and real images are also presented.

4.1 Methodologies for Motion Detection and Estimation

The relative motion between the objects in a scene and a camera gives rise to the apparent motion of objects in a sequence of images. This motion may be characterized by observing the apparent motion of a discrete set of features or brightness patterns in the images. Motion of objects in the scene can be derived by analyzing the motion of features or brightness patterns associated with the objects in the image sequence. Two distinct approaches have been developed for the computation of motion from image sequences.

The first approach, known as the *feature-based approach*, is based on extracting a set of relatively sparse, but highly discriminatory, two-dimensional features in the images corresponding to three-dimensional object features, such as corners, lines corresponding to edges demarcating the surfaces of the object in 3-D, etc. Such points and/or lines are extracted from each image, and inter-frame correspondence is then established between these features. Constraints are formulated based on assumptions such as rigid body motion i.e, the 3-D distance between two features on a rigid body remains the same after object/camera motion. Such constraints usually result in a system of non-linear equations. The observed displacement of the 2-D image features are used to solve these equations, leading ultimately to the computation of motion parameters for objects in the scene.

The other approach involves computing the two-dimensional field of instantaneous velocities of brightness values in the image plane or optical flow. A relatively dense flow field is estimated, usually at every pixel in the image. The optical flow is then used in conjunction with added constraints or information regarding the scene to compute the actual three-dimensional relative velocities between scene objects and camera.

4.1.1 Feature-based Motion Estimation

In this approach, point and line features are extracted from each image and inter-frame correspondence over two or more frames is established. Equations relating the relative position and motion between the camera and the imaged scene to the feature motion in the image are developed and solved to estimate the structure and motion. Many linear and non-linear methods for computing the structure and motion from line and point features are reported in the literature [6, 15, 16, 25, 26, 27, 28, 36, 37, 38, 50, 62, 63, 67, 68, 71, 72, 79, 89, 91, 92, 103, 104, 105, 106, 123, 126, 127, 128, 129, 131, 133].

Non-linear methods iteratively solve non-linear equations derived to relate 3-D motion parameters with the observables in the image plane as in [27, 28, 36, 62, 106], whereas methods using linear algorithms provides closed-form solutions to the motion parameters [37, 38, 63, 67, 68, 71, 72, 88, 123, 126, 128, 129]. The challenge in the case of non-linear methods is to solve non-linear equations in the form of objective functions, to minimize the error between the observed and the predicted motion.

Although various numerical methods could be applied to these non-linear equations, the global minimum may not be reached unless the initial guess is sufficiently close to the true value [28, 38, 128]. In addition a non-linear approach is computationally expensive, due to the exhaustive search of the solution space. Although linear algorithms use different methods to determine the unknowns, they share the same key structure: determine the intermediate parameters, which are called the essential parameters based on the epipolar constraint, then compute the motion parameters from the essential parameters. Even though the solution to a linear system is guaranteed (except for degenerate cases), the solution is highly sensitive to noise [37, 38, 123].

To improve the accuracy of estimation, researchers have proposed using either more data points than the required minimum in a two-view algorithm, or use data from a sequence of images [6, 15, 16, 27, 28, 91, 106]. Algorithms that use multiple images of a rigid scene to produce a more accurate reconstruction either process data from all of the images simultaneously, as in the case of batch algorithms [27, 28, 91, 104, 106, 126, 131], or update the previous estimate using the measurement data from the new frame by maintaining some notion of state using an extended Kalman filter [6, 15, 16]. An optimal estimation procedure to overcome the noise sensitivity of linear algorithms and inaccuracy of non-linear algorithms due to an incorrect initial guess is suggested in [28, 127]. This is a non-linear optimization procedure, where the initial value for the non-linear algorithm is provided by a conventional linear algorithm.

A few researchers have considered the problem of reconstructing scenes using straight line segments [25, 26, 37, 50, 67, 68, 103, 104, 129, 133]. When lines are used as features, two views are no longer sufficient and a minimum of three views are required [37]. This is because 3-D lines possess an additional degree of freedom when compared to the 3-D points. In other words, one can slide a 3-D line along itself and obtain the same line. When the lines are on the same plane, however, a linear algorithm can be formulated from two views [114]. Four or more lines are required to uniquely estimate the motion parameters. The non-linear algorithm described in [67] requires six line correspondences over three frames. The linear algorithm described in [68] requires 13 line correspondences over the same number of frames. In reality, the features detected could be sparse and, as a result, reliable estimation cannot be achieved using a single cue. An integrated approach for motion and structure estimation is described in [114], which integrates the measurement from various features such as points, lines, regions, and texture using a non-linear optimization procedure.

4.1.2 Optic Flow-based Motion Estimation

In this approach, the instantaneous changes in image brightness values in the image are analyzed to yield a dense velocity map called the image flow or the optical flow. The three dimensional motion and structure parameters are then computed based on various assumptions and/or additional information. No correspondence between features in successive images is required. The optical flow technique relies on local spatial and temporal derivatives of image brightness values.

The image velocities are, in general, functions of the motion of viewed objects relative to the camera, objects' locations in 3-D space and 3-D structure of the objects. The recovery of the 3-D motion and structure information from the sequence of monocular images can be decomposed into two steps: 1) compute image plane velocities from changes in image intensity values, and 2) use optical flow to compute 3-D motion and structure. Since the optical flow constraint equation relating the optical flow to the spatial and temporal image intensity gradient is not sufficient by itself to specify the optical flow uniquely, additional assumptions are made to constrain the solution. Popular assumptions include: 1) optical flow is smooth and neighboring points have similar velocities, 2) optical flow is constant over an entire segment of the image, and 3) optical flow is the result of restricted motion for example, planar motion.

Horn and Schunck [53] combined the gradient constraint equation with a global smoothness term to constrain the estimated velocity field by minimizing the error over a domain of interest. Lucas and Kanade [74] implemented a weighted least squares fit of local first-order constraint to a constant model for optical flow in a small spatial neighborhood. Nagel [82, 83, 84] used the second-order derivative to measure the optical flow along with the oriented-smoothness constraint, in which smoothness is not imposed across steep intensity gradients. Since accurate numerical differentiation may be impractical (due to either noise, a small number of frames, or because of aliasing in the image acquisition process), region-based matching methods are used for computing optical flow in [3, 94]. Such approaches define the image velocity as the shift that yields the best fit between image regions at different times. Finding the best match amounts to maximizing a similarity measure such as normalized correlation [94] or minimizing a distance measure such as the sum of squared difference [3].

Fleet and Jepson [40] investigated the extraction of motion information using Fourier techniques. Their method defines the component velocity in terms of the instantaneous motion of level phase contours (zero crossing in Difference of Gaussian or Laplacian of Gaussian image is viewed as level-phase crossings) in the output of band-pass velocity-tuned filters. Given the component velocity estimates from the different filter channels, a linear velocity model is fit to each local region. Heeger [52] used 3-D Gabor filters tuned to different spatiotemporal frequency bands and described a method for combining the outputs of the filters to compute local velocity vectors.

Having computed the optical flow, there still remains the problem of computing the motion and structure of the object in 3-D space. The estimation of structure and motion is based on the key assumption that the optical flow varies smoothly and the surface of the object is smooth. A system of equations that relates the optical flow and its first and second-order derivatives to the 3-D structure and motion parameters is derived. This non-linear over-determined system of equation may or may not yield a unique solution. A detailed analysis of numerous cases has been presented by Subbarao [112] and Waxman [124, 125], who have derived closed-form solutions for these cases. Subbarao shows that the solution, in general, is unique and at most four solutions are possible in certain cases. Neghadaripour [86] also addressed the ambiguity in interpreting optical flow produced by curved surfaces in motion. The ambiguities inherent in interpreting noisy flow fields are discussed by Adiv [2].

4.1.3 Multiple Motion Segmentation and Estimation

The general paradigm for time-varying imagery analysis in cases of feature-based approach uses feature extraction, feature motion, and motion parameter estimation. In the formalism of optical flow, the first two steps are merged in the computation of optical flow. The works mentioned in the above review assume that the feature points, or optical flow vectors, result from a single rigid object in motion. The main foci of such work are the minimum number of features required to compute a solution, the possibility of multiple solutions, and the effect of noise. If an image contains two or more objects moving independently, a segmentation procedure becomes necessary before any estimation can be done. In the optical flow method, this consists of grouping pixels corresponding to distinct objects into separate regions (i.e., segmenting the optic flow map, and then computing the three-dimensional coordinates of surface points in the scene corresponding to each pixel in the image at which the flow is computed). In feature-based analysis, the computing structure corresponds to forming groups of image features for each object in the scene and then computing the 3-D coordinates of the object feature associated with each image feature.

In autonomous navigation, it is essential to obtain such a 3-D description of the static environment in which the vehicle is traveling, and to estimate the range from an obstacle to the camera, in order to avoid collision by changing the vehicle's nominal path. The methods for range estimation described in [7, 22, 95, 96, 97, 107, 108, 110, 115] used Kalman filter to recursively estimate the range at feature points in the image. The range estimation procedure described in [118] used a simple incremental weighted least squares method for estimating the 3-D position of a stationary object using known camera state parameters. This algorithm extended the epipolar plane image analysis described in [8, 14] to general camera motion by assuming the camera motion to be piecewise linear. Adiv segmented the optical flow based on fit to an affine model [2]. The algorithm further grouped the resulting regions to fit a model of a planar surface undergoing 3-D motions in perspective projection. In [78], optical flow associated with the contour chain points is computed and chain points are grouped based on the spatial proximity and coherency of the apparent movement. Thompson and Pong [121] and Nelson [87] detected moving objects on the basis of simple flow clustering or inconsistency with the background flow. Black [12] described an approach that formulates a model of surface patches in terms of constraints on intensity and motion, while accounting for discontinuities. An incremental minimization scheme is used to segment the scene over a sequence of images.

4.2 Problem Statement and the Proposed Approach

The focus of this work is to detect obstacles in video image sequences obtained from a moving camera. In general, an obstacle is defined as an object, either stationary or moving, which is in the path of the moving camera. We assume that the background is planar or is piecewise planar. We are interested in detecting ground-based obstacles.

Since the camera is moving resulting in image motion everywhere on the image plane, simple image differencing approaches (such as those described in [51, 58, 57]) are not applicable. Thompson and Pong [121] have shown by example that, different motion and structure situation can result in identical optical flow distribution and hence, it is impossible to derive motion or structure information uniquely, given only the image and the optical flow field. If the camera motion and the background model are available, methods using warping (such as the one described in [20, 44, 115]) can be used to detect obstacles. Warping parameters are computed by matching image windows in two frames (as in [20]) or using the known camera motion and plane parameters (as in [44, 115]). Obstacles are detected by thresholding the residual optical flow computed using the warped images. If nothing is known about the camera motion or the background, approximate motion models are hypothesized and are verified for individual flow vectors or for a group of flow vectors as in [1].

In [1] an affine motion model with six parameters is used to describe the optical flow at a pixel in the image plane. A six dimensional Hough space is formed. Each flow vector votes to one or more bins of this Hough space, depending on the error between the model flow (computed using the affine motion model) and the actual flow (computed at the pixel using an optical flow algorithm). The success of the Hough method relies on prior knowledge about the range of parameter values. The accuracy of the method relies on the resolution of the Hough space.

In our approach, we assume that the background (i.e., the runway) is planar or is piecewise planar, and we use a planar motion model to describe the image motion. Even in cases where the background is not planar, the assumption is valid if the ratio of distance to the scene point from the camera to the variation in the depth of scene points is large. Based on this model, we develop a least square model fitting algorithm to segment the image and detect the obstacles. Unlike the Hough method, we do not have to rely on knowledge of the range of the model parameters. Instead, we estimate the parameters from the available data and refine the estimate recursively by removing the outliers.

Optical flow is computed using Lucas and Kanade's algorithm [74], and using Simoncelli's [93] modification to compute the covariance of the estimated optical flow. Unlike the conventional approach, which theoretically computes the optical flow at every pixel, we compute the optical flow only at selected feature points. Since the optical flow is large, and to avoid errors due to temporal aliasing, we computed the optical flow using hierarchical computation method and Gaussian pyramid. The details of the optical flow computation are given in the Section 4.3. The planar motion model and the segmentation algorithm to segment single and multiple motion are described in Section 4.4.

Various stages of our motion-based segmentation algorithm were tested using both real and synthetic image sequences provided by the NASA Ames Research Center. Frames 0, 25, 50, and 75 of the 90-frame real image sequence, *runway_crossing_new* (obtained by a camera mounted on-board a landing aircraft with a truck moving across the runway) are shown in

Fig. 24. In this imaging situation, the aircraft approaches the runway at an average velocity of (60.0, 3.0 8.0) feet/second. and the truck is moving across the runway at an average velocity of (13.0, 20.0, 2.0)[4]feet/second. The distance to the truck from the camera during the flight duration of approximately three seconds ranges from 500-370 feet. In addition to the truck and the runway lines, tire marks can be seen very clearly on the image. These tire marks are used as additional features for estimating the runway plane parameters and the detection of obstacles using motion-based segmentation. Synthetic images are created using a simulation software provided by the NASA Ames Research Center. Fig. 25 shows frames 1, 50, 100 and 150 of the sequence, *landing_normal_32L*, obtained using this software. The image sequence (which was originally in color) was converted to gray scale and the experiments were conducted on the gray scale image. These images were simulated for the case of a camera assumed to be mounted on-board a landing aircraft while another aircraft was crossing the runway. The image sequence was simulated using models for runway, aircraft, and textures for the runways, taxiways, sky, etc.

4.3 Optical Flow Computation

Many algorithms for computing optical flow have been described in the literature. (These were briefly introduced in the previous section.) Barron et al.[10] has classified these algorithms into four classes: gradient-based, region-based, phase-based, and energy-based, and has reported a good, quantitative evaluation of performance of various existing algorithms in each of these four classes. Despite their differences, many of these techniques can be

[4]The nonzero velocity of the truck in the vertical direction is due to the slope of the runway with respect to the reference coordinate system.

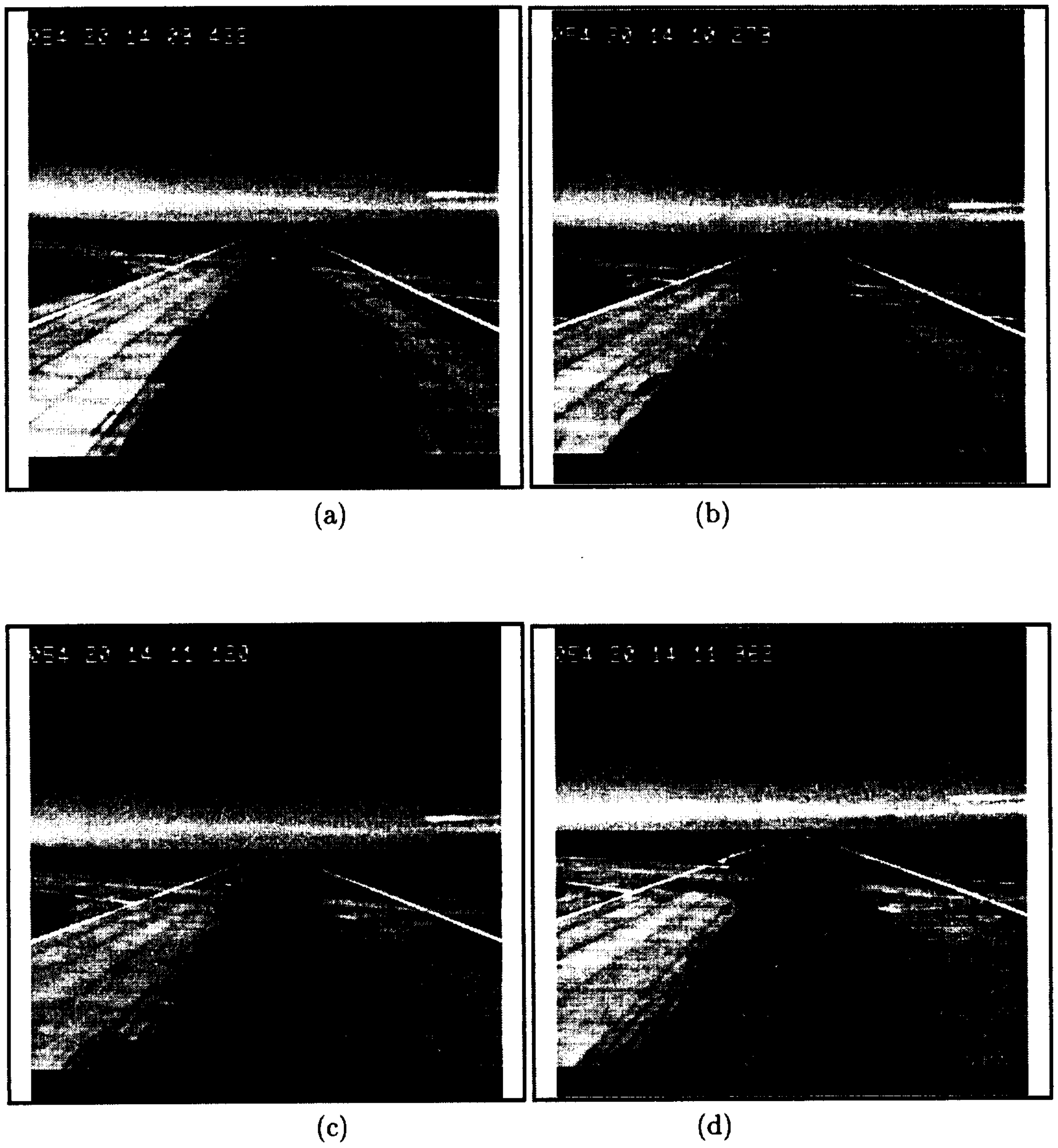


Figure 24: Real image sequence: *runway_crossing_new* (a)Frame 0 (b)Frame 25 (c)Frame 50 (d)Frame 75

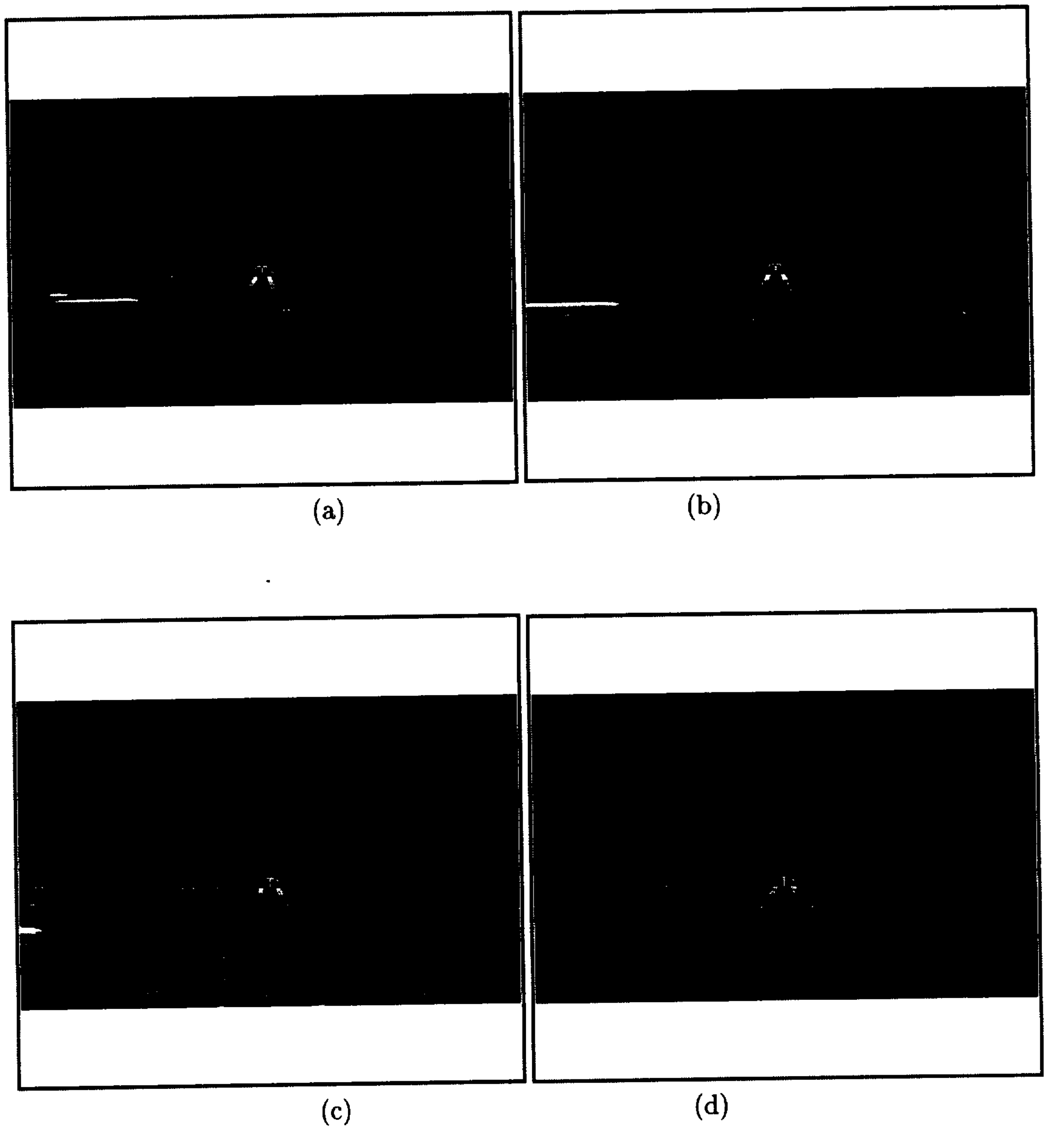


Figure 25: Synthetic image sequence: *landing_32L_normal* (a)Frame 1 (b)Frame 50 (c)Frame 100 (d)Frame 150

viewed conceptually in terms of following three stages of processing:

- Prefiltering or smoothing with low-pass/band-pass filters, in order to extract signal structures of interest and enhance the signal-to-noise ratio,
- Extraction of basic measurements, such as spatiotemporal derivatives (to measure normal components of velocity) or local correlation surfaces, and
- Integration of these measurements to produce a 2-D flow field, which often involves assumptions about the smoothness of the underlying flow field.

According to Barron's survey, Lucas and Kanade's method [74] provides estimation to sub-pixel accuracy and performs most consistently and reliably over all their test images. Due to these reasons we used the Lucas and Kanade's algorithm but with a modification suggested by Simoncelli et al. [93] that uses a statistical model to account for noise. The method not only provides the estimates of optical flow, but also their covariances. The algorithm requires at least five frames, and optical flow is computed for the central frame. Time and space gradients are found at positions of interest, and are used to estimate the optical flow.

Let $I(x, y, t)$ be the image intensity signal as a function of position of a pixel (x, y) in the image plane and time t. Assuming that there is no change in illumination, the brightness of a particular point in the pattern remains constant. Hence, the total derivative of the image intensity function must be zero at each position in the image at every time. This results in the standard gradient formulation of the optical flow problem written as

$$\vec{I_s} \cdot \vec{u} + I_t = 0 \tag{23}$$

where

$$\vec{I_s} = \begin{pmatrix} I_x \\ I_y \end{pmatrix} \tag{24}$$

and I_x, I_y and I_t are the spatial and temporal derivatives of the image I, and $\vec{u} = (u, v)$ is the optical flow at position (x, y) and time t that the derivatives have been computed. Because the constraint is formulated only in terms of the first derivative, image intensity is implicitly approximated as a planar function.

The squared error function based on the derivative constraint can be written as

$$E(\vec{u}) = \left[\vec{I_s} \cdot \vec{u} + I_t\right]^2 \tag{25}$$

Linear Least-Squares Estimate (LLSE) of $\vec{u}$ as a function of $\vec{I_s}$ and $\vec{I_t}$ is computed by setting the gradient of this equation to the zero vector:

$$\Delta E(\vec{u}) = M \cdot \vec{u} + \vec{b} = \vec{0} \tag{26}$$

where

$$M = \vec{I_s}\vec{I_s^T} = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}, \vec{b} = \begin{pmatrix} I_x I_t \\ I_y I_t \end{pmatrix} \tag{27}$$

The matrix M is always singular, since the solution is based on a planar approximation to the image surface at a point and, therefore, suffers from the aperture problem. Eq. 23 only places constraint on the velocity vector in the direction of $\vec{I_s}$; that is, on the component of flow in the direction of the gradient.

In order to eliminate the singularity problem, one can assume that the velocity vector is constant in a local region. This can be done by writing an error function based on the normal constraints from each point within a patch, indexed by a subscript $i \in \{1, 2, 3, \ldots n\}$:

$$E(\vec{u}) = \sum_i \left[\vec{I_s}(x_i, y_i, t) \cdot \vec{u} + I_t(x_i, y_i, t)\right]^2 \tag{28}$$

Computing the gradient of this expression with respect to $\vec{u}$ gives:

$$\Delta_u E(\vec{v}) = \sum_i \left(M_i \cdot \vec{u} + \vec{b_i}\right) \tag{29}$$

with solution

$$\vec{u} = -\left(\sum_i M_i\right)^{-1} \left(\sum_i \vec{b_i}\right) \tag{30}$$

where

$$M_i = M(x_i, y_i, t), \vec{b_i} = \vec{b}(x_i, y_i, t)$$

as in Eq. 27. Lucas and Kanade's approach [74] based on a Taylor series approximation to the solution of a matching problem in the context of stereo vision results in a solution identical to the equation 30. A weighting function w_i is also included in the summation, in order to emphasize the information closer to the center of the averaging region.

Although the above gradient-based method can provide estimates over a greater area than those obtained by feature matching, there will be errors in the estimated flow due to errors in the derivative computation, aliasing, imprecision in the derivative filters, etc. Even if the derivative measurements are error-free, the constraint in Eq. 23 may fail to be satisfied because of changes in lighting or reflectance, or the presence of multiple motions. Optical flow computation can be viewed as an estimation problem resulting in a probabilistic framework that allows these uncertainties to be represented in the computation [93]. Uncertainties computed in the form of a covariance matrix can be used as a weighting function in the interpretation stage.

Consider the total derivative constraint in Eq. 23. Let η_1 and η_2 be independent additive Gaussian noise terms describing the errors resulting from a failure of planarity assumption and errors in the temporal derivative measurements. Then Eq. 23 can be rewritten as

$$\vec{I_s} \cdot (\vec{u} - \vec{\eta_1}) + I_t = \eta_2, \; \eta_i \sim N(0, \Sigma_i)$$

or

$$\vec{I_s} \cdot \vec{u} + \vec{I_t} = \vec{I_s} \cdot \eta_1 + \eta_2 \tag{31}$$

Eq. 31 describes the conditional probability, $P\left(I_t \mid \vec{u}, \vec{I_s}\right)$. The desired conditional probability can be written using the Bayes' rule as

$$P\left(\vec{u} \mid \vec{I_s}, I_t\right) = \frac{P\left(I_t \mid \vec{u}, \vec{I_s}\right) \cdot P(\vec{u})}{P(I_t)} \tag{32}$$

The prior distribution $P(\vec{u})$ is chosen to be a zero-mean Gaussian with covariance Σ_p. The mean $\mu_{\vec{u}}$ and the covariance $\Sigma_{\vec{u}}$ may be derived using standard techniques [33]:

$$\begin{aligned} \mu_{\vec{u}} &= -\Sigma_{\vec{u}} \vec{I_s} \left(\vec{I_s^T} \Sigma_1 \vec{I_s} + \Sigma_2\right)^{-1} I_t \\ \Sigma_{\vec{u}} &= \left[\vec{I_s} \left(\vec{I_s^T} \Sigma_1 \vec{I_s} + \Sigma_2\right)^{-1} \vec{I_s^T} + \Sigma_p^{-1}\right]^{-1} \end{aligned} \tag{33}$$

If Σ_1 is chosen to be a diagonal matrix with diagonal entry σ_1 and the scalar variance of η_2 is $\sigma_2 \equiv \Sigma_2$, then the mean and variance of the estimated optical flow can be written as

$$\begin{aligned} \mu_{\vec{u}} &= -\Sigma_{\vec{u}} \cdot \frac{\vec{b}}{\left(\sigma_1 \parallel \vec{I_s} \parallel^2 + \sigma_2\right)} \\ \Sigma_{\vec{u}} &= \left[\frac{M}{\left(\sigma_1 \parallel \vec{I_s} \parallel^2 + \sigma_2\right)} + \Sigma_p^{-1} \right]^{-1} \end{aligned} \tag{34}$$

Since the distribution is Gaussian, the Maximum Likelihood Estimation (MLE) is simply the mean, $\mu_{\vec{u}}$. The optical flow is computed at selected feature positions detected using a feature detection algorithm described in the following section.

4.3.1 Feature Detection

Flow fields generated by existing algorithms are noisy and partially incorrect. Algorithms for interpreting these fields fail under such conditions. Although gradient-based techniques can produce a dense optical flow field, the estimated optical flow is not reliable at regions of low contrast, since the gradient is small, and at edge points, due to the aperture effect. But full optical flow can be determined at corners. Hence, we compute the optical flow only at corner-like feature points, unlike the conventional optical flow algorithms, which theoretically compute the optical flow at every pixel in the image plane. We do not use the matching approach to compute the disparity; instead, a gradient-based method is applied to the local region centered around the pixel.

Conventional feature-based methods for computing the optical flow identify distinct 2-D features on the image, then an inter-frame correspondence is performed to compute the image disparity at each feature point. It has been shown that, at any given pixel, theoretically, the process of computing the optical flow using region matching by Sum of Squared Difference (SSD) of image intensity is identical to computing the optical flow using a gradient-based approach. Optical flow methods proposed by Anandan [3] and Singh [94] compute optical flow using region matching approaches. These approaches can be viewed as feature-based. But, features are not explicitly identified using any criterion such as variance measure, interestingness or corner measure etc. Instead a local region at every pixel is identified as an interesting region. In this work, we integrated both feature-based and field-based approaches. Unlike Anandan's approach, we computed the optical flow only at distinct feature points where optical flow computation is expected to be reliable. Instead of region-based matching, we use the gradient-based approach applied to the local region surrounding the feature point.

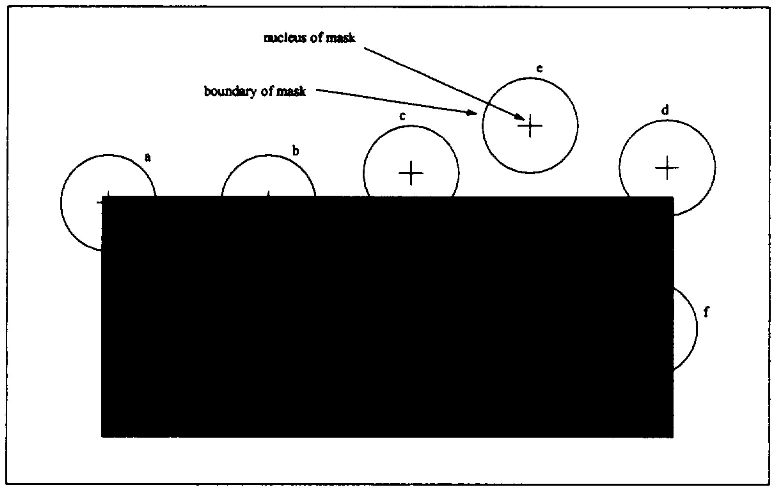
Barnard and Thompson [9], in their stereo disparity analysis, used points with high variance in all four directions as features computed using the Moravec interest operator [80]. Meygret and Thonnat [78] used contour chain points to compute the optical flow using matching. Regions of size 11×11 pixel with high variance are used as features in [95] to compute range to scene points using a Kalman filter-based recursive approach. We used the SUSAN (Smallest Univalue Segment Assimilating Nucleus) corner detector proposed by Smith [99]. This corner operator has demonstrated better performance than other corner detectors [98], and has also been used successfully to detect and track feature motion [100]. A brief description of the SUSAN corner detector is given below.

Smith and Brady [98, 100] have used a new approach known as the SUSAN principle to locate edges and corners in an image. Consider a rectangular box, as shown in Fig. 26(a). A circular mask is shown at seven different positions in the image. The central pixel of the mask is known as the nucleus. The area of the mask having the same (or similar) intensity as the intensity of the nucleus is known as "USAN" (Univalue Segment Assimilating Nucleus). In Fig. 26(b), the USAN is shown in white. Note that the USAN area is largest in the uniform portions of the image, smaller in the edge areas, and smallest near the corners of the image. Using this principle, one can locate the edges and corners of an image by taking local minima of USAN. Since the minimum is taken, the principle is known as SUSAN or Smallest Univalue Segment Assimilating Nucleus.

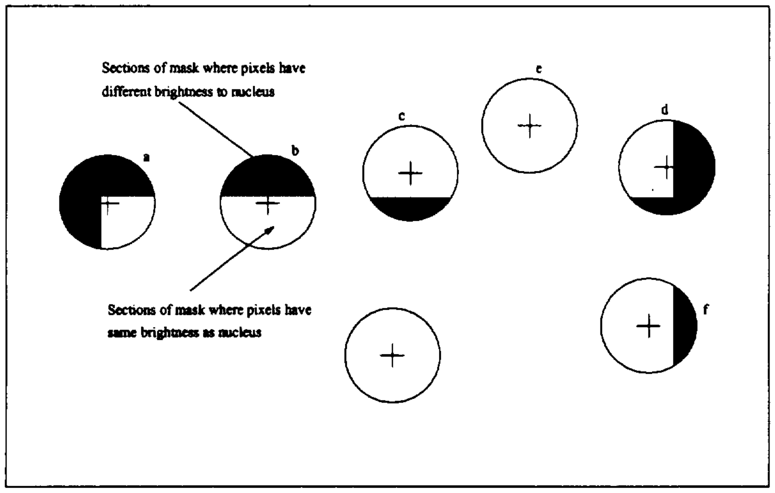
The SUSAN corner detector was applied to video image sequences captured from an aircraft. Fig. 27 shows zoomed part of frame number 50 of the image sequence *runway_crossing_new*. The output of the corner detector is shown in Fig. 28. The detector detects corner-like features wherever there is intensity variation in at least two directions. It can be seen that not only do the features correspond to the moving truck, but extraneous features such as tire-marks are detected by the corner detector.

4.3.2 Hierarchical Approach for Optical Flow Computation

The gradient-based approach for optical flow computation will fail if there is too much spatiotemporal aliasing (i.e., if the displacement being measured is greater than one half of a cycle of the highest spatial frequency present in the pre-filtered image sequence). In the case of a feature-based approach, the matching process that must accommodate large displacements can be very expensive to compute because of the large search space. Simple intuition suggests that, if large displacements can be computed using low resolution image, great savings in computation will be achieved. Higher resolution image can then be used to improve the accuracy of the displacement estimate by incrementally estimating small



(a)


Sections of mask where pixels have different brightness to nucleus
Sections of mask where pixels have same brightness as nucleus
a
b
c
d
e
f

(b)

Figure 26: SUSAN principle (a) Circular masks at different places on the image (b) USAN shown in white color.

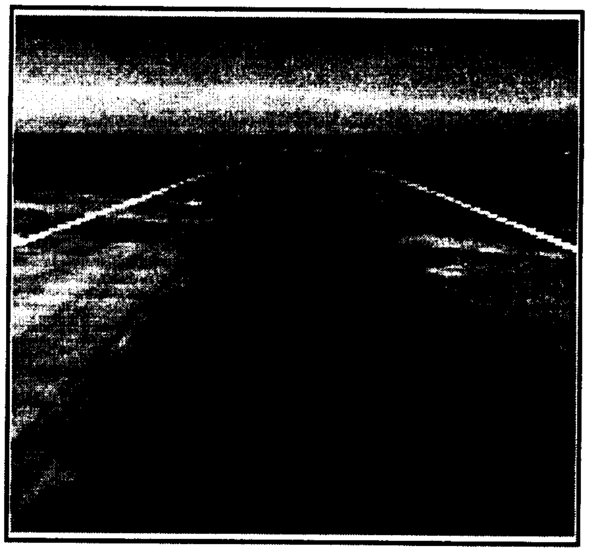


Figure 27: Zoomed part of frame 50 in the image sequence *runway_crossing_new.*

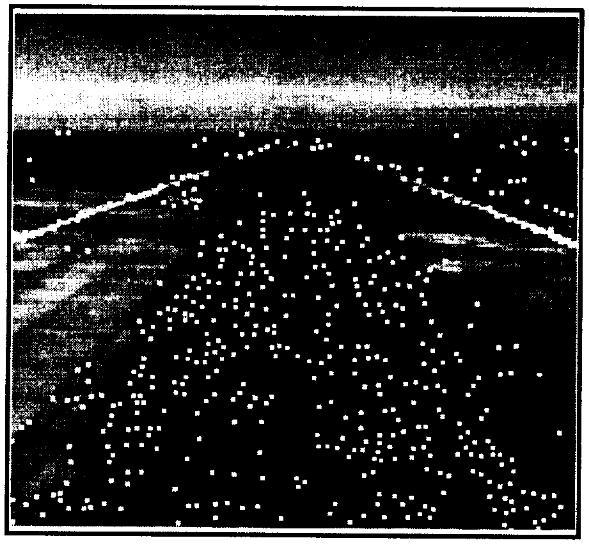


Figure 28: Detected corner-like features superimposed on the zoomed part of frame 50 in the image sequence *runway_crossing_new.*

displacements. It can also be argued that it is not only efficient to ignore the high resolution image information when computing large displacements, it is, in fact, necessary to do so. This is due to aliasing of high spatial frequency components undergoing large motion. Aliasing is the source of false matches in correspondence solutions or (equivalently) local minima in the objective function used for minimization in optical flow computation. Hence, matching or minimization in a multi-resolution framework helps eliminate problems of this type.

Hierarchical approaches have been used by various researchers in the past. Bergan et al. [11] described a hierarchical estimation framework for the computation of different model-based representations of motion information. Meygret and Thonnat [78] described an optical flow estimation algorithm based on a multi-resolution technique for matching contour chain points. A multi-grid approach for computing optical flow is described in [34]. Anandan [3] used large-scale intensity images to obtain rough estimates of image motion, which were then refined using intensity information at smaller scales. Optical flow is computed at every pixel in the image using region-based matching. We compute the flow only at feature points detected using the SUSAN corner detector in the high resolution image.

Fig. 29 describes the hierarchical optical flow computation framework used in this work. The basic components of this framework are:

- pyramid construction,
- optical flow computation,
- image warping, and

There are a number of ways to construct the image pyramids [90]. We built a Gaussian

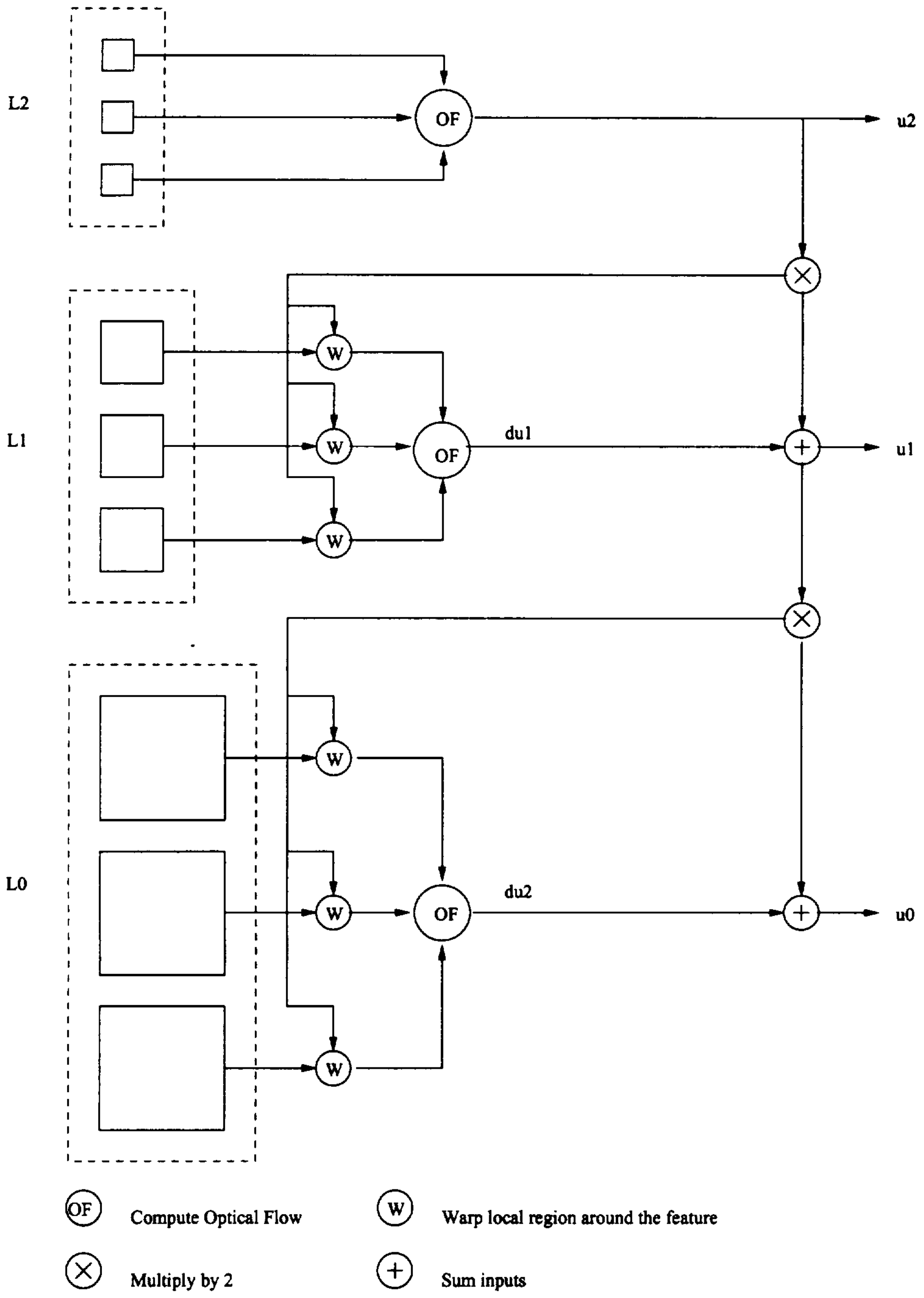


Figure 29: Diagram of the hierarchical optical flow estimation framework

pyramid on each frame of the image sequence [18, 90]. The Gaussian pyramid is a sequence of images in which each is a low-pass filtered copy of its predecessor. Let I_i be the i_{th} frame of the original image sequence; this becomes the bottom or zero level, L_0 of the pyramid. Let us represent this image as I_i^0. Each pixel of the Gaussian pyramid image at the next pyramid level L_1, image I_i^1, is obtained as a weighted average of pixels from I_i^0 within a 5×5 window. Each pixel of I_i^2 at pyramid level L_2 is then obtained by the same pattern of weights from I_i^1 at level L_1. The sample distance is doubled at each level. As a result, each image in the sequence is represented by an array which is half as large as its predecessor. The process which generates each image from its predecessor is called a REDUCE operation, since both resolution and sample density are decreased. Let I_i^0 be the original image and let I_i^N be the image at the top level N of the pyramid. Then for $0 < l \leq N$ we have

$$I_i^l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) I_i^{l-1}(2i+m, 2j+n) \tag{35}$$

The weighting function w is called the generating kernel which is separable, normalized, and symmetric. The one dimensional kernel is given by $w = [c\ b\ a\ b\ c]$ such that $a + 2c = 2b$. Combining these constraints we have: $a =$ free variable, $b = \frac{1}{4}$, and $c = \frac{1}{4} - \frac{a}{2}$.

Optical flow is computed using the gradient-based approach explained in the previous section. This method requires a minimum of five image frames; optical flow is computed for the center frame. Let I_i be the image frames used to compute the optical flow where $i = 1, 2, 3, 4, 5$. Initially, Gaussian pyramid images are computed for each frame. We have computed a Gaussian pyramid up to 3 levels (L_0, L_1, L_2), where level L_0 represents the original image. Features are detected in frame I_3 at level L_0, and their positions are predicted for level L_2. Optical flow is computed for each feature at level L_2 using the Gaussian pyramid images at that level. Let u_i^2 be the optical flow computed for a feature F_i at level L_2.

The optical flow computed at level L_2 and the feature position are upsampled to level L_1. Using the upsampled optical flow, the pyramid images at level L_1 are warped. The warping operation is defined as

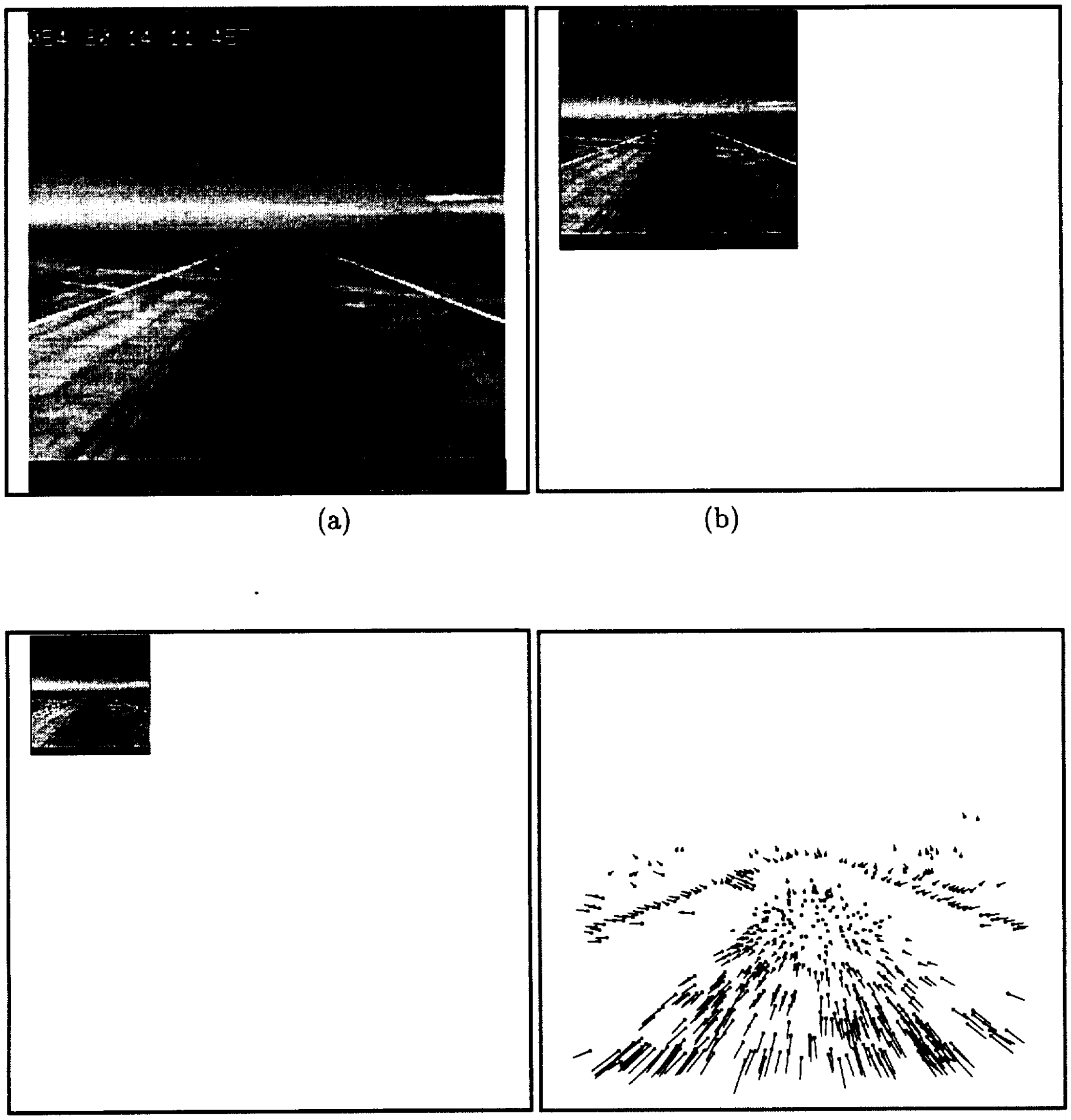
$$I_{warped}(x, y) = I_{original}(x - u(x, y), y - v(x, y)) \tag{36}$$

However, warping is done only for a local region around the position defined by the above relation. We use bilinear interpolation to evaluate the $I_{original}$ at fractional-pixel locations. The optical flow is computed using the warped images, resulting in "optical flow correction" δu_i^1 to be composed with the original estimate, to give a new optical flow estimate u_i^1. The optical flow at level L_1 after correction is given by:

$$u_i^1 = 2u_i^2 + \delta u_i^1 \tag{37}$$

The process of propagation of flow values and computation of flow correction is repeated at each level of the pyramid until the flow fields are at the resolution of the original image sequence.

We applied our multi-resolution optical flow computation algorithm on the image sequence *runway_crossing_new*. The Gaussian pyramid images and the optical flow obtained using our hierarchical framework for this sequence are shown in Fig. 30.



(a) (b) (c) (d)

Figure 30: Gaussian Pyramid images: (a)Level0 image (b)Level1 image (c)Level2 image (d)Optical flow at Level0

4.4 Motion-based Segmentation

Various approaches for motion estimation based on point or line features assume that the features undergo similar motion or features belong to a single object in motion. Optical flow computed using either the field-based or the feature-based approach computes the apparent image motion without the knowledge of the type of motion or the number of objects in the scene being viewed. Hence, before any estimation can be done using these optical flow vectors, flow vectors belonging to different objects and the background need to be grouped to identify image regions corresponding to the object/background.

In the context of obstacle detection, the problem would be to distinguish the obstacle from the background. If the camera is also moving, then background motion will be the dominant motion. If the camera motion and the model of the terrain in which the camera is moving are available, image warping can be used to compensate for the ego-motion of the camera and identify the obstacles. This is the approach taken in [20, 44, 115]. If nothing is known about the camera motion, then optical flow vectors are grouped based on some approximate motion model to form object hypothesis, as in [1]. The hypothesis is then verified by computing the deviation of the actual optical flow from the hypothesized model flow.

We use the motion model for segmenting the video image sequence where the background (i.e., the runway, in our application) is assumed to be planar. This is an approximate model used to hypothesize the background region. Even if the background is not perfectly planar, a planar motion model is valid in situations where the ratio of the range to the scene point to the variation in the range to other scene points is very small. As the camera approaches the runway, the planar motion assumption may not be valid. We still use the planar motion model to segment the image by assuming that the background is piecewise planar and applying the planar motion model to small local regions separately. In the next section, we derive the image velocity equation for the planar motion model case and describe an algorithm for recovering model parameters using a least square fit method.

4.4.1 Planar Motion Model

The imaging geometry and various coordinate systems used in this analysis are shown in Fig. 31. The earth reference coordinate axes $X_e, Y_e, and Z_e$ are rigidly affixed to the Earth at an arbitrarily selected but known point (on the runway). The helicopter body axes $X_b, Y_b, and Z_b$ are assumed to be located at the center of gravity of the helicopter, with the X_b axis pointing forward and Z_b axis pointing downward. The sensor coordinate system axes $X_s, Y_s, and Z_s$ are attached to the sensor, with the axis X_s passing through the optical axis of the sensor and originating at the focal point of the sensor. In this work, we use "sensor coordinate system" and "camera coordinate system" interchangeably. The image plane is located at a distance f from the focal point of the sensor, and is perpendicular to the optical axis of the sensor.

Let (X, Y, Z) be the 3D coordinates of a stationary point P in the sensor coordinate system. The projection of the point P in the image plane is given by

$$x = A\frac{fY}{X},\ y = f\frac{Z}{X} \tag{38}$$

where A is the aspect ratio. As the camera moves, the image point of P will also move.

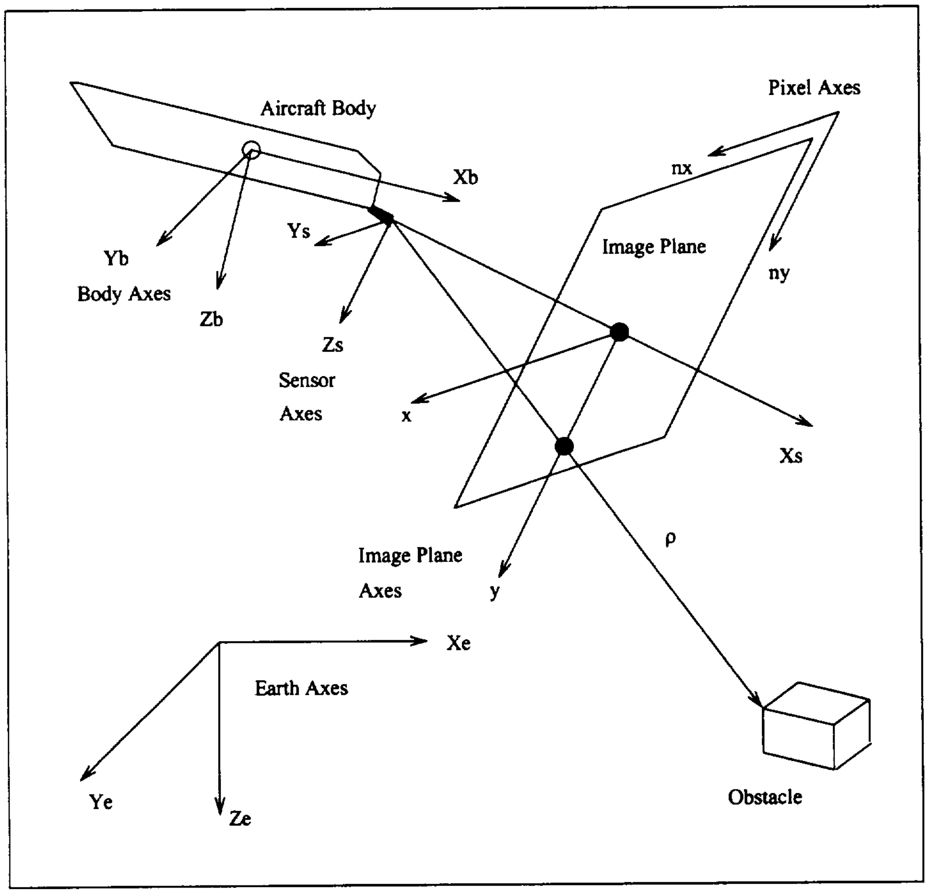


Figure 31: Imaging geometry

Differentiating the above perspective equation with respect to time, we have the image velocity $\vec{u} = (u, v)$

$$\begin{aligned} u &= \frac{\partial x}{\partial t} = Af\frac{X\dot{Y} - Y\dot{X}}{X^2} \\ v &= \frac{\partial y}{\partial t} = f\frac{X\dot{Z} - Z\dot{X}}{X^2} \end{aligned} \tag{39}$$

Let $\rho = (X, Y, Z)$ be the position vector for the object point P in the sensor coordinate system. If point P is assumed to be fixed in the Earth frame, the rate of change of ρ in the camera's axes system can be determined using Coriolis' equation [72]:

$$\begin{aligned} \dot{\rho} &= \begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{pmatrix} = -T - R \times \rho \\ \dot{\rho} &= \begin{pmatrix} -V_X \\ -V_Y \\ -V_Z \end{pmatrix} \times \begin{pmatrix} -Z\omega_Y + Y\omega_Z \\ -X\omega_Z + Z\omega_X \\ -Y\omega_X + X\omega_Y \end{pmatrix} \end{aligned} \tag{40}$$

where $T = (V_X, V_Y, V_Z)$ and $R = (\omega_X, \omega_Y, \omega_Z)$ are the camera's translational and rotational velocities.

Substituting for $\dot{\rho}$ in Eq. 39, we obtain a well-known optical flow equation which relates the camera motion, object motion in the image, and object's range:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_t \\ v_t \end{pmatrix} + \begin{pmatrix} u_\omega \\ v_\omega \end{pmatrix} \tag{41}$$

where

$$\begin{pmatrix} u_t \\ v_t \end{pmatrix} = \frac{1}{X} \begin{pmatrix} x & -Af & 0 \\ y & 0 & -f \end{pmatrix} \begin{pmatrix} V_X \\ V_Y \\ V_Z \end{pmatrix}$$

$$\begin{pmatrix} u_\omega \\ v_\omega \end{pmatrix} = \begin{pmatrix} Ay & \frac{xy}{f} & -fA(1 + \frac{x^2}{f^2A^2}) \\ -\frac{x}{A} & f(1 + \frac{y^2}{f^2} & -\frac{xy}{fA} \end{pmatrix} \begin{pmatrix} \omega_X \\ \omega_Y \\ \omega_Z \end{pmatrix} \tag{42}$$

As is evident from the above equation, optical flow can be decomposed into two components – (u_t, v_t) due to translation, and (u_ω, v_ω) due to rotation. Only the translational component of the image velocity is a function of the range.

Let the point P be on a planar surface whose plane equation in the camera coordinate system is given by:

$$k_1X + k_2Y + k_3Z = 1 \tag{43}$$

Substituting for X and Y from the projective transformation equation, we have:

$$k_1 + k_2\frac{x}{Af} + k_3\frac{y}{f} = \frac{1}{X} \tag{44}$$

Substituting for the $\frac{1}{X}$ in the image velocity equation, we have the optical flow equation for the planar motion case as:

$$\begin{aligned} u &= a_1 + a_2x + a_3y + a_7x^2 + a_8xy \\ v &= a_4 + a_5x + a_6y + a_8y^2 + a_7xy \end{aligned} \tag{45}$$

where

$$
\begin{aligned}
a_1 &= -Ak_1 f V_Y - fA\omega_Z \\
a_2 &= k_1 V_X - k_2 V_Y \\
a_3 &= -k_3 A V_Y + A\omega_X \\
a_4 &= -k_1 f V_Z + f\omega_Y \\
a_5 &= -\frac{k_2 V_Z}{A} - \frac{\omega_X}{A} \\
a_6 &= k_1 V_X - k_3 V_Z \\
a_7 &= \frac{k_2 V_X}{Af} - \frac{\omega_Z}{fA} \\
a_8 &= \frac{k_3 V_X}{f} + \frac{\omega_y}{f}
\end{aligned}
\qquad (46)
$$

From the above equation, it is clear that the instantaneous motion of a planar surface undergoing rigid motion can be described as a second order function of image coordinates involving eight parameters. The eight coefficients $(a_1, \ldots, a_8)$ are functions of the motion parameters and the the surface parameters. If the ego-motion parameters are known, then the three-parameter vector k can be used to represent the motion of the planar surface. Otherwise the eight-parameter representation can be used. In either case, the flow field is a linear function in the unknown parameters. Since our objective is to segment the image based on structure and motion, we used the eight-parameter representation for the flow field.

In the following sections we describe a least square fitting algorithm for recovering the eight parameters from a set of optical flow vectors, followed by our new segmentation algorithm based on this approach for segmenting multiple motions from optical flow field.

4.4.2 Recovering Model Parameter from Optical Flow

It is clear from the above analysis that each data point [i.e., optical flow (u, v) at pixel (x, y)] results in two equations containing eight unknown parameters. Hence, at least four points are required to solve for model parameters, assuming that all four points belong to the same planar motion. If more than four points are available, then a Least Square Fit can be used to compute the best model which fits all the points.

4.4.3 Least Square Model Fitting

Let (u_i, v_i) be the optical flow at pixel (x_i, y_i). The image velocity equation can be written as:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} 1 & x_i & y_i & 0 & 0 & 0 & x_i^2 & x_i y_i \\ 0 & 0 & 0 & 1 & x_i & y_i & x_i y_i & y_i^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{pmatrix}$$

$$\bar{u}_i = X_i \bar{a} \tag{47}$$

The least square solution for computing the motion model parameter vector $\bar{a}$ is given by

$$\bar{a} = (X^T X)^{-1} X^T \bar{u} \tag{48}$$

where

$$\bar{u} = \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ . \\ . \\ . \\ u_n \\ v_n \end{pmatrix}, \bar{X} = \begin{pmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 & x_1^2 & x_1 y_1 \\ 0 & 0 & 0 & 1 & x_1 & y_1 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & 0 & 0 & 0 & x_2^2 & x_2 y_2 \\ 0 & 0 & 0 & 1 & x_2 & y_2 & x_2 y_2 & y_2^2 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ 1 & x_n & y_n & 0 & 0 & 0 & x_n^2 & x_n y_n \\ 0 & 0 & 0 & 1 & x_n & y_n & x_n y_n & y_n^2 \end{pmatrix}$$

If Σ_u is the covariance of the computed optical flow, the least square solution for $\bar{a}$ is given by:

$$\bar{a} = (X^T \Sigma_u^{-1} X)^{-1} X^T \Sigma_u^{-1} \bar{u} \tag{49}$$

where the covariance of the estimated motion model parameter is:

$$\Sigma_a = (X^T \Sigma_u^{-1} X)^{-1} \tag{50}$$

4.4.4 Detecting Outliers

If the optical flow computed is exactly equal to the expected 2-D projection of the actual 3-D velocity, then the motion model parameters computed using the above least square method will be accurate. Since the optical flow is computed using the gradient approach, based on the flow constraint equation which is applied on the image locally, computed optical flow is only approximately equal to the 2-D projection of 3-D velocity. In addition, the computed optical flow will be noisy due to errors in computing the gradient, multiple motion, etc. Data points with large errors or flow vectors due to a different motion, called the outliers, can significantly bias the solution. Such points need to be identified and removed before reliable estimation can be done.

Let $(u_p(x,y), v_p(x,y))$ be the practical optical flow computed using the gradient-based approach. Let $\bar{a}'$ be the estimated motion model parameters using all the data points available. Then, the theoretical optical flow $(u_t(x,y), v_t(x,y))$ at (x,y) according to the recovered model is given by:

$$\begin{aligned} u_t(x,y) &= a_1' + a_2'x + a_3'y + a_7'x^2 + a_8'xy \\ v_t(x,y) &= a_4' + a_5'x + a_6'y + a_8'y^2 + a_7'xy \end{aligned} \tag{51}$$

The deviation of the point from the estimated model is computed as the deviation of the practical flow from the model (theoretical) flow given by the Mahalanobis distance:

$$d = (u_p - u_t)\Sigma_u^{-1}(u_p - u_t)^T \tag{52}$$

Points with $d > d_t$ are considered to be outliers, where d_t is a threshold value decided experimentally.

4.4.5 Algorithm for Recovery of Single Motion Model

From the above analysis, it is clear that under ideal situations where all flow vectors result from a single planar surface motion, a least square fit approach can be easily used to recover the motion model parameter. If there is no noise in the estimated optical flow the recovered model parameter will be identical to the actual parameters. If noise in the optical flow is small, then the least square estimate can provide a good estimate of the parameter vector. If there are few data points with large errors, then these points can have significant effect on the accuracy of the recovered model parameters. These noise data points, called the outliers, need to be detected and removed systematically before reliable estimates can be obtained. A simple iterative algorithm based on the least square fit method for recovering the motion model from a set of N noisy flow vectors is described in the following four steps.

1. Compute the model parameter $\bar{a}'$ using N data points in the least square sense. Data points include the coordinate of a pixel in the image plane (x, y) and the optical flow (u_p, v_p) computed at that pixel using the gradient-based approach described earlier.

2. For each data point, compute the theoretical optical flow (u_t, v_t) using the model parameter $\bar{a}'$ computed in the previous stage.

3. Compute the Mahalanobis distance d between the practical flow and the theoretical model flow. If d is above a threshold d_t, consider the data point as an outlier, else consider the point for refining the estimate in future iterations.

4. If there are no outliers detected in this iteration, the least square algorithm is said to converge and the model parameter obtained in this iteration is the best model parameter describing the planar surface motion. If one or more outliers are detected in this iteration, ignore these data points and repeat steps $1 - 4$ for the remaining data points.

4.4.6 Segmentation and Estimation of Multiple Motion

The algorithm described above can be used only when a sufficient number of good points are available. The least square fit algorithm fails if more than 50% of the data points are in error, which is the actual breakdown point for a least square method. This can happen for two reasons: 1) Majority of flow vectors, though are resulting from a single planar surface in motion, many of these vectors could be in error in addition to the few flow vectors resulting from a differently moving object. 2) The data points come from different objects moving in different directions. It is also possible that the scene may not be perfectly planar; thus, the optical flow vectors computed using an optical flow algorithm, although correct, may not follow the planar surface model.

In [20], the plane parameters were first computed by matching a portion of a previous image containing the planar region to the current image. Model parameter thus recovered was used to warp the second image to the first. The residual optical flow was then computed using the warped image to detect obstacles. In [115], the runway was assumed to be planar. Using the known camera motion and the plane parameters, images were warped and residual flow was computed. Large residual flow vectors due to obstacles were removed. The residual flow at other pixels was assumed to be caused by errors in the initial model parameter and was used to improve the model's accuracy. In [44], warping parameters were computed using the known camera motion and plane parameters. Features were tracked by computing a moving average of the optical flow. Pixels with large residual flow were detected as obstacles.

In this work, we considered a more general problem. We computed the model parameter vector from the available flow vectors, and identified the outliers in the set of data points by computing the deviation of the optical flow from the estimated motion model, as explained earlier. We assumed that the dominant motion was due to the planar surface in motion, and that the outliers were mainly due to obstacles. Our algorithm can be used for two applications:

- By computing the dominant motion, obstacles are detected in the form of outliers. In addition to detecting outliers, our algorithm also provides the motion model parameter for the planar surface on which the object is moving.
- Planar motion model is a good assumption if the range to the scene is large compared to the focal length. As the camera approaches the ground, however, the background may not be perfectly planar, in which case piecewise planar assumption can be used.

Although Adiv's work had a similar objective [1], his study uses the affine motion model to describe motion at any point, and uses Hough space to detect the multiple motions. Each optical flow vector votes to a set of bins in the six-parameter Hough space based on the difference between the optical flow value computed using an optical flow algorithm [53] and the theoretical optical flow expected for the model defined by the six parameter values associated with the bin. All flow vectors from a single rigid object in motion are expected to vote into the same bin, resulting in clusters in the Hough space, where each cluster corresponds to separate moving objects. The success of this method depends on the knowledge about the range of values for each of the parameters and the resolution of the parameter space. The higher the resolution, the more accurate the segmentation, although computationally it is going to be very expensive. In [122], an adaptive Hough approach is used where, initially, a low resolution Hough space is used. A higher resolution Hough space is then derived around the clusters detected in the first Hough space.

4.4.7 Split and Merge Algorithm

We assume that the runway or the terrain in which the camera is moving is either planar or is piecewise planar. We started by hypothesizing the runway to be perfectly planar, and applied planar motion model to all of the available flow vectors. If the initial hypothesis failed then we hypothesized the runway to be piecewise planar, and applied planar motion models to each of the pieces separately. The outline of the algorithm is shown in the block diagram in Fig. 32.

The algorithm starts with the hypothesis that all flow vectors belong to a single planar surface. The model parameter is computed using the least square approach, and outliers are detected by computing the deviation of each of the points from the computed model. The model parameter is refined by iteratively removing the outliers from the initial set of flow vectors. The algorithm is said to converge when no more outliers are detected. If the algorithm converges with less than 50% outliers, then the computed plane parameter represents the ego-motion of the camera and the structure of the background. Note that $\bar{\omega}$, $\bar{V}$, and $\bar{k}$ can be computed from $\bar{a}$. The outliers are either due to noise or independently moving objects. If the outliers are distributed randomly, they can be simply ignored. If the

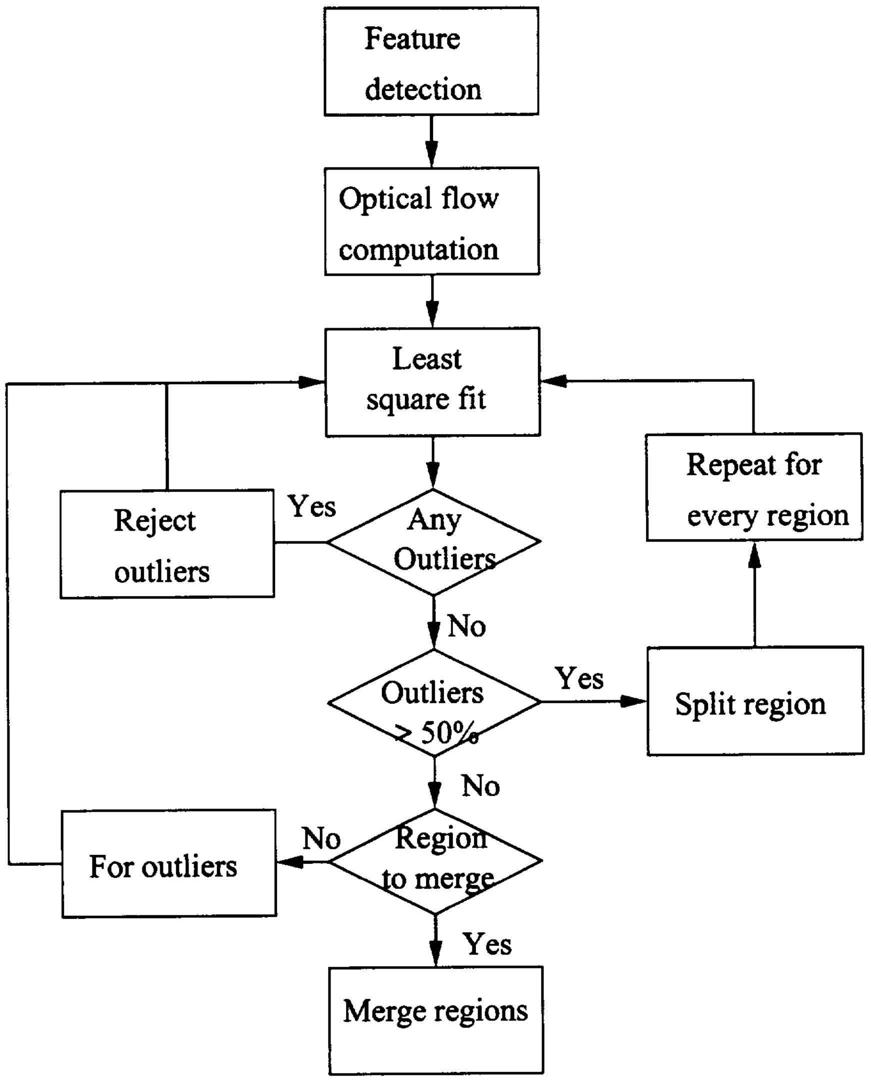


Figure 32: Split and merge algorithm

optical flow vectors are close together, then they can be considered to belong to a single object; a separate motion model parameters can be recovered from these sets of outliers.

If the least square fit algorithm converges with more than 50% outliers, then the solution is completely dropped. The background/runway is assumed to be piecewise planar. Piecewise planarity can be applied in two ways.

1. The algorithm can be applied to non-overlapping local regions of size $N \times N$ over the entire image and model parameters can be computed for each of these local regions. The regions can be then merged based on the spatial constraint and the merging error. This is not practical in this study because it is not clear how the feature points are distributed on the image plane and, therefore, local regions may not be well defined or evenly distributed.

2. We follow the split and merge algorithm used for intensity image segmentation. In this application, we split the image into four parts. The iterative least square algorithm is applied independently to each region. Where necessary, the regions were split recursively until the least square algorithm succeeds for a given region. Regions are then merged if the error after merging is below a threshold.

4.4.8 Experimental Results

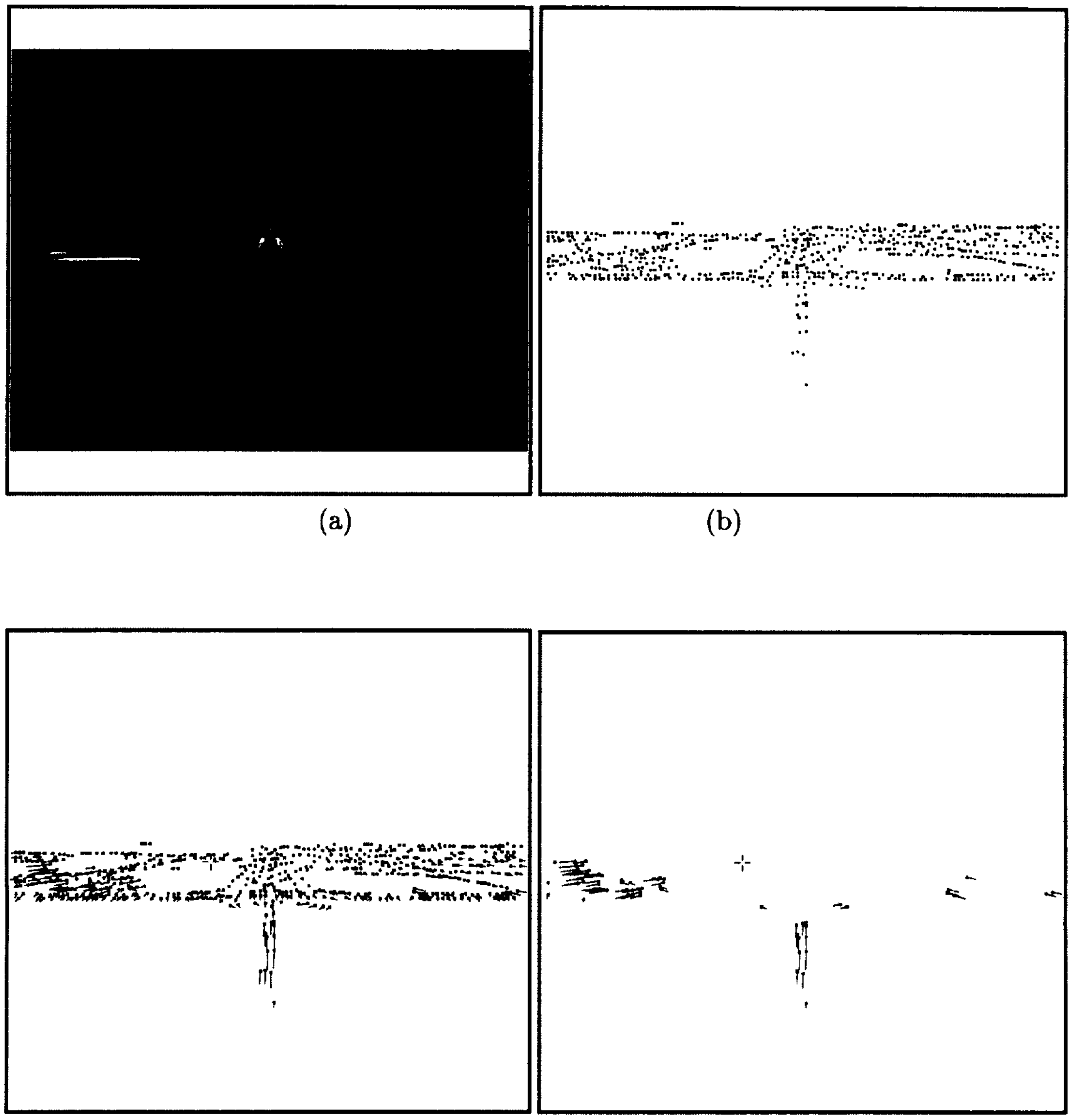
The motion-based segmentation algorithm for detection of obstacles based on the planar motion model for the background/runway with the incorporation of the split and merge technique is tested using both simulated and real image sequences. Fig. 33(a) shows frame 3 of an image sequence obtained using the NASA simulation software. Fig. 33(b) shows the features detected in frame 3, and Fig. 33(c) shows the optical flow computed for the feature points using our hierarchical framework. Fig. 33(d) shows flow vectors violating the planar motion model constraint, which have been identified as potential obstacles. The plane crossing the runway is very clearly detected as an obstacle in the images. (Few false alarms are due to noise in the estimated optical flow.)

Fig. 34(a) shows frame 50 of the real image sequence *runway_crossing_new* obtained from a camera mounted on-board a landing aircraft. Features detected in this image are shown in Fig. 34(b). Fig 34(c) shows the computed optical flow using our optical flow algorithm, and 34(d) shows the set of feature vectors identified as obstacles.

4.5 Performance Characterization

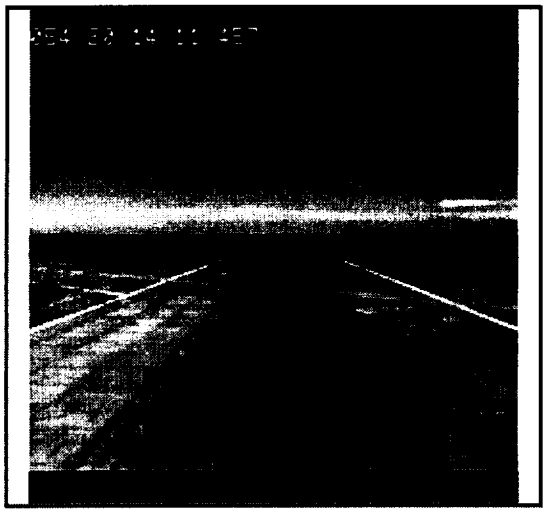
In this section, we evaluate the performance of our motion model recovery algorithm, assuming that the image contains a single planar surface in motion. The proposed system for recovering the motion model parameters from a sequence of images contains two sub-algorithms. First, optical flow is computed using the gradient-based optical flow method, then these optical flow vectors are used to compute the planar motion model parameters. Evaluation of the performance of this motion estimation algorithm requires determining the performance of each of these sub-algorithms.

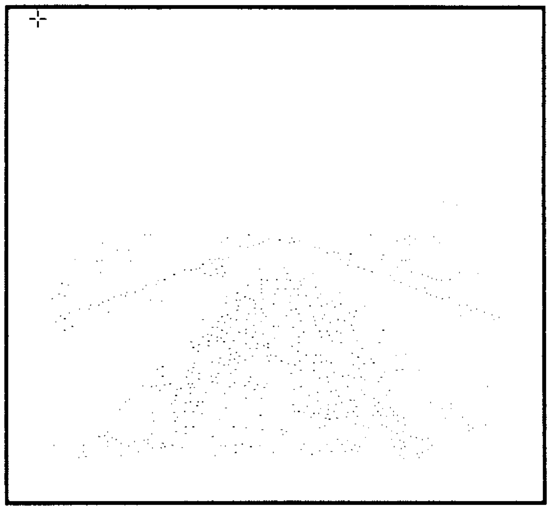
Unlike the approach described in [47] for characterizing the computer vision algorithms in terms of the six components of protocol (i.e., modeling, annotating, estimating, validating, propagating, and optimizing), we evaluate the algorithm in terms of the general function for which it is being used. For example, the optical flow algorithms are developed by applying the optical flow constraint equation within a local region. Thus, the ideal input

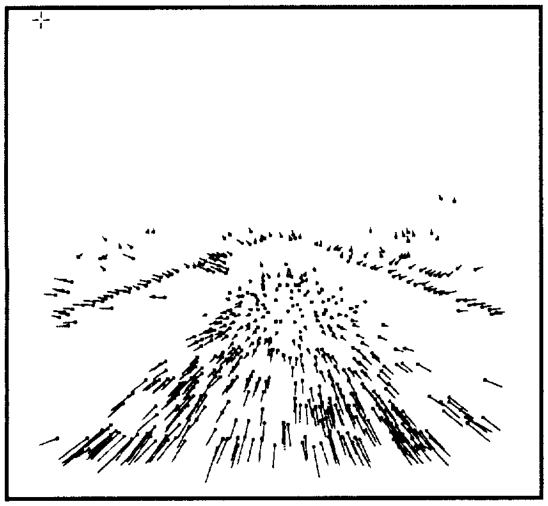

(a) (b)

(c) (d)

Figure 33: Results obtained for a simulated landing image sequence: (a)Frame 3 of image sequence *Landing_normal_32L* (b)Detected Features (c)Computed optical flow (d)Optical flow vectors detected as due to obstacles


(a)


(b)


(c)

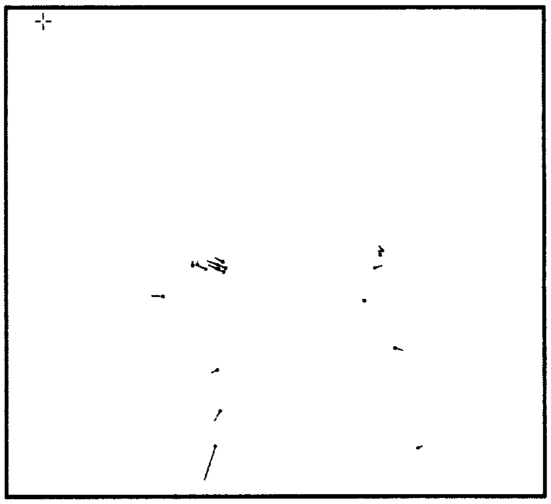

(d)

Figure 34: Results obtained for a real image sequence: (a)Frame 60 of image sequence *runway_crossing_new* (b)Detected Features (c)Computed optical flow (d)Optical flow vectors detected as due to obstacle

to this algorithm requires generation of data according to the following constraint equation within a local neighborhood.

$$I_x u + I_y v = -I_t \tag{53}$$

where I_x, I_y, and I_t are the spatial and temporal gradients computed using five or more images. The output of the algorithm is the least square solution to the above equation within the local region. Hence, according to [47], model inputs are gradient values with proper noise models for error in gradient computation. The gradient masks for the computation of gradient, and the local region size within which the constraint is applied, are the tuning parameters of the algorithm.

Optical flow algorithms have never been evaluated by designing a protocol as suggested in [47], partly because of the difficulty in creating the ideal data or proper modeling of the noise functions associated with the derivative computation, or perhaps in annotating or gathering a representative data set. The computation model underlying the algorithm for optical flow involves assumptions about velocity smoothness, single motion, constant illumination, etc., which are violated, to a large extent, in real imaging situations. Optical flow is the apparent motion of brightness pattern in the image and is only approximately equivalent to the 2-D projection of 3-D velocity.

Various optical flow algorithms were evaluated, and quantitative measures such as flow density, mean, and standard deviations in error were reported in [10]. Unlike the approach described in [47] for characterizing the computer vision algorithms, Barron et al. [10] do not characterize the algorithms by selecting proper model for input, output, and perturbation in input and output datasets. They used a set of image sequences to compare the performance, and used angular deviation between the computed and the true displacement vector as the performance metric.

Our objective has been to evaluate the optical flow algorithm and motion estimation algorithms together, in terms of their ability to recover the motion model parameter from a given sequence of images. Hence, the input to this whole vision system is the sequence of images which are input to the optical flow algorithm. The output of this system is motion and structure parameters in the form of an eight-parameter vector $\bar{a}_p$ and the covariance of the estimated parameter vector Σ_a. To create the image sequences, we assumed that the camera was moving in front of a planar surface with sinusoidal and square texture. Using the known camera motion and plane parameter, we computed the actual (theoretical) planar motion model parameter vector $\bar{a}_t$ and the theoretical optical flow (u_t, v_t) at every point on the image. The optical flow algorithm computed the optical flow (u_p, v_p) and the covariance of the estimated optical flow Σ_u.

We evaluated our optical flow algorithm using synthetic images of planar surfaces in motion. Since the covariance of the estimated optical flow will be used as weighting function in the segmentation stage, we feel that quantitative evaluation of the optical flow should consider this weighting function. We use the Mahalanobis distance between the true optical flow and the estimated optical flow as the performance measure, which is given by

$$E_u = (u_p - u_t)\, \Sigma_u^{-1}\, (u_p - u_t)^T \tag{54}$$

Input to the algorithm will be image sequences obtained using a camera moving in front of a planar surface. We used square and sinusoidal texture for the plane. A sequence of

15 images was obtained for different velocities of the camera. Figures 35 and 36 show frames 1, 5, 10, and 15 of a 15-frame image sequence simulating a camera moving with a velocity of (15, 0, 0) in front of a planar surface, where the camera was at a height of 100 feet from the planar surface with an inclination of 60° with respect to the ground plane. These image sequences were input to the optical flow algorithm. Using the planar motion model and known camera geometry, a parametric form for optical flow was derived. Using this parametric form, the theoretical optical flow was computed at pixels where practical optical flow using the flow algorithm was available. The Mahalanobis distance Eu between the theoretical optical flow and the practical optical flow at pixel (x, y) on the image plane was computed.

Figures 37(a) and 38(a) show the probability distribution of pixels n_i with error $E_u(i)$ as a percentage of the total number of available optical flow values N. Since E_u is a continuous function, the E_u axis has been quantized at discrete intervals. Figures 37(b) and 38(b) show the cumulative distribution of pixels with error E_u, where, at any point $E_u(i)$ along the X axis, the corresponding Y axis represents the total number of pixels with $E_u \leq E_u(i)$. These plots show the likely ratio of good points and the goodness of those points in the form of Mahalanobis distance. Although the exact number of points may vary, the distribution remains almost identical for different image sequences.

The second stage of our algorithm uses these optical flow vectors to recover the model parameter. We consider the model recovery algorithm as a black box which receives sets of optical flow vectors as input and outputs the model parameter that best fits the data points input to the unit. The optical flow vectors were already grouped into bins of decreasing accuracy in the earlier stage. These bins are input to the model recovery algorithm in steps; at every step, a new set of data is added to the earlier set. Model recovery is done first, using the set of data points which are most accurate. As each new set of data points is added, the performance of the algorithm degrades. Let the recovered model parameter at each step be given by the vector $\bar{a}_p$, with covariance of the estimation given by Σ_a. Let the theoretical parametric model using which the images were created be $\bar{a}_t$. We use the Mahalanobis distance between the theoretical and practical model parameters as the performance metrics, as given by

$$E_a = (\bar{a}_p - \bar{a}_t)\, \Sigma_a^{-1}\, (\bar{a}_p - \bar{a}_t)^T \tag{55}$$

Figures 37(c) and 38(c) show the plot of variation of E_a with E_u.

4.6 Summary

In this section, we proposed a new motion-based segmentation algorithm for detecting obstacles in a sequence of images obtained from a video camera mounted on-board a landing aircraft. Unlike the PMMW images, images obtained using a video sensor are of high resolution and high quality. Features like tire marks, runway markers, etc., are clearly visible in these image sequences. Thus, a simple histogram-based thresholding based on homogeneity assumptions cannot be used. We have assumed that the runway is perfectly planar or is piecewise planar and used a planar motion model to group features of the runway. Optical flow identifying objects of nonzero height with reference to the runway plane – stationary or moving – violates the planar motion model; these are detected as obstacles.

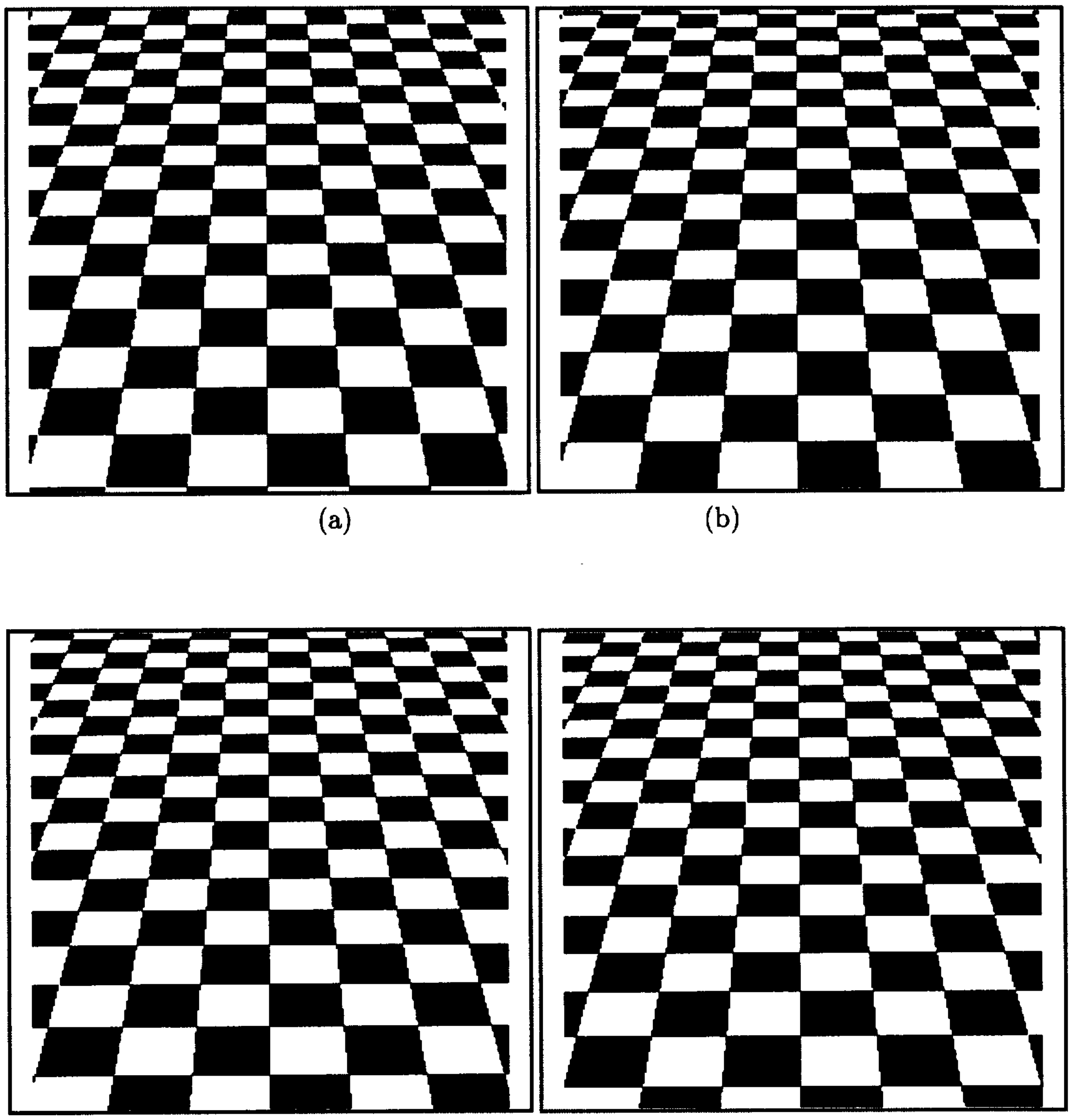


Figure 35: Simulated images of planar surface in motion: (a)Frame 1 (b)Frame 5 (c)Frame 10 (d)Frame 15

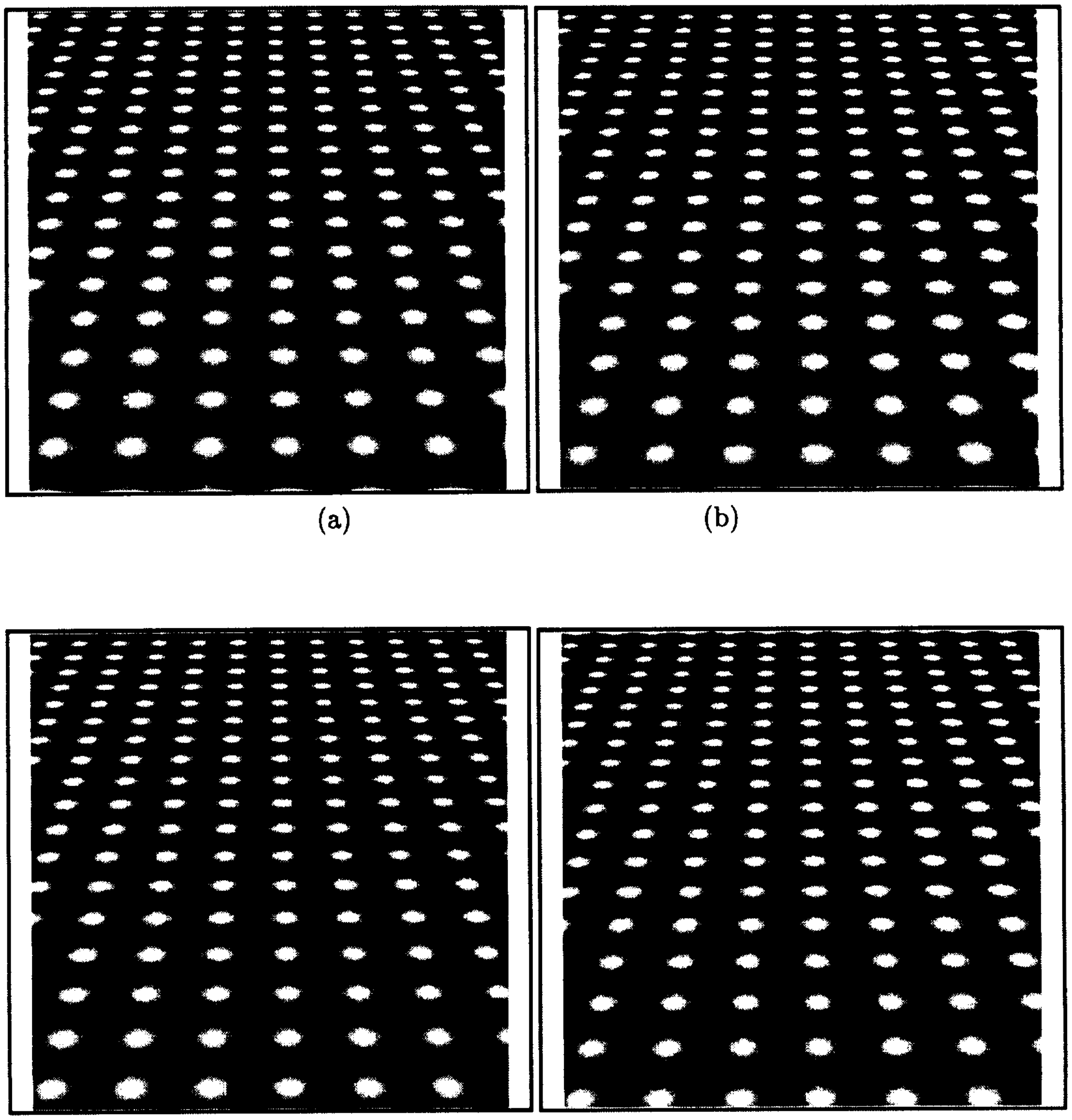
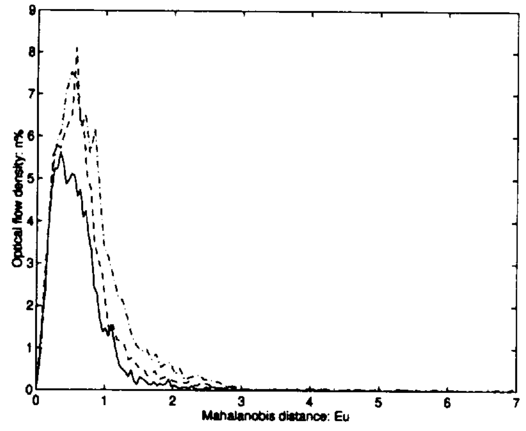
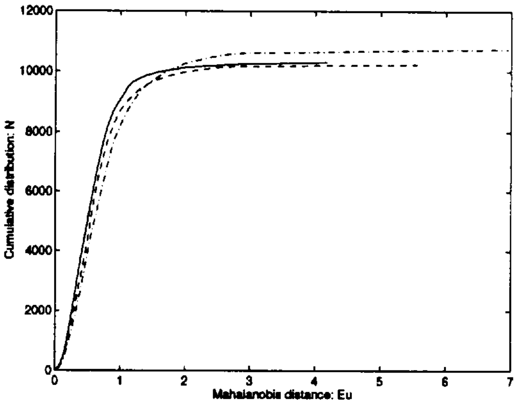

(a) (b) (c) (d)

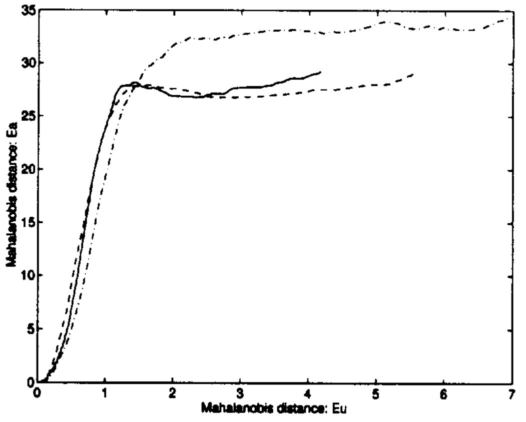
Figure 36: Simulated images of planar surface in motion: (a)Frame 1 (b)Frame 5 (c)Frame 10 (d)Frame 15



(a)

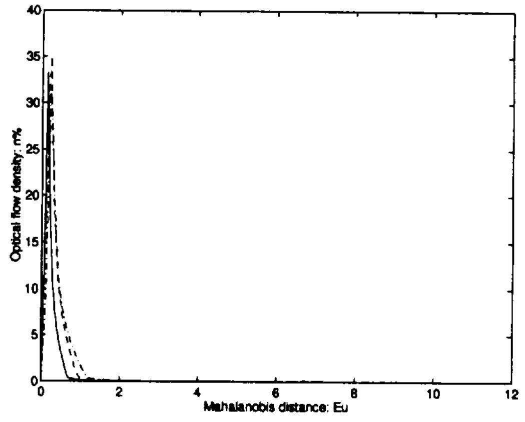


(b)

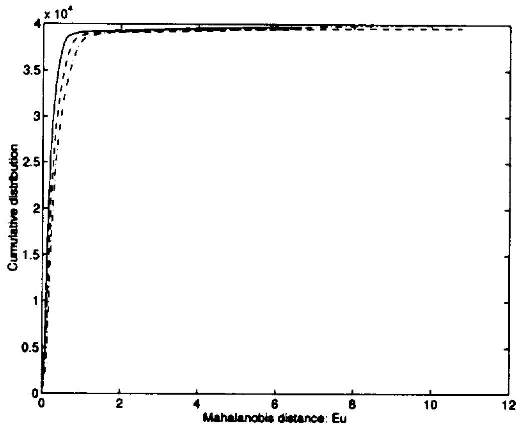


(c)

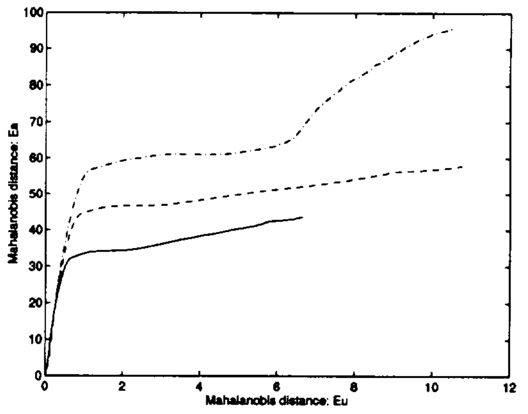
Figure 37: Performance plots: (a)Optical flow density vs. E_u (b)Cumulative distribution of flow vector vs. E_u (c)E_a vs E_u



(a)



(b)



(c)

Figure 38: Performance plots: (a)Optical flow density vs. E_u (b)Cumulative distribution of flow vector vs. E_u (c)E_a vs E_u

To compute the optical flow, we proposed combining both feature-based and flow-based approaches in a hierarchical framework. We computed the optical flow at corner-like feature points, where the optical flow was expected to be more reliable, compared to regions of uniform intensity and edge regions. Features are detected using the SUSAN corner detector, and optical flow is computed using the Maximum Likelihood Estimate based on an optical flow constraint equation applied to a small local region around the feature. Since the optical flow due to ego-motion of the camera is large, we proposed a hierarchical approach using a Gaussian pyramid for computing the optical flow. This makes our approach applicable to general situations where ego-motion and structure of the background are unknown, unlike warping approaches, which require prior knowledge about motion and structure to detect obstacles by compensating for the ego-motion.

The motion-based segmentation algorithm proposed in this work was based on a planar motion model. It starts by hypothesizing that every flow vector is due to a single planar motion. Least square model fitting was used to compute the best model parameter for the set of optical flow vectors computed in the previous stage. The hypothesis was verified by computing the Mahalanobis distance between the computed optical flow and the estimated model flow. Optical flow vectors not satisfying the estimated planar motion model were identified as outliers and rejected. The model parameters were recomputed without the outliers. If the initial hypothesis failed (too many outliers), the solution was dropped and the image was split into four regions. The planar motion model was then applied to each of the regions recursively. We tested our algorithm using both synthetic and real images.

In our proposed approach for obstacle detection by motion-based segmentation, in addition to segmentation by hypothesizing the background to be a planar surface we also computed the motion and structure parameters in the form of an eight-parameter vector $\bar{a}$. Actual camera motion parameters and the plane parameter can be estimated from the model parameter vector; however, computation can be done only to a scale factor. This is a fundamental problem in any motion and structure estimation method that uses monocular image sequences. If either the camera motion parameter or the plane parameter is known, then the other parameter can be computed uniquely. For segmentation, however, estimation of these parameters is not required.

Adiv's Hough-based approach had a similar objective; however, in his method, prior knowledge of range of parameters was required to create the Hough space. The accuracy of his approach depended on the resolution of the Hough space. Segmentation using the Hough method is very time consuming. In general, to form a Hough space of six dimensions (Adiv used affine model, and used only first order terms in the flow equations) with a resolution of B bins in each dimension will require $\bigcirc(B^6)$ memory space. Voting by N features into this Hough space will require $\bigcirc(B^6N)$ computer operation. To improve accuracy, resolution needs to be increased, resulting in exponential growth both in required memory space and run-time. Our algorithm requires formation of $P \times P$ matrix (where $P = 8$, the dimension of the parameter vector $\bar{a}$) using N features in $\bigcirc(N)$ time and computing the inverse of the matrix using $\bigcirc(P^3)$ run-time algorithms. (Note that, though X is a $2N \times P$ matrix, matrix X^TX is of dimension $P \times P$ and can be computed directly without having to form matrix X in only $\bigcirc(N)$ time.) The overall running time for recovery of motion model parameter in one iteration is linear in number of features N. Detecting outliers in an iteration requires only $\bigcirc(N)$ operations. Thus, detecting an obstacle by applying our algorithm to a group of N features in K iterations takes only $\bigcirc(KN)$ computer operations.

The algorithm was able to detect an obstacle in a planar background by segmenting the dominant ego-motion and identifying the outliers with respect to the ego-motion. If the image contains too many objects, however, our algorithm might fit a single motion model to all the motion vectors resulting from such image sequences. Splitting the image can localize such motions, although too many splits could result in an insufficient number of features to compute the motion model parameters reliably. One approach might be to use a high resolution Hough space for each local region, where the range of the Hough space is decided by the model parameter estimated using our algorithm.

Density of feature vectors is also critical to the success of our algorithm. If the planar surface is not textured, our optical flow algorithm will result in sparse flow vectors. The performance of the planar motion model recovery approach will degrade as the ratio of potential outliers (points from the obstacles as well as noisy flow vectors) to points from the planar surface increases. One obvious solution is to use a conventional optical flow algorithm, which produces dense optical flow vectors.

We have also evaluated the optical flow algorithm and the motion model recovery method using least square fit. Since the objective of the approach is to recover the planar motion model from a sequence of images, input images are created using a known model and recovered model parameters are compared to the input model parameter. Barron et al. used angular deviation between the known optical flow and the computed optical flow as the performance metric. We used Mahalanobis distance between the true optical flow and computed optical flow as the performance measure. Optical flow vectors were sorted based on the Mahalanobis distance. These optical flow vectors were used to recover the model parameters. Model parameters thus recovered were evaluated using the Mahalanobis distance between the true model parameter and the recovered model parameters.

5 Structure and Motion Estimation from Line Features

In the motion-based segmentation approach described in Section 4, optical flow is computed at corner-like feature points. Based on the computed optical flow, feature points are sorted as belonging to separate planar surfaces in motion by computing the planar motion model in the form of an eight-parameter vector $\bar{a}$, then measuring the individual flow vector's fitness to the model. The actual motion and structure parameters can be computed from the model parameter vector $\bar{a}$. The motion and structure parameters include the relative motion between the camera and the planar surface, and the plane parameters. In case of monocular image sequences, the motion and structure parameters can be computed only to a scale factor, because a camera moving with a velocity $\vec{V}$ in front of a planar surface at distance d and a camera moving with a velocity $k\vec{V}$ in front of the planar surface at distance kd, produces the same optical flow on the image plane, where k is a scale factor. However, if one of the two data are available (i.e., camera motion or plane parameter), the other parameter can be uniquely determined.

In this section, we propose a method for computing 3-D position and velocity using line features. Most of the point features used in the previous experiments were due to the texture created by the tire marks, and not to true physical corners. Line features are due to the runway markers, and are prominent and robust. First, we propose a method for computing the plane parameters from line features, assuming that the line features in an image correspond to lines on a 3-D plane and that the camera motion is known. We also propose an algorithm for estimating the position and velocity of 3-D objects using line features, assuming the object is moving on a planar surface. Since the end points of a line cannot be detected reliably, we propose an approach for matching and tracking line segments in image sequences.

5.1 Estimation of Plane Parameters

Few methods for computing motion and structure using line features have been proposed in the past [25, 37, 50, 68, 67, 103, 129, 133]. In general, a minimum of three views are required to solve motion and structure problem using line features, unlike the two frames required in case of point features. When the lines are on the same plane, however, two views are sufficient [114]. We propose a new method for computing the plane parameter from line features using the knowledge of the camera motion and assuming that line features result from the 3-D lines on a single planar surface.

5.1.1 Analysis

Fig. 39 shows the imaging geometry used in this analysis. The world coordinate system (WCS) is represented by the three mutually perpendicular axes X, Y, and Z. C and C' are positions of the camera at two different time instants t_1 and t_2. The camera coordinate system (CCS) is fixed to the camera at the optical center and is represented by three axes X_s, Y_s, and Z_s, with X_s pointing in the direction of the optical axis of the camera. The image plane is situated at a distance f, the focal length of the camera, from the optical center, with axes x and y. L_1 is a line on a planar surface whose equation in the WCS is given by $AX + BY + CZ + D = 0$. Let s_1 and s_1' be the line segments in the image plane corresponding to the line L_1 with the camera center located at C and C' respectively. Let

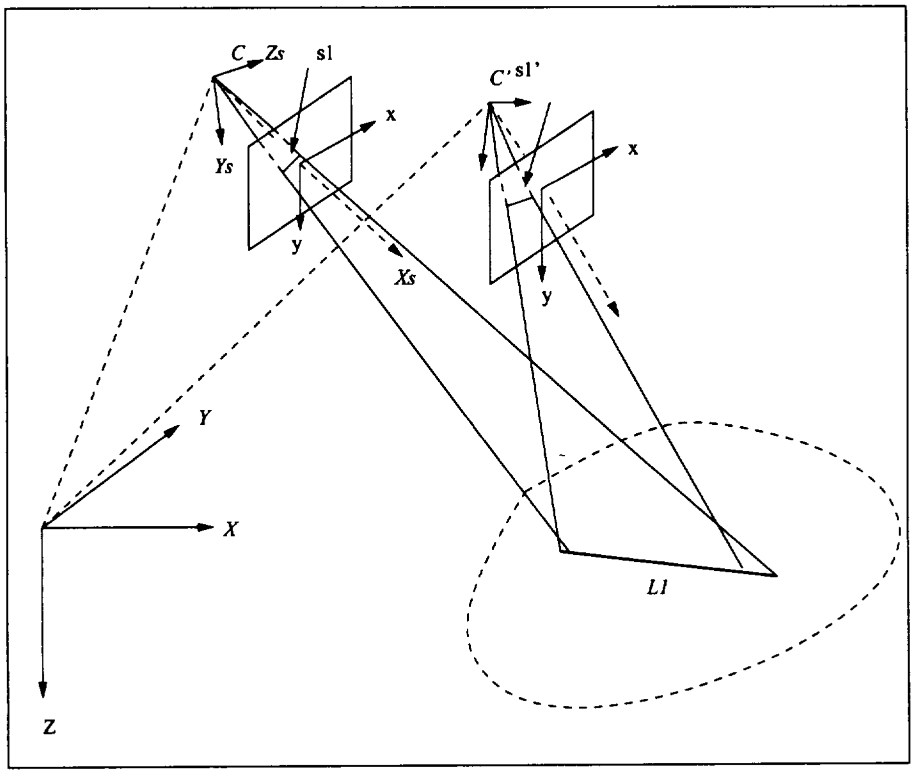


Figure 39: Camera Geometry: s_1 and s_1' are images of line L_1 at two different time instants

the equations of the line segments s_1 and s'_1 in the image plane be

$$\begin{aligned} ax + by + c &= 0 \\ a'x + b'y + c &= 0 \end{aligned} \tag{56}$$

The camera center C and the line segment s_1 form a plane P_1, which intersects the plane P_2 formed by the camera center C' and the line segment s'_1 at the 3D line L_1. The equations of these projecting planes in the respective CCS are given by

$$\begin{aligned} aX_s + bY_s + cZ_s &= 0 \\ a'X'_s + b'Y'_s + c'Z'_s &= 0 \end{aligned} \tag{57}$$

Note that $(a,\ b,\ c)$ and $(a',\ b',\ c')$ are the normal to the planes P_1 and P_2, respectively, in the respective CCS. If R_{wc} and R'_{wc} are the rotation vector from WCS to the CCS at C and C', then normal $\vec{n}_1$ and $\vec{n}'_1$ to the projection plane P_1 and P_2 in the WCS is given by

$$\begin{aligned} \vec{n}_1 \equiv [a_1\ b_1\ c_1]^T &= R_{wc}^{-1}\,[a\ b\ c]^T \\ \vec{n}'_1 \equiv [a'_1\ b'_1\ c'_1]^T &= R_{wc}'^{-1}\,[a'\ b'\ c']^T \end{aligned} \tag{58}$$

These two planes intersect to form the line L_1 in the WCS. The direction vector of this line is given by the cross product of $\vec{n}_1$ and $\vec{n}'_1$:

$$L_1 \equiv [b_1c'_1 - b'_1c_1,\ a'_1c_1 - a_1c'_1,\ a_1b'_1 - b_1a'_1]^T \tag{59}$$

The equation of the plane P_3 normal to L_1 is:

$$(b_1c_1' - b_1'c_1)\,X + (a_1'c_1 - a_1c_1)\,Y + (a_1b_1' - b_1a_1')\,Z + t = 0; \tag{60}$$

where t is a constant. The planes P_1, P_2, and P_3 intersect along the line L_1 whose equation is given by

$$\begin{pmatrix} a_1 & b_1 & c_1 \\ a_1' & b_1' & c_1' \\ b_1c_1' - b_1'c_1 & a_1'c_1 - a_1c_1' & a_1b_1' - b_1a_1' \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ t \end{pmatrix}$$

$$[A] \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ t \end{pmatrix}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = A^{-1} \begin{pmatrix} d_1 \\ d_2 \\ t \end{pmatrix} = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + \begin{pmatrix} l \\ m \\ n \end{pmatrix} t \tag{61}$$

where (X_i, Y_i, Z_i) is a point on the line L_1, (l, m, n) is the direction vector of the line, and t is a constant. From corresponding line segments s_1 and s_1' in two frames, we can compute the line segment L_1. If L_1 and L_2 are two lines in the same 3-D plane with the equation

$$L_i \equiv \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} + \begin{pmatrix} l_i \\ m_i \\ n_i \end{pmatrix} t \tag{62}$$

where i = 1, 2, then the normal to the plane is given by:

$$\begin{aligned} A &= m_1 n_2 - m_2 n_1 \\ B &= l_2 n_1 - l_1 n_2 \\ C &= l_1 m_2 - l_2 m_1 \end{aligned} \tag{63}$$

This will fail if the two lines in 3-D are parallel. This can be solved by considering a line connecting two points $(X_1,\ Y_1,\ Z_1)$ on L_1 and a point $(X_2,\ Y_2,\ Z_2)$ on L_2. Then the normal to the plane is given by:

$$\begin{aligned} A &= m_1(Z_2 - Z_1) - n_1(Y_2 - Y_1) \\ B &= n_1(X_2 - X_1) - l_1(Z_2 - Z_1) \\ C &= l_1(Y_2 - Y_1) - m_1(X_2 - X_1) \\ D &= (X_2 - X_1)(n_1 Y_1 - m_1 Z_1) + (Y_2 - Y_1)(Z_1 l_1 - n_1 X_1) + (Z_2 - Z_1)(m_1 X_1 - l_1 Y_1) \end{aligned} \tag{64}$$

5.1.2 Feature Detection and Matching

The above approach requires a minimum of two line correspondences in two frames. In this work, we used line features resulting from the runway markers, since they are long, easy to detect, and can be easily tracked. First, edges were detected by applying the Canny edge detector to the input image sequence [19]. To detect the straight lines from the edge image, we used the software package *Object Recognition Toolkit (ORT)* developed by Etemadi et al. [35]. This software package takes an input edge image and reports the detected straight line segments, circular arcs, and various junction between lines, such as the T and Y junctions. In our application, we were interested only in straight line segments. Each line was represented by several parameters including the two end points, mid-point, length, orientation, the variance in length and orientation, etc. Edge images obtained for frames 50 and 55 of the image sequence *runway_crossing_new* using the Canny edge detector are shown in Fig. 40(a) and Fig. 40(b), and the line segments detected by applying the ORT software to the edge image are shown in Fig. 40(c) and Fig. 40(d). Noisy line segments due to tire marks and other small line segments were later ignored by using a proper threshold on the length of the line segments.

The next step was to use proper representation for each line segment and develop a method for matching line segments in the current frame f_1 to the line segments in the next frame f_2 using the representation. It is clear from Fig. 40 that the endpoints and the midpoints which were commonly used in tracking line features were unreliable because the segments can be broken from one frame to another, and that endpoints and midpoints were not located in the image plane and so could not be detected. Therefore, we use the normal representation of line given by the equation:

$$x\ cos\theta + y\ sin\theta = \rho \tag{65}$$

where ρ is the perpendicular distance from the origin to the line, and θ is the angle of the line with respect to the x axis. This representation was used in the detection of line segments using the Hough transform [45] by grouping collinear points of an edge image in the (ρ, θ) parameter space. According to this representation, a straight line in (x, y) space becomes

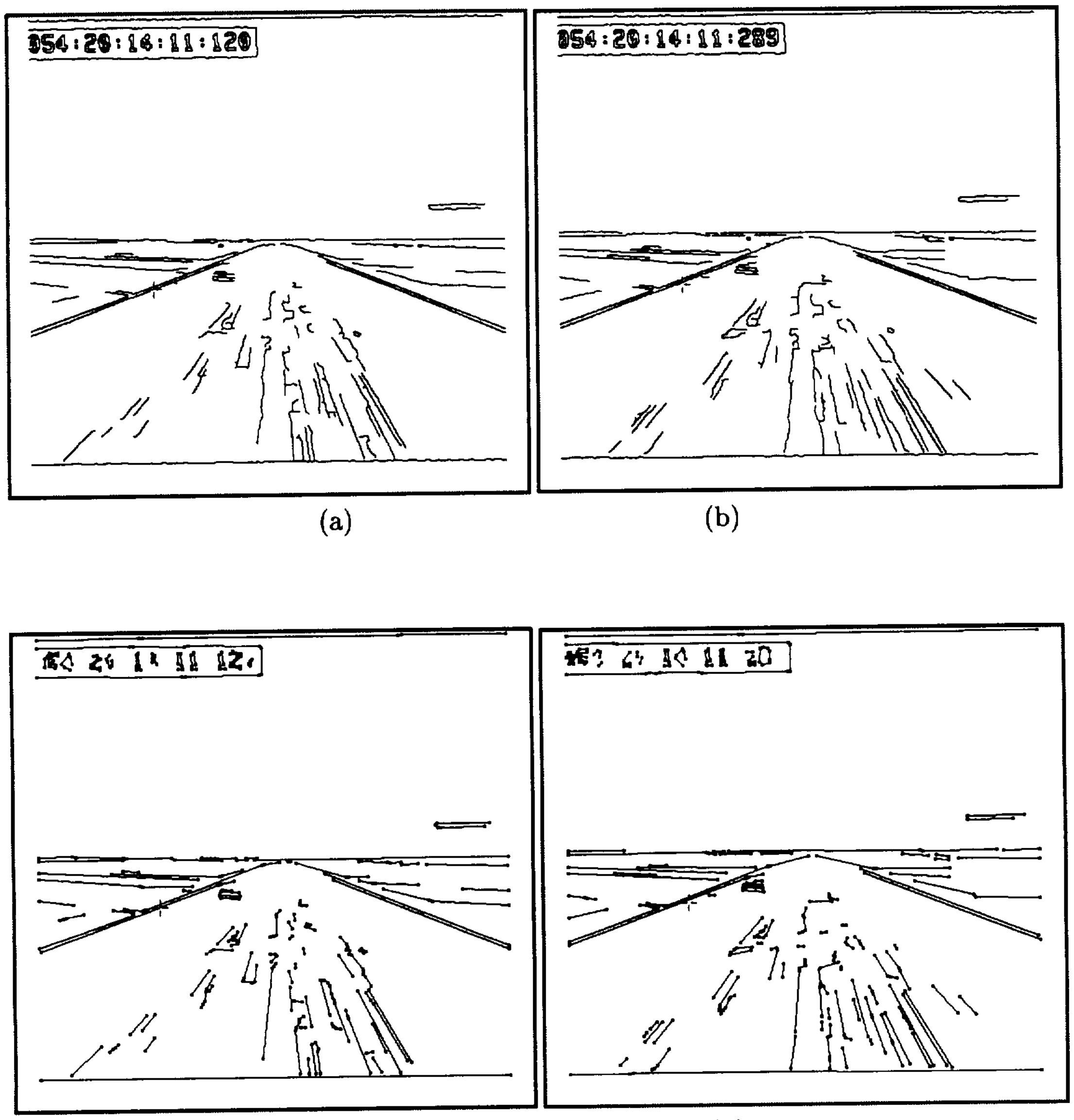


Figure 40: Canny edge image: (a)Frame 50 (b)Frame 55. Line Segments detected using ORT software: (c)Frame 50 (d)Frame 55.

a point in the (ρ, θ) space. Except for collinear lines, each line segment in (x, y) space corresponds to a unique point in the (ρ, θ) space. If the (ρ, θ) parameter space is quantized at N regular intervals in each of the two dimensions, then searching for a matching line is equivalent to searching the bin corresponding to the normal parameter (i.e., ρ and θ) of the given line. Based on the normal representation for line segments, we have developed a three-stage algorithm to find a best match s_i' in frame f_2 for a line segment s_i in frame f_1:

- Compute the prediction s_i^p in frame f_2,
- Search for initial matches using Hough space, and
- Find the best match among the initial matches.

The position of a line segment s_i in frame f_2 was predicted by assuming the position of every pixel along the line by computing the optical flow at those pixels. A least square approach was then used to fit a best line to the predicted positions. Since full optical flow cannot be computed at edge pixels, we computed only the normal flow (i.e., the optical flow in the direction of the gradient) at every pixel along the line segment. Normal flow computation was sufficient for this application, since we were interested only in matching the line segments, and not the whole line. Thus, the motion of the line in the direction of the line segment was not required. In addition, normal flow could be computed more accurately than full flow, since computation of normal flow does not need additional assumption such as the smoothness constraint used in the computation of full optical flow. Normal flow at any pixel (x, y) in the image plane can be computed immediately from the optical flow

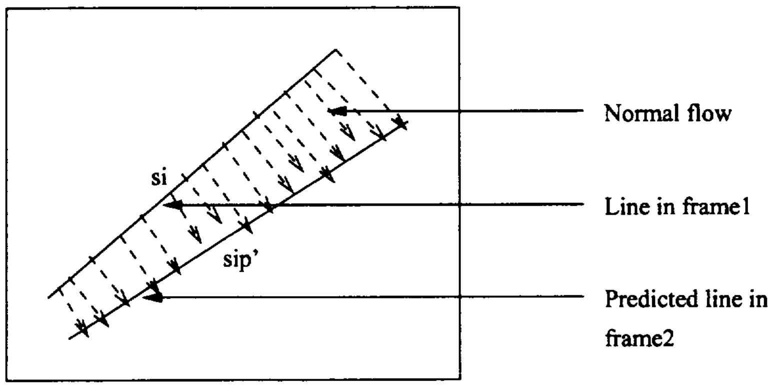


Figure 41: Predicting the line in the next frame using normal flow

constraint equation as:

$$u_n = -\frac{I_t}{\sqrt{I_x^2 + I_y^2}} \tag{66}$$

where I_x, I_y, and I_t are spatial and temporal gradients. Using the computed normal flow at every pixel along the line segment s_i in frame f_1, the pixel positions for the corresponding line s_i' in the next frame can be predicted as shown in Fig. 41. A best line was then fit to the predicted pixel positions in frame f_2 to compute the predicted line segment s_{ip} [33].

Predicted line s_{ip} is only an approximation to the actual line segment s_i' in frame f_2. However, the prediction can be used to minimize our search for the best match in frame f_2. To search for possible candidates for matching, we first map every line segment from frame f_2 to a $N \times N$ Hough space. Each line segment detected in frame f_2 votes to a separate bin in the two-dimensional Hough space, based on the ρ and θ values of the line segment. Hough parameters are also computed for the predicted line segment s_{ip}. Under ideal conditions, when there is no noise, the predicted line segment s_{ip} will be aligned with the actual matching line segment s_i' in f_2 and, hence, will map to the correct bin. However, due to error in the process of edge detection, line detection, normal flow computation, line fitting, etc., the predicted line parameter may not be exactly equal to that of the actual line segment found in frame f_2. Hence, a search space of 3×3 around the predicted parameter in the Hough space was used to find the matching line segment as shown in Fig. 42(a). Sometimes multiple matches could be found within the search region. Another stage of filtering might be necessary to find the best match.

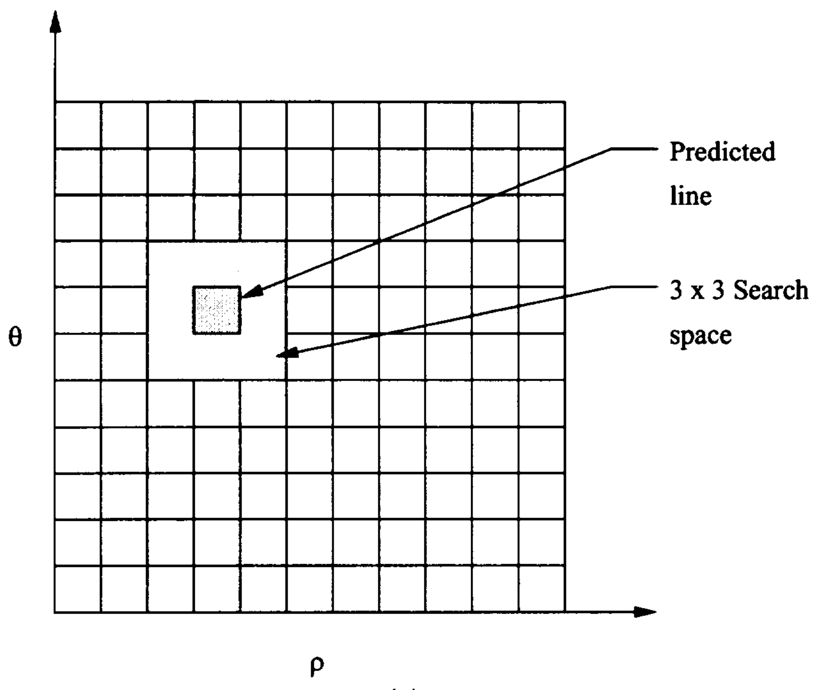
If multiple matches were detected, a best match was found by computing the spatial proximity of the predicted line to the actual line segment, as shown in Fig. 42(b). Perpendicular distances from the the endpoints of the predicted line to the each of the line segments matched in the Hough space were computed. If (x_p, y_p) and (x_q, y_q) are the endpoints of the predicted line, and $a_i x + b_i y + c_i = 0$ is the equation of the ith matched line using the Hough space, the distance measure for the ith matched line is computed as:

$$\begin{aligned} d_i &= d_p + d_q \\ &= \frac{a_i x_p + b_i y_p + c_i}{\sqrt{a_i^2 + b_i^2}} + \frac{a_i x_q + b_i y_q + c_i}{\sqrt{a_i^2 + b_i^2}} \end{aligned} \tag{67}$$

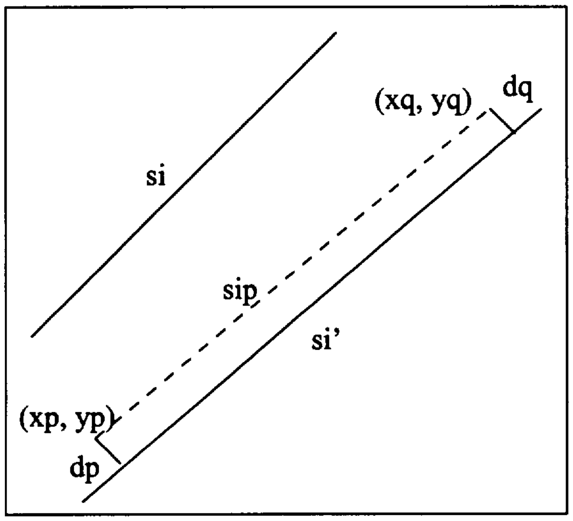
The line segment s_i' with minimum distance d_i is selected as the best match.

5.1.3 Discussion

We tested our proposed approach for computing the plane parameter for the runway using the line features due to the runway markers and known camera motion. A real image sequence, *runway_crossing_new*, was used in this experiment. Using this sequence, we conducted two tests to explore the feasibility of the proposed approach. In the first test, we



(a)



(b)

Figure 42: Matching line segments: (a)Initial match: A 3 × 3 search region is used around the predicted parameter value in the hough space. (b)Final match: Perpendicular distance at the two end points are computed

used the line segments resulting from the runway markers; in the second test, we used the line features resulting from the taxiway. Although line features could be matched accurately, the computed plane parameters were not correct. Failure of the proposed approach can be attributed to either the degenerating case arising from the actual imaging geometry when the camera is moving in the direction of the 3-D line, or the inability to resolve a small angle between the projecting planes when the direction of motion of the camera is at 90° to the 3-D line.

Consider a situation where an aircraft is flying at a height of $50ft$ and approaching for landing at an angle of 3° to the runway. In this situation, the runway markers - - which are in the direction of the optical axis of the camera - - are almost parallel to the camera motion. The projecting planes P_1 and P_1' due to the 3-D line L_1 and its corresponding image segment s_1 and s_1' at two positions of the camera C and C' are almost aligned. Therefore, normals to these planes are identical, resulting in a degenerate case.

The best line features to consider were those that were at 90° to the direction of camera motion. In the above landing situation, the range to the aim point is $\approx 950ft$. Without loss of generality, we assume a cross runway at a range of $300ft$. from the current position of the aircraft, and the velocity of the aircraft $60ft/sec$ in the direction of motion. Our aim is to find the angle between the projecting planes resulting from the edge of the cross runway at two different camera positions.

To make the worst case analysis, we will consider two frames that are one second apart. From simple trigonometry, we can show that, when the aircraft is at a height of $50ft$, the projecting plane due to the cross runway is at an angle of $\approx 9.5°$ to the runway plane intersecting the runway plane $300ft$ from the aircraft. After 1 second the same cross runway will be $\approx 240ft$ from the aircraft, and the projecting plane due to the cross runway will be at an angle of $\approx 11.1^\circ$ to the runway plane. This means that the two projecting planes P_1 and P_1' 30 frames apart resulting from a 3-D line L_1 at a range of $300ft$ from the initial position of the aircraft on a planar runway are at an angle of only 1.6° to each other. It is quite impossible to resolve this angle with all the error in line detection, optical flow computation, etc.

5.2 Estimation of 3D Position and Velocity

Although image regions due to moving objects can be detected using the motion-based segmentation algorithm described in Section 3, estimating the position and velocity of such objects using a monocular image sequence is still an ill-posed problem. If the motion of the camera is known and the object is stationary, the position of the object in 3-D can be computed using motion stereo. The range estimation algorithms for estimating the distance to static objects using monocular image sequence, described in [108, 109, 110] use motion stereo to compute the initial range to the object. This initial value is used to start a Kalman filter, which recursively refined the estimate by including the new observations obtained by tracking the object in subsequent frames. If both the camera and the object are in motion, then the solution for position and velocity of the object is not unique - - even when the position and velocity of the camera are known accurately. Two possible solutions to this problem are either to apply certain constraints on the object motion (*constraint-based solution*) or to use stereo image sequences (*binocular stereo* or *trinocular stereo*).

In the constraint-based solution approach, the object is assumed to be moving on a known terrain (i.e., $X = f(Y, Z)$). Hence, the range, X, to the object is known, and the object's velocity in the direction of range is constrained by the equation:

$$V_X = f_Y\left(Y, Z\right) V_Y + f_Z\left(Y, Z\right) V_Z$$

where f_Y and f_Z are the partial derivatives of $f\left(Y, Z\right)$ and V_X, V_Y, and V_Z are the X, Y, and Z components for the velocity vector. Since X and V_X are constrained, the position and velocity of the object in 3-D can be uniquely computed using two frames and a Kalman filter similar to those described in [108, 120] can be used to refine the estimate by tracking image features in subsequent frames.

The stereo method uses two or more cameras and computes the range to the object by triangulation of corresponding image points in two frames. A Kalman filter can be initialized with the position and velocity estimate obtained from the initial frames. The estimate can be refined by considering additional measurements obtained by tracking the features in subsequent frames.

In this research, we use the constraint-based solution approach. We assume that the object is moving on a planar surface. Given the camera's motion and the feature locations in successive images, estimation of the motion parameters of the tracked object can be formulated as a state estimation problem using a Kalman filter. The position and velocity of the camera are known from the GPS and INS, and the initial estimate of the plane parameter is available from the motion-based segmentation algorithm described in Section 4 or the line-based estimation approach described in Section 5.1. The Kalman filter improves the initial estimate by combining the redundant measurements obtained by tracking the features in subsequent frames. The modeling of the motion kinetics and the derivation of the Kalman filter based on the model are described in the following section.

5.2.1 Analysis

We assume that the object is moving on a planar surface. The motion of the object in the 3-D space is purely translational with constant velocity $V = (V_X, V_Y, V_Z)$ and the equation of the plane is $n_X X + n_Y Y + n_Z Z + d = 0$. The velocity, V_X, in the direction of range is given as:

$$V_X = \frac{-n_Y V_Y - n_Z V_Z}{n_X} \tag{68}$$

The equation of the plane in the camera coordinate system is given by:

$$k_1 X + k_2 Y + k_3 Z = 1 \tag{69}$$

Plane parameter $\bar{k} = [k_1\ k_2\ k_3]$ in the CCS can be computed as:

$$\bar{k} = -\frac{R\bar{n}}{\bar{T} \cdot \bar{n} + d} \tag{70}$$

where $\bar{T}$ is the translation of the camera and R is the rotation matrix of the camera at time t_1 with respect to the origin of the WCS and is given as:

$$\bar{T} = \begin{pmatrix} T_X \\ T_Y \\ T_Z \end{pmatrix}, \; R = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \tag{71}$$

Let (X, Y, Z) be the coordinates of a point P on the object at time t_1 in the camera coordinate system (CCS). The image coordinate of the point at t_1 is given as:

$$x = f\frac{Y}{X};\ y = f\frac{Z}{X} \tag{72}$$

The coordinate of the same point at time t_2 ($t_2 = t_1 + \tau$, where τ is the time interval between the two consecutive frames) in the CCS is given by:

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_X - \tau V'_X \\ t_Y - \tau V'_Y \\ t_Z - \tau V'_Z \end{pmatrix} \tag{73}$$

where $[t_X,\ t_Y,\ t_Z]^T$ is the translation of the camera in CCS at time t_1. The velocity vector $\bar{V}' = [V'_X,\ V'_Y,\ V'_Z]^T$ for the object at time t_1 in the CCS is given by $\bar{V}' = r\bar{V}$. The image coordinates of the point at time t_2 is given by:

$$\begin{aligned} x' &= f\frac{(r_{21}X + r_{22}Y + r_{23}Z) + (T_Y - \tau V'_Y)}{(r_{11}X + r_{12}Y + r_{13}Z) + (T_X - \tau V'_X)} \\ y' &= f\frac{(r_{31}X + r_{32}Y + r_{33}Z) + (T_Z - \tau V'_Z)}{(r_{11}X + r_{12}Y + r_{13}Z) + (T_X - \tau V'_X)} \end{aligned} \tag{74}$$

Substituting from 69 and 72 in equation 74 we get

$$\begin{aligned} x' &= \frac{a_1 f + a_2 x + a_3 y}{a_7 f + a_8 x + a_9 y} \\ y' &= \frac{a_4 f + a_5 x + a_6 y}{a_7 f + a_8 x + a_9 y} \end{aligned} \tag{75}$$

where

$$a_1 = r_{21} + (T_Y - \tau V_Y')\, k_1,\ a_2 = r_{22} + (T_Y - \tau V_Y')\, k_2,\ a_3 = r_{23} + (T_Y - \tau V_Y')\, k_3$$

$$a_4 = r_{31} + (T_Z - \tau V_Z')\, k_1,\ a_5 = r_{32} + (T_Z - \tau V_Z')\, k_2,\ a_6 = r_{33} + (T_Z - \tau V_Z')\, k_3$$

$$a_7 = r_{11} + (T_X - \tau V_X')\, k_1,\ a_8 = r_{12} + (T_X - \tau V_X')\, k_2,\ a_9 = r_{13} + (T_X - \tau V_X')\, k_3 \tag{76}$$

Defining the state vector $S = (n_X\ n_Y\ n_Z\ V_Y\ V_Z)$ and the measurement vector $M = (m_1\ m_2) = (x\ y)$, the state equation and measurement equation in discrete time system can be expressed as:

$$\begin{aligned} S_{k+1} &= \Phi_k S_k + w_k \\ M_k &= m(S_k) + v_k \\ &= [m_1(S_k),\ m_2(S_k)] + v_k \end{aligned} \tag{77}$$

where the state transition matrix Φ_k is a Unity matrix, and w_k and v_k are the process and measurement noise respectively. Zero mean Gaussian noise is assumed, such that $w_k \sim N(0, Q_k)$ and $vk \sim N(0, R_k)$. It is clear from Eq. 75 that the measurement equation is a nonlinear function of the state vector. An extended Kalman filter is thus necessary. The measurement equation is linearized about the current estimate of S giving:

$$M_k = H_k\left(\hat{S}_k^-\right) S_k + v_k \tag{78}$$

where

$$\begin{aligned} H_k\left(\hat{S}_k^-\right) &= \left.\frac{\partial m(S)}{\partial S}\right|_{S=\hat{S}_k^-} \\ &= \left.\begin{pmatrix} \frac{\partial m_1}{\partial n_X} & \frac{\partial m_1}{\partial n_Y} & \frac{\partial m_1}{\partial n_Z} & \frac{\partial m_1}{\partial V_Y} & \frac{\partial m_1}{\partial V_Z} \\ \frac{\partial m_2}{\partial n_X} & \frac{\partial m_2}{\partial n_Y} & \frac{\partial m_2}{\partial n_Z} & \frac{\partial m_2}{\partial V_Y} & \frac{\partial m_2}{\partial V_Z} \end{pmatrix}\right|_{S=\hat{S}_k^-} \end{aligned} \tag{79}$$

The Kalman filter consists of two parts.

1. *Measurement Update*: The measurement update is done whenever a new measurement is available. Prior to processing a new measurement M_k, we have the estimated value of the state $\hat{S}$ and the covariance of the estimate $P(k)$, $Q(k)$ and $R(k)$. The new measurement improves our estimate of the state and its covariance. The updated values are

$$\begin{aligned} \hat{S}_k^+ &= \hat{S}_k^- + K_k\left[M_k - m\left(\hat{S}_k^-\right)\right] \\ P_k^+ &= \left[I - K_k H_k\left(\hat{S}_k^-\right)\right] P_k^{-1} \end{aligned} \tag{80}$$

where the Kalman filter gain K_k is computed as

$$K_k = P_k^- H_k^T \left[H_k\left(\hat{S}_k^-\right) P_k^- H_k^T\left(\hat{S}_k^-\right) + R_k\right]^- \tag{81}$$

2. *Time Update*: This part of the filter accounts for the system dynamics, and propagates the state and its covariance matrix until the next measurement is made. The

propagated values are:

$$\begin{aligned} \hat{S}_{k+1}^{-} &= \Phi_k \hat{S}_k^{+} \\ P_{k+1}^{-} &= \Phi_k P_k^{+} \Phi_k^T + Q_k \end{aligned} \tag{82}$$

Starting from Eq. 81, the Kalman filter will run recursively as new measurements are available. The initial estimate for $\bar{n}$ is provided by the segmentation algorithm, and the estimate for $\bar{V}$ is computed by triangulation on the first two frames using the Eq. 75.

5.2.2 Experimental Results

We tested the above tracking and estimation algorithm using the real image sequence *runway_crossing_new*. We used the line features in this experiment. Line features in the image are tracked using their two endpoints. The matching algorithm described in Section 5.1 was used to match the line segments in successive frames. Separate Kalman filter is used to track each of the endpoint of a line. Figures 43(a) and 43(b) shows the zoomed portion of the frames 50 and 51 of the original image sequence containing the truck. Figures 43(c) and 43(d) shows the edge image obtained using the Canny edge detector, and Figures 43(e) and 43(f) shows the ORT software output, with the endpoints of the lines clearly marked on the image. Line segments were tracked using the matching algorithm described in section 5.1.2. The Kalman filter did not converge to the proper value for the motion and structure parameter. This is because line segments were broken from frame to frame, and their endpoints could not localized. Hence estimates done using the endpoints were not correct.

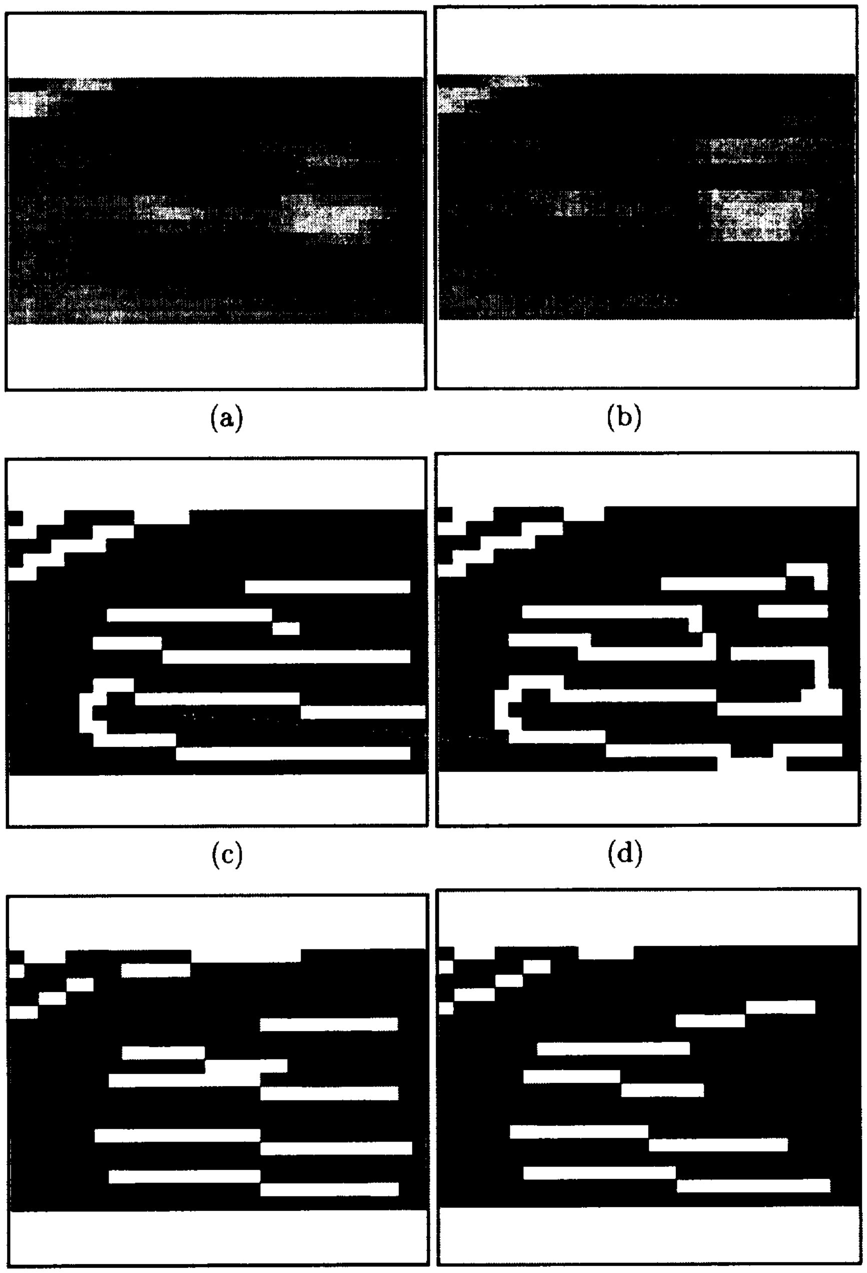

(a) (b) (c) (d) (e) (f)

Figure 43: Tracking line segments: (a), (b) Zoomed portion of the original image containing the truck; (c), (d) canny edge image; (e), (f) ORT software output

5.3 Summary

In this section, we explored the feasibility of using line features for estimating the motion and structure parameters from image sequences obtained from a single moving camera. Line features are prominent, and are expected to be more reliable than point features. Line features due to runway markers or other man-made textures on the runway can be used to estimate the plane parameter of the planar runway. However, due to a very small angle of separation between the projecting planes formed by the line segments and the corresponding runway markers, and error in localizing the image features the plane parameter could not be recovered reliably for this application.

Estimating the motion of an object from a monocular image sequence obtained from a moving camera is a ill-conditioned problem. In this section, we propose solving this problem using a constraint-based solution approach, where we assume that the object is moving on a planar surface and that the motion of the camera is known. We have derived a Kalman filter for computing the plane parameter and the velocity of the object simultaneously. Either point or line features can be used to estimate the motion and structure. We proposed using line features. We tracked the line features using the endpoints. Due to difficulty in matching the endpoints, we were not able to get good results.

We also described algorithms for matching line segments using normal representation. We follow the predict-and-match paradigm, where the line segment is first predicted by computing the normal flow for the line segment. Matching is done in two stages: first, approximate matches are found by searching the Hough space, and then a fine match is performed by using the minimum distance constraint.

6 Conclusions

The ability to detect obstacles in a sequence of images is an essential first step in automation of low altitude flight, landing, takeoff, and taxiing phases of aircraft navigation. In this research, we explored the feasibility of computer vision algorithms for detecting obstacles in monocular image sequences obtained by on-board sensors of two different modalities – a low-resolution PMMW sensor and a conventional video sensor – for incorporation into an aircraft-based Synthetic Vision System (SVS). Using this information and various other data, SVS can complement the ground-based systems in various functions such as runway incursion, and obstacle detection and avoidance.

To facilitate seeing through fog, the SVS is envisioned to be equipped with a PMMW sensor. Because of the low resolution and poor quality of the images obtained using these sensors, we proposed and implemented a model-based recognition approach for detecting image regions corresponding to the runway. A histogram-based thresholding approach was then applied within the runway region for detecting obstacles. To detect obstacles in video image sequences, we proposed and implemented a new motion-based segmentation algorithm. This included a novel combination of feature-based and flow-based approaches within a hierarchical multi-resolution frame work for computing the optical flow, where a gradient-based approach is applied over a small local region around corner-like features. These optical flow vectors are then grouped into different regions by recursively applying the planar motion model on image regions.

Though the main objective of the research was obstacle detection, we also focussed on such other issues as estimating camera position from image-based features and improving camera position information using multiple features. This is useful, since the camera position information available from the GPS is not accurate and the frequency interval of the data is not sufficient for many problem solving methods in computer vision.

We have also explored the use of line features for estimating the motion and structure from monocular image sequences. We have described an algorithm for computing the plane parameter using only line features. Line features due to the runway markers are used. At least two line segments in two frames and knowledge of camera motion parameters are required to compute the plane parameter for the runway. A Kalman filter based recursive algorithm for estimating the motion of an object using point or line features contained within the obstacle region is described.

6.1 Primary Contribution

- Computation of reliable optical flow has been always a difficult task. In this work, we proposed a selective, gradient-based approach to be used at points where the computation of optical flow is expected to be more reliable. This differs from the conventional optical flow algorithm where, theoretically, optical flow is computed at every pixel. Inclusion of too much noisy optical flow would result in large errors in the interpretation stage. Preselecting the reliable data points in advance would not only improve the performance of future computational procedures, but would also improve the execution time of both the optical flow computation algorithm and the interpretation algorithm.

- Field-based approaches fail if the optical flow to be computed is large. In many real life applications, such as aircraft navigation, road navigation, etc., ego-motion of the sensor results in large optical flow. Although feature-based approaches can measure large optical flow, they are computationally very expensive due to the large search space. We propose using our selective, gradient-based approach within a hierarchical framework, and using the Gaussian pyramid to compute large ego-motion. Optical flow computed at lower resolution is predicted to higher resolution, and a corrective flow is computed and added to the original flow.

- We proposed a new recursive motion-based segmentation algorithm which could be used to recover both single and multiple motion. In cases of single motion, the algorithm can tolerate larger error in the optical flow by splitting the image into smaller regions, then detecting the outliers within each region separately. In cases of multiple motions, the proposed framework hypothesizes each motion to follow a planar motion model and uses the split and merge approach to hypothesize and verify these regions.

  The Hough method for multiple motion segmentation requires prior knowledge of the range of the model parameters, and the accuracy of the method depends on the resolution of the Hough space. The warping procedure requires complete knowledge of camera motion and the terrain in which the camera is moving. We did not make any assumptions about the camera motion. We only made use of planar motion model to hypothesize the flow vectors as belonging to one or more region, and verified the accuracy of the hypotheses by comparing the deviation of the computed optical flow from that of the hypothesized model flow.

- Our motion-based segmentation approach for detecting multiple motions and estimating the motion and structure parameter in terms of planar surface motion model parameter is computationally more efficient than the Hough-based approach. In general, to form a Hough space of six dimensions with a resolution of B bins in each dimension, will require $\bigcirc(B^6)$ memory space. Voting by N features into this Hough space will require $\bigcirc(B^6 N)$ computer operation. In general, using our algorithm to segment multiple motions by applying the least square model fit approach recursively on a group of N features in K iterations takes only $\bigcirc(KN)$ computer operations.

- An analytical model for computing the error in camera position estimated using image-based features has been developed as was a set of equations for computing the error in various camera parameters. A new algorithmic approach to calibrate cameras using image-based features is described in this report.

- We have proposed using the Mahalanobis distance as the measure for evaluating the performance of the optical flow algorithm, unlike earlier approaches, which simply compute the density, average error, and the standard deviation to measure performance. Since the theoretical models used for computing the optical flow (optical flow constraint with local smoothness model) differ from the one used in the interpretation stage (planar motion model), we have proposed a new performance characterization which relates the error in estimated optical flow to the error in recovered parameters using these optical flow vectors.

6.2 Future Research

In this report we developed algorithms for processing images from PMMW sensors and video sensors separately. An interesting problem, which has applications, both in robotics and navigation, is integration of information from different sensors. This can be done in two ways:

- First, images from different sensors could be fused into a single composite image, then the fused image could be processed for detection of obstacles and other objects of interest.
- First, a different class of algorithms could be used to process and extract information from each of the sensor images, then the extracted information could be fused.

Such a multi-sensor system can essentially improve the resolution and thereby improve the performance of the segmentation and estimation algorithm. This will also avoid the necessity for switching between the sensors, depending on the lighting and the weather condition.

In this work, we have used only monocular image sequences. It is widely known that it is impossible to estimate the position and velocity of a moving object obtained from a moving camera unless additional constraints (shape and size of the object, terrain information in which the object is moving, etc.) are provided. Use of stereo, however, can provide unique solution. Stereo image processing poses additional problems, such as stereo correspondence, wide enough base distance, etc. More research would be required to use the stereo images for this application.

Finally, there are real time implementation issues. We have evaluated some of the on-the-shelf image processing hardware for implementation into our model-based object detection in PMMW image sequences. Computer vision and image processing problems are computationally intensive. There are two ways to speed up the computation. The first is to build an algorithm-specific architecture, which would require analysis of the algorithms and development of data-flow architecture suitable for the application. The second method is to split the computation among networked work stations or on a multiprocessor system.

References

[1] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 7(4):384–401, 1985.

[2] G. Adiv. Inherent ambiguities in recovering 3-d motion and structure from a noisy flow field. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 70–77, 1985.

[3] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989.

[4] S. Atiya and G. D. Hager. Real-time vision-based robot localization. *IEEE Transaction on Robotics and Automation*, 9(6):785–800, December 1993.

[5] N. Ayache and O. D. Faugeras. Maintaining representation of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 6(6), December 1989.

[6] A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure and focal length. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 17(6):384–401, 1995.

[7] B. S. B. Roberts and B. Bhanu. Inertial navigation sensor integrated motion analysis for obstacle detection. In *Proc. of Digital Avionics Systems Conference*, pages 131–136, 1991.

[8] H. H. Baker and R. C. Boles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *International Journal of Computer Vision*, 3(1):33–49, 1989.

[9] S. T. Barnard and W. B. Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):527–534, 1980.

[10] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 150 – 156, 1992.

[11] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Higorani. Hierarchical model-based motion estimation. In *Proc. of European Conference on Computer Vision*, pages 237–252, 1992.

[12] M. J. Black. Combining intensity and motion for incremental segmentation and tracking over long image sequences. In *Proc. of European Conference on Computer Vision*, pages 485–493, 1992.

[13] S. D. Blostein and T. S. Huang. Error analysis in stereo determination of 3-d point positions. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 9(6):752–765, November 1987.

[14] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision*, 1(1):7–55, 1987.

[15] T. J. Broida and R. Chellappa. Estimation of object motion parameters from noisy images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(1):90–99, 1986.

[16] T. J. Broida and R. Chellappa. Estimating kinematics of a rigid object from a sequence of monocular images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11(5):497–513, 1991.

[17] M. . A. Burges and R. A. Hayes. Synthetic vision - a view in the fog. *IEEE AES Systems Magazine*, 8(3):6–13, March 1993.

[18] P. J. Burt. Fast filter transforms for image processing. *Computer Graphics and Image Processing*, 16(1):20–51, 1981.

[19] J. Canny. A computational approach to edge detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[20] S. Carlsson. Object detection using model-based prediction and motion parallax. In *Proc. of European Conference on Computer Vision*, pages 297–306, 1990.

[21] Z. Chen, D. C. Tseng, and J. Y. Lin. A simple vision algorithm for 3-d position determination using a single calibration. *Pattern Recognition*, 22(2):173–187, 1989.

[22] V. H. L. Cheng and B. Sridhar. Consideration for automated nap-of-earth rotor-craft flight. *Journal of American Helicopter Society*, 36(2):61–69, 1991.

[23] H. L. Chou and W. H. Tsai. A new approach to robot location by house corners. *Pattern Recognition*, 19(6):439–451, 1986.

[24] R. P. G. Collinsion. *Introduction to Avionics*, chapter Autopilots and Flight Management Systems. Microwave Technology Series 11. Chapman and Hall, 1996.

[25] J. L. Crowley, P. Stelmaszyk, and C. Discours. Measuring image flow by tracking edge-lines. In *Proc. of International Conference on Computer Vision*, pages 658–664, 1988.

[26] J. L. Crowley, P. Stelmaszyk, and C. Discours. Measurement and integration of 3-d structures by tracking edge lines. *International Journal of Computer Vision*, 8(1):29–52, 1992.

[27] N. Cui, J. Weng, and P. Cohen. Extended structure and motion analysis from monocular image sequences. In *Proc. of International Conference on Computer Vision*, pages 222–229, 1990.

[28] N. Cui, J. Weng, and P. Cohen. Recursive batch estimation of motion and structure from monocular image sequences. *Computer Vision, Graphics and Image Processing*, 59(2):154–170, 1994.

[29] S. Devadiga and R. Kasturi. Real-time implementation of pmmw image sequence processing: A feasibility study. An interim report for NASA Grant NAG-1-1371, Analysis of Image Sequences from Sensors for Restricted Visibility Operations, 9 1993.

[30] E. D. Dickmanns and F. R. Schell. Autonomous landing of airplanes by dynamic vision. In *Proc. of IEEE Workshop on Applications of Computer Vision*, pages 172–179, December 1992.

[31] K. C. Drake, E. S. McVey, and R. M. Inigo. Sensing error for a mobile robot using line navigation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 7(4):485–490, July 1985.

[32] K. C. Drake, E. S. McVey, and R. M. Inigo. Sensor roll angle error for a mobile robot using line navigation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 10(4):727–731, September 1988.

[33] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* A Wiley-Interscience Publication, New York, 1973.

[34] W. Enkelmann. Investigations of multi-grid algorithms for estimation of optical flow fields in image sequences. *Computer Vision, Graphics and Image Processing*, 43(2):150–177, 1988.

[35] A. Etemadi, J. P. Smith, G. Matas, J. Illingworth, and J. Kittler. Low-level grouping of straight line segments. Technical report, Department of Electronic and Electrical Engineering, University of Surrey, Guildford, U. K., 1993.

[36] J. Fang and T. S. Huang. Some experiments on estimating the 3-d motion parameters of a rigid body from two consecutive image frames. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 6(5):545–554, 1984.

[37] O. D. Faugeras, F. Lustman, and G. Toscani. Motion and structure from point and line matches. In *Proc. of International Conference on Computer Vision*, pages 25–34, 1987.

[38] O. D. Faugeras and S. Maybank. Motion from point matches: Multiplicity of solutions. *International Journal of Computer Vision*, 4(3):225–246, 1990.

[39] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of ACM*, 24(6):381–395, June 1981.

[40] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1):77–104, 1990.

[41] J. D. Foley, A. V. Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics - Principles and Practice.* Addison-Wesley Publishing Co., 2 edition, 1990.

[42] I. Fukuchi. Tv image processing to determine the position of a robot vehicle. *Pattern Recognition*, 14(1-6):101–109, 1981.

[43] S. Ganapathy. Decomposition of transformation matrices for robot navigation. In *Proc. of International Conference on Robotics and Automation*, pages 130–139, 1984.

[44] T. Gandhi, S. Devadiga, R. Kasturi, and O. Cmaps. Detection of obstacles on the runway by ego-motion compensation and tracking of significant features. In *Proc. of IEEE Workshop on Applications of Computer Vision*, pages 168–173, 1996.

[45] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 1992.

[46] R. M. Haralick. Determining camera parameters from the perspective projection of a rectangle. *Pattern Recognition*, 22(3):225–230, 1989.

[47] R. M. Haralick. Performance characterization protocol in computer vision. In *Proc. of DARPA Image Understanding Workshop*, pages 667–673, 1994.

[48] R. M. Haralick and Y. H. Chu. Solving camera parameters from the perspective projection of a parameterized curve. *Pattern Recognition*, 17(6):637–645, 1984.

[49] P. E. Hart. Stereo-graphic perception of 3-d scenes. Technical Report SRI Project 7642, Stanford Research Institute, Menlo Park, California, August 1969.

[50] R. I. Hartley. Projective reconstruction from line correspondences. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 903–907, 1994.

[51] A. S. Haynes and R. Jain. Detection of moving edges. *Computer Vision, Graphics and Image Processing*, 21(3):345–367, 1983.

[52] D. J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1(4):279–302, 1987.

[53] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(?):185–204, 1981.

[54] K. C. Huang, C. N. Shyi, J. Y. Lee, and T. C. Lee. Robot location determination in a complex environment by multiple marks. *Pattern Recognition*, 21(6):567–580, 1988.

[55] A. Huertas, W. Cole, and R. Nevetia. Detecting runways in complex airport scenes. *Computer Vision Graphics and Image Processing*, 51(2), 1990.

[56] D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge. Tracking non-rigid objects in complex scenes. In *Proc. of International Conference on Computer Vision*, pages 93–101, 1993.

[57] R. Jain, D. Militzer, and H.-H. Nagel. Separating non-stationary scene components in a sequence of real world tv images. In *Proc. of International Joint Conference on Artificial Intelligence*, pages 425–428, 1977.

[58] R. Jain and H.-H. Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 1(2):206–214, 1979.

[59] A. C. Kak. Depth perception for robots. Technical Report TR-EE-83-44, Purdue University, 1983.

[60] D. Koller, K. Danilidis, and H.-H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, 1993.

[61] D. J. Kriegman, E. Trendel, and T. O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Transaction on Robotics and Automation*, 5(6):792–803, December 1989.

[62] R. Kumar, A. Tirumalai, and R. C. Jain. A nonlinear optimization algorithm for the estimation of structure and motion. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 209–215, 1989.

[63] C. H. Lee and T. S. Huang. Finding point correspondence and determining motion of a rigid object from two weak perspective views. *Computer Vision Graphics and Image Processing*, 52(3):309–327, 1990.

[64] H. J. Lee and C. T. Deng. Camera models determination using multiple frames. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 127–132, June 1991.

[65] R. Lee, S. Y. Chen, P. C. Lu, and W. H. Tsai. Flying object location using a single view of a semi-circle running track. In *Proc. of Workshop on Computer Vision, Graphics and Image Processing*, pages 178–187, August 1989.

[66] H. J. Liu and C. T. Deng. Camera models determination using multiple frames. In *Proc. of IEEE Conference on Computer Vision and Image Processing*, pages 127–132, 1991.

[67] Y. Liu and T. S. Huang. Estimation of rigid body motion using straight line correspondence. *Computer Vision Graphics and Image Processing*, 43(1):37–52, 1988.

[68] Y. Liu and T. S. Huang. A linear algorithm for motion estimation using straight line correspondences. *Computer Vision Graphics and Image Processing*, 44(1):35–37, 1988.

[69] Y. Liu, T. S. Huang, and O. D. Faugeras. Determination of camera location from 2-d to 3-d line and point correspondence. In *Proc. of the IEEE Conference in Computer Vision and Pattern Recognition*, pages 82–88, June 1988.

[70] Y. Liu, T. S. Huang, and O. D. Faugeras. Determination of camera location from 2-d and 3-d line and point correspondences. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 12(1):28–37, January 1990.

[71] H. C. Longuet. A computer program for reconstructing scene from two projections. *Nature*, 293(?):133–135, 1981.

[72] H. C. Longuet and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London B*, 208(?):385–397, 1980.

[73] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[74] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of DARPA Image Understanding Workshop*, pages 121–130, 1981.

[75] M. J. Magee and J. K. Aggarwal. Determining the position of a robot using a single calibration object. In *Proc. of IEEE Conference on Robotics*, pages 140–149, 1984.

[76] E. S..Mcvey and J. W. Lee. Some accuracy and resolution aspects of computer vision distance measurements. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 4(6):646–649, November 1982.

[77] G. Medioni and R. Nevetia. Matching images using linear features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):675–685, 1984.

[78] A. Meygret and M. Thonnat. Segmentation of optical flow and 3-d data for the interpretation of mobile objects. In *Proc. of 3rd International Conference on Computer Vision*, pages 238–245, 1990.

[79] A. Mitiche, S. Seida, and J. K. Agarwal. Interpretation of structure and motion from line correspondences. In *Proc. of International Conference on Pattern Recognition*, pages 1110–1112, 1986.

[80] H. P. Moravec. Towards automatic visual obstacle avoidance. In *Proc. of 5th International Conference on Artificial Intelligence*, pages 584–589, 1977.

[81] D. C. Murdoch. *Analytical Geometry with an Introduction to Vectors and Matrices.* John Wiley and Sons, Inc., 1966.

[82] H. H. Nagel. Displacement vectors derived from second order intensity variations in image sequences. *Computer Graphics and Image Processing*, 21(1):85–117, 1983.

[83] H. H. Nagel. On the estimation of optical flow: Relations between different approaches and some new results. *Artificial Intelligence*, 33(?):299–324, 1987.

[84] H. H. Nagel and W. Enkelmann. An investigation of smoothness constraint for the estimation of displacement vector fields from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):565–593, 1986.

[85] T. C. Naugen and T. S. Huang. Quantization errors in axial stereo motion on rectangular tessellated image sensors. In *Proc. of International Conference on Pattern Recognition*, pages 13–16, 1992.

[86] S. Neghadaripour and B. K. P. Horn. Determining 3-d motion of planar objects from image brightness measurements. In *Proc. of International Joint Conference on Artificial Intelligence*, pages 18–23, 1985.

[87] R. C. Nelson. Qualitative detection of motion by a moving observer. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 173–178, 1992.

[88] J. Philip. Estimation of 3-d motion of rigid objects from noisy observations. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 13(1):61–66, 1991.

[89] J. W. Roach and J. K. Agarwal. Determining the movement of objects from a sequence of images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2(6):554–562, 1980.

[90] A. Rosenfeld, editor. *Multi-resolution Image Processing and Analysis*, chapter The Pyramid as a Structure for Efficient Computation. Springer-Verlag, 1984.

[91] H. Sarait and K. E. Price. Motion estimation with more than two frames. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 12(5):417–434, 1990.

[92] S. M. Sietz and C. R. Dyer. Complete scene structure from four point correspondence. In *Proc. of International Conference on Computer Vision*, pages 330–337, 1995.

[93] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distributions of optical flow. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 310–315, 1991.

[94] A. Singh. An estimation theoretic framework for image flow computation. In *Proc. of IEEE International Conference on Computer Vision*, pages 168–177, 1990.

[95] P. Smith, B. Sridhar, and B. Hussien. Vision-based range estimation using helicopter flight data. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 202–208, June 1992.

[96] P. N. Smith. Nasa image data base user's guide. Technical report, NASA Ames Research Center, Moffet Field, CA, 1990.

[97] P. N. Smith, B. Sridhar, and R. E. Suorsa. Multiple-camera/motion stereoscopy for range estimation in helicopter flight. In *Proc. of American Control Conference*, pages 1339–1343, June 1993.

[98] S. M. Smith. *Feature-based Image Sequence Understanding.* PhD thesis, Department of Engineering Science, Oxford University, 1992.

[99] S. M. Smith. A new class of corner detector. In *Proc. of British Machine Vision Conference*, pages 139–148, 1992.

[100] S. M. Smith. Asset-2: Real-time motion segmentation and shape tracking. In *Proc. of International Conference on Computer Vision*, pages 237–244, 1995.

[101] S. Snyder, B. Schipper, L. Vallot, N. Parker, and G. Spitzer. Differential gps/inertial navigation approach/landing flight test results. *IEEE PLANS*, March 1992.

[102] T. Soni and B. Sridhar. Modeling issues in vision-based aircraft navigation during landing. In *Proc. of IEEE Workshop on Applications of Computer Vision*, pages 89–96, December 1994.

[103] M. E. Spetsakis. A linear algorithm for point and line-based structure from motion. *Computer Vision Graphics and Image Processing*, 56(2):230–241, 1992.

[104] M. E. Spetsakis and J. Aloimonos. Closed form solution to the structure from motion problem from line correspondences. In *Proc. of National Conference on Artificial Intelligence*, pages 738–743, 1987.

[105] M. E. Spetsakis and J. Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4(3):171–184, 1990.

[106] M. E. Spetsakis and J. Aloimonos. A multi-frame approach to visual motion perception. *International Journal of Computer Vision*, 6(3):245–255, 1991.

[107] B. Sridhar and G. Chatterji. Computer aided system for detecting runway incursions. *IS&T/SPIE International Symposium on Electronic Imaging, Image and Video Processing*, 2220:328–337, 1994.

[108] B. Sridhar and A. V. Pathak. Analysis of image-based navigation system for rotorcraft low-altitude flight. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):290–299, March/April 1992.

[109] B. Sridhar and R. Suorsa. Comparison of motion and stereo methods in passive ranging. *IEEE Transactions on Aerospace and Electronic Systems*, 27(4):741–746, July 1991.

[110] B. Sridhar, R. Suorsa, and B. Hussien. Passive range estimation for rotor-craft low-altitude flight. *Machine Vision and Applications*, 6(1):10–24, 1993.

[111] B. Sridhar, R. Suorsa, P. Smith, and B. Hussien. Vision-based obstacle detection for rotor-craft flight. *Journal of Robotics Systems*, 9(6):709–727, 1992.

[112] M. Subbarao. Solution and uniqueness of image flow equations for rigid curved surfaces in motion. In *Proc. of International Conference on Computer Vision*, pages 687–692, 1987.

[113] K. Sugihara. Some location problems for robot navigation using a single camera. *Computer Vision Graphics and Image Processing*, 42(1):112–129, 1988.

[114] S. Sull and N. Ahuja. Integrated 3-d analysis and analysis guided synthesis of flight image sequences. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(4):357–372, 1994.

[115] S. Sull and B. Sridhar. Runway obstacle detection by controlled spatiotemporal image flow disparity. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 385–390, June 1996.

[116] J. R. Swink and R. T. Goins. High speed research system study: Advanced flight deck configuration effects. NASA Contractor Report 189650, National Aeronautics and Space Administration, Langley Research Center, Hampton, Virginia 23665-5255, 1991.

[117] Y. L. Tang, S. Devadiga, R. Kasturi, and R. H. Sr. Model-based approach for detection of objects in low resolution passive-millimeter wave images. In *Proc. IS&T/SPIE Symposium on Electronic Imaging Science and Technology, vol. 2182*, pages 320–330, 1994.

[118] Y. L. Tang and R. Kasturi. Accurate estimation of object location in an image sequence using helicopter flight data. *Journal of Robotics and Computer Integrated Manufacturing*, 11(2):65–72, 1994.

[119] Y. L. Tang and R. Kasturi. Detection of runway in image sequences. In *Proc. IS&T/SPIE Symposium on Electronic Imaging Science and Technology, vol. 2421*, pages 181–190, 1995.

[120] Y. L. Tang and R. Kasturi. Tracking moving objects during low altitude flight. *Journal of Machine Vision and Applications*, 9(1):20–31, 1996.

[121] W. B. Thompson and T. C. Pong. Detecting moving objects. *International Journal of Computer Vision*, 4(1):39–57, 1990.

[122] T. Y. Tian and M. Shah. Recovering 3-d motion of multiple objects using adaptive hough transform. In *Proc. of International Conference on Computer Vision*, pages 284–289, 1995.

[123] R. Y. Tsai and T. S. Huang. Uniqueness and estimation of 3-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 6(1):13–27, 1984.

[124] A. W. Waxman, B. K. Parsi, and M. Subbarao. Closed form solutions to image flow equations. In *Proc. of Conference on Artificial Intelligence*, pages 12–23, 1984.

[125] A. W. Waxman and K. Wohn. Image flow theory: A framework for 3-d inference from time-varying imagery. In C. Brown, editor, *Advances in Computer Vision*. Erlbaum Publishers, 1987.

[126] J. Weng, N. Ahuja, and T. S. Huang. Motion and structure from point correspondences: A robust algorithm for planar cases with error estimation. In *Proc. of International Conference on Pattern Recognition*, pages 247–251, 1988.

[127] J. Weng, N. Ahuja, and T. S. Huang. Optimal motion and structure estimation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 15(9):864–884, 1993.

[128] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from two perspective views: Algorithms, error analysis and error estimation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11(5):451–476, 1989.

[129] J. Weng, T. S. Huang, and N. Ahuja. Motion and structure from line correspondences: Closed form solution, uniqueness and optimization. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 14(3):318–336, 1992.

[130] P. R. Wolf. *Elements of Photogrammetry*. Mc. Grawhill, NY, 1974.

[131] Y. Yasumato and G. Medioni. Robust estimation of 3-d motion parameter from a sequence of image frames using regularization. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8(4):464–471, 1988.

[132] S. K. Young, R. A. Davidheiser, B. Hauss, M. Mussetto, M. M. Shoucri, and L. Yujiri. Passive millimeter wave imaging. *TRW Space and Defense - Quest*, 13(2):3–20, Winter 1990/1991.

[133] Z. Zhang. Estimating motion and structure from correspondence of line segments between two perspective images. In *Proc. of the International Conference on Computer Vision*, pages 257–262, 1995.