Visual Odometry Estimation Using Selective Features

Vishwas Venkatachalapathy
vv3156@rit.edu

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
Visual Odometry Estimation Using Selective Features

By

Vishwas Venkatachalapathy

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Dr. Raymond W Ptucha
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, NY
July, 2016

Approved By:

Dr. Raymond W Ptucha
Primary Advisor – R.I.T. Dept. of Computer Engineering

Dr. Andreas Savakis
Secondary Advisor – R.I.T. Dept. of Computer Engineering

Dr. Clark Hochgraf
Secondary Advisor – R.I.T. Dept. of Computer Engineering
To my beloved parents Mr. Venkatachalamapathy and Mrs. Geetha, and my precious sister Pooja.
I take this opportunity to express my profound gratitude and deep regards to my primary advisor Dr. Raymond W Ptucha for his exemplary guidance, monitoring and constant encouragement throughout this thesis. Dr. Ptucha dedicated his valuable time to review my work constantly and provide valuable suggestions which helped in overcoming many obstacles and keeping the work on the right track. I would like to express my deepest gratitude to Dr. Andreas Savakis and Dr. Clark Hochgraf for accepting to be the thesis review committee members. I am grateful for their valuable time and cooperation during the course of this thesis. I also take this opportunity to thank my research group members for all the constant support and help provided by them.
Abstract

The rapid growth in computational power and technology has enabled the automotive industry to do extensive research into autonomous vehicles. So called self-driven cars are seen everywhere, being developed from many companies like, Google, Mercedes Benz, Delphi, Tesla, Uber and many others. One of the challenging tasks for these vehicles is to track incremental motion in runtime and to analyze surroundings for accurate localization. This crucial information is used by many internal systems like active suspension control, autonomous steering, lane change assist and many such applications. All these systems rely on incremental motion to infer logical conclusions. Measurement of incremental change in pose or perspective, in other words, changes in motion, measured using visual only information is called Visual Odometry. This thesis proposes an approach to solve the Visual Odometry problem by using stereo-camera vision to incrementally estimate the pose of a vehicle by examining changes that motion induces on the background in the frame captured from stereo cameras.

The approach in this thesis research uses a selective feature based motion tracking method to track the motion of the vehicle by analyzing the motion of its static surroundings and discarding the motion induced by dynamic background (outliers). The proposed approach considers that the surrounding may have moving objects like a truck, a car or a pedestrian body which has its own motion which may be different with respect to the vehicle. Use of stereo camera adds depth information which provides more crucial information necessary for detecting and rejecting outliers. Refining the interest point location using sinusoidal interpolation further increases the accuracy of the motion estimation results. The results show that by using a process that chooses features only on the static background and by tracking these features accurately, robust semantic information can be obtained.
# Table of Contents

Acknowledgements ........................................................................................................ iii
Abstract ........................................................................................................................... iv
List of Figures .................................................................................................................. vi
List of Tables ................................................................................................................... vii

Chapter 1  Introduction ................................................................................................. 1
  1.1. Odometer and Odometry ..................................................................................... 1
  1.2. Visual Odometry ............................................................................................... 2
  1.3. Visually Aided Inertial Odometry .................................................................... 2
  1.4. Stereo and Monocular Visual Odometry .......................................................... 2

Chapter 2  : Motivation from Previous Work .............................................................. 5

Chapter 3  : Datasets .................................................................................................... 9

Chapter 4  : Methodology .......................................................................................... 11
  4.1. Proposed Algorithm ...................................................................................... 12
  4.2. Lens Distorsion: ............................................................................................. 14
  4.3. Rectification/ Calibration : ............................................................................. 16
  4.4. Feature Detection .......................................................................................... 19
  4.5. Feature Description and Matching ................................................................. 25
  4.6. Depth Computation ....................................................................................... 28
  4.7. Pose Estimation ............................................................................................. 35

Chapter 5  : Experiments ............................................................................................ 38

Chapter 6  : Conclusion .............................................................................................. 50

Bibliography .................................................................................................................. 51

Chapter 7  Appendix A ............................................................................................... 55
  7.1. Stereo Camera Setup....................................................................................... 55
  7.2. Accessing images from Cameras .................................................................... 58
  7.3. Calibration of the Cameras ............................................................................ 61
  7.4. Compile and Debug the code: ...................................................................... 63
List of Figures

Figure 3-1 Sequence path traced in KITTI dataset [47]............................................... 9
Figure 3-2 Setup used for data collection in KITTI dataset [47].................................. 9
Figure 3-3 Path traced by the robot in New college dataset [46]............................... 10
Figure 3-4 Robot used for new College Dataset [46]................................................. 10
Figure 4-1 Block diagram of the proposed approach.................................................... 11
Figure 4-2 Checkerboard pattern before and after removing lense distortion.............. 15
Figure 4-3 Stereo camera setup.................................................................................... 16
Figure 4-4 Stereo camera pose rectification............................................................... 17
Figure 4-5 Feature matching in the stereo pair............................................................ 17
Figure 4-6 Multiple orientations of the checkerboard to estimate camera calibration parameters........................................................................................................... 18
Figure 4-7 Image showing the interest point under test and the 16 pixels on the circle [27]. ...................................................................................................................... 19
Figure 4-8 Pixel p and its neighboring pixels in a vector form [5].................................... 21
Figure 4-9 Fast key points, green dots show the Non-maximally suppressed corners [5]......................................................................................................................... 22
Figure 4-10 Features concentrated around regions with high intensity variations ....... 23
Figure 4-12 Image bucketing or windowing............................................................... 23
Figure 4-13 Features generated from adaptive feature generation. ......................... 24
Figure 4-14 Graph showing no. of feature generted by using fixed FAST thresholding. 24
Figure 4-15 Graph showing no. of features generted by using adaptive FAST thresholding.......................................................................................................................... 24
Figure 4-16 Feature tracking....................................................................................... 25
Figure 4-17 Optical flow features being captures for t and t-1 time instances.............. 27
Figure 4-18 Stereo images overlaid from KITTI dataset, notice the feature matches are along parallel (horizontal) lines[50].............................................................. 28
Figure 4-19 A disparity map computed on frames from KITTI VO dataset [50].......... 29
Figure 4-20 Projection matrix for left and right stereo cameras............................... 29
Figure 4-21 Feature tracking through DoG [40] pyramid. .......................................... 30
Figure 4-22 Feature matching from left to right pyramid.......................................... 31
Figure 4-23 Sinusoidal Sub pixel interpolation. ......................................................... 32
Figure 4-24 Motion of a pixel w.r.t to its depth.......................................................... 32
Figure 4-25 Geometrical representaion of stereor camera setup................................. 33
Figure 4-26 Triangular congruency in the stereo camera setup................................. 33
Figure 4-27 Outlier feature detection using prediction error....................................... 37
7-1 Camera baseline distance...................................................................................... 56
Figure 7-2 Stereo camera setup on golfkart.............................................................. 56
Figure 7-3 Stereo Camera Configuration .................................................................... 57
7-4 Login snapshot of Hik-Vision Camera .............................................................. 58
Figure 7-5 Output Video config snapshot ................................................................ 59
Figure 7-6 Output Camer ID snapshot......................................................................... 59
Figure 7-7 Output Streaming protocol and its authentication snapshot..................... 59
Figure 7-8 Checker board pattern for camera calibration.......................................... 62
Figure 7-9 Checker board pattern for camera calibration.......................................... 62
List of Tables

Table 5-1 Subpixel regression Statistics. ................................................................. 39
Table 5-2 Execution time for each step. ................................................................. 40
Table 5-3 RMS Error for data based on date ......................................................... 45
Table 5-4 RMS Error for data based on content. .................................................... 45
Table 5-5 Translational and rotational result for all the sequences of KITTI dataset. 46
Table 5-6 New college dataset results for translation and rotation......................... 49
Table 5-7 Result comparison with state of the art approaches. .............................. 49
Chapter 1  Introduction

One of the significant challenges for both autonomous cars and robots is to find the current position and heading, either globally or locally. To understand globally, is to know the exact position in the real world (e.g. global positioning system), and to understand locally is with reference to a particular starting point. This knowledge is very essential when the return path has to be traced or when the path changes and then rerouting has to be done for these robots or moving objects. Hardware sensors can gather acceleration and rotation information, but lack the potential to detect any other information, such as, wheel slip and drift over time. Visual odometry can provide that crucially needed extra information, that we humans make use of everyday. Visual Odometry is a concept that came to life inspired by human’s ability to analyze motion using visual data. Visual information is so rich of information, and if analyzed could provide a lot more than what’s necessary. Humans analyze visual information using our incredible brain that has evolved over millions of years, and just now computers are starting to possess some of these capabilities. This thesis research focuses on problems and solutions in analyzing visual data to capture self-motion of an object. Visual data can provide information regarding the surroundings, obstacles and also reconstruction of the scene to make informed decisions. Different camera setups can help visualize the world in either 2D or 3D perspective.

1.1. Odometer and Odometry

Odometer is a device used to calculate the distance travelled based on the rotations that the wheel undergoes along with the wheel base, and the wheel radius measurements. Odometry is a common term used to measure motion vectors and pose variation in robotics. The pose measurement is continues and has to be done at discrete time intervals. Measurement of velocity and rotation along x, y and z axis is common in robots and cars using inertial measuring unit (IMU). IMU uses inertial changes and changes in center of gravity to estimate these parameters. Wheel encoders are also used to measure speed. These hardware sensors can only perform what they were designed to do and cannot be upgraded to process or to collect any other information.
1.2. **Visual Odometry**

Motion Estimation / Pose estimation at discrete time intervals using visual data like images or depth data from sensors like cameras and Lidars is termed as Visual odometry. Visual data is captured from a sensor rigidly attached to the body of robot, for which the motion estimation is of interest. This visual data is used to generate real world motion trajectory using the visual data stream. The visual data may also be used for inferring other information like objects in the scene, localization and many more applications. Use of different sensors provides different information to be processed. Stereo cameras, like the human eyes, are two identical cameras fitted into a solid structure to provide images along with stereoscopic depth. A single monocular camera provides image data that would lack a degree of freedom when compared to the stereo cameras, but can be very efficient when compared with a ranging sensor.

1.3. **Visually Aided Inertial Odometry**

The idea of combining both the visual and the inertial information to get good results was proposed during the early research for the space exploration rovers. This idea uses visual and inertial data to infer the change in pose of the object. This approach uses either loose coupling or tight coupling of the data. Loose coupling is when both the visual and the inertial data are processed independently and the results are refined or coupled together. In case of tight coupling both the visual and inertial information are used together to predict the result.

1.4. **Stereo and Monocular Visual Odometry**

Stereo and monocular camera systems are used widely today for various applications. Both provide a continuous visual image feed, which can later be used for any specific use. Stereo camera is usually a two or more camera system rigidly fixed to a platform in a known geometry. Visual odometry estimation using such sensors is called stereo visual odometry. Monocular cameras are single camera setups and can be used in monocular visual odometry. Stereo cameras have the advantage of possessing disparity and hence the depth map form camera parameters, which adds to the information available. Monocular systems can only measure motion in terms of pixel motion; rather stereo visual odometry can measure motion in real world coordinates in meters. Some approaches today has replicated the stereo system by
using a ranging sensor along with monocular cameras. Farther the objects in the scene more erroneous it is to compute depth, and if majority of the objects in the scene are farther away in the scene, when compared to the baseline distance between the cameras, its beneficial to use a monocular visual odometry algorithm like Semi direct monocular Visual Odometry (SVO) [2].

For this thesis research, stereo visual odometry estimation is investigated. Adaptive feature detectors and selective features for motion estimation are used, such as Horn’s quaternion equation [1]. The use of adaptive feature detectors enhances the feature count and hence the information content gathered from the image. The selective feature extractor helps in avoiding features on moving objects, hence avoiding dynamic background and only considering static background for motion estimation. The use of Horn’s quaternion equation [1], aided by a perspective transform for motion estimation, helps to find motion estimation quicker and more reliably. The motion estimation process often produces speckle errors and hence smoothening of results generally improves results. The use of multiple previous frames for motion refinement helps in selecting robust and reliable features on the static background and using them for accurate motion estimation. Current state of the art algorithms improve results by post processing, like loop closure detection for trajectory correction and localization for position refinement. Without such post processing, there usually is a huge error that gets accumulated over time. The approach described in this thesis tries to reduce the accumulated run time error. When used with loop closure detection or other post processing, this can yield much more accurate results.

Novel contributions in this thesis research include:

- Use of adaptive feature generation, to generate dynamically distributed sparse features throughout the image.
- Use of windowing and adaptive Features from Accelerated Segment Test (FAST) thresholding to acquire constant number of robust features for efficient tracking through multiple frames.
- Use of sub-pixel interpolation while finding feature correspondence and feature tracking for precise location information.
• Use of Sum of Absolute Difference (SAD) /Normalized Cross Correlation (NCC) with sub pixel interpolation for efficient feature matching.

• Feature profiling with weights based on their result contribution and there tracking history for efficient pose estimation results.
Chapter 2 : Motivation from Previous Work

Visual odometry, finds its roots from a problem commonly known as structure from motion (SFM). SFM is a problem of recovering relative camera pose of the body and its 3D structure from a set of camera’s, which could be either calibrated or non-calibrated (epipolar plane). It was initially solved in [3], [4] and [5]. The concept of visual odometry was coined in 2004 in [3] and used dense stereo matching along with optical flow to estimate motion. In [4] and [5] concepts related to 3D projections, camera calibration, and baseline optimization were introduced. C Harris and J Pike [4] put forth the idea of position integration from consecutive frames to find out the end position with respect to the origin. SFM covers wider application like 3D reconstruction, but still needs visual odometry to track the position at which different image sets are taken. These image sets may be consecutive or in-ordered, and hence is usually processed offline. Such applications are time consuming and its time complexity increases with increase in number of image sets. The resultant structure and the pose of the cameras with which the images were captured are processed using offline optimizations like bundle adjustment [6]. Post processing algorithms like Bundle adjustment can be used to refine the local estimate of the trajectory.

While bundle adjustment [6] works on image sets that are captured non-consecutively, visual odometry processes image sets taken sequentially to track incremental changes that help in building a resultant motion map. Visual odometry is estimated in real-time, processes sets of image frames independently.

In early 1980’s, Moravec [7] started to solve the problem of a vehicle’s egomotion from visual input alone. Much of the early research following Moravec [45] was aimed at precise visual odometry for planetary rovers and it gained much more interest by NASA’s Mar’s exploration program. It was during this period where a lot of advantages and drawbacks of using visual only method for tracking vehicle’s egomotion was discovered and these outcomes inspired this thesis’ research into visual odometry. Providing 6-degree-of-freedom (DoF) for rover’s motion and overcoming wheel slippage in rough terrains were some important problems. Moravec’s [45] work laid the foundation of egomotion estimation by presenting the first motion-estimation approach.

Moravec’s work [45] was tested on a planetary rover who had a single camera sliding on a rail, which was called a slider stereo. The robot would move and stop for the camera to take pictures at nine equidistant points on the slider, thus depicting a
stereo camera approach. Since the camera was mounted on a slider which was level and the camera’s pose was fixed, the camera had epiploic geometry. The cameras baseline distance was the length of the slider bar and this information made calculations easier. The main assumption is that neither the robot, nor the surrounding moves during the image capturing stage. Once the images were captured, corners in one image were detected using Morvec’s corner detector [9] and these corners are matched to the right image using NCC (Normalized Cross Correlation). These corners are tracked to the next consecutive frame capturing the incremental motion of the robot using optical flow. Variance in the overall flow and discrepancies in the neighboring pixel depth information of the features can be outlined for outlier rejection. With the set of 3D points tracked between subsequent frames, rigid body transformation is used to align triangulated 3D points. Weighted least square of the triangulation vector of features based on their weights was used to reduce mean error in solving the equation obtained from two sets of 3D points. Once the camera captures the nine images and analyze these images for motion estimation, the robot would move. The motion in between the image capturing stage was very minimal and hence the speed at which the robot could travel was restricted. This was a major drawback. Moravec visualized the stereo camera by setting up a camera free to slide on an axis perpendicular to the scene being captured. As the sliding is done at known distances and the images captures are from single camera, they depict stereo image pair. This approach proved to be more accurate in terms of depth computation, as the stereo computation could be done over multiple images captured at discrete known distances.

Another single camera approach used to estimate the egomotion was triangulating the points in 3D space with the help of optical flow in frames between time instances- thus the name Monocular visual Odometry (MO). MO lacks the scale factor in egomotion estimation. This drawback can be countered with direct measurement of scale with the help of IMU’s or range sensors. The stereo camera setup is only effective for objects and scenes at a certain depth and farther the depth farther the error in predicting the depth using stereo image pair. The approach to compute depth relies on the congruency of the triangle formed between the baseline distance of the cameras and the depth of the scene or the object. At farther distances the base line distance tends towards zero and is not favored. Hence at this instance, monocular visual odometry approaches are much beneficial.
Shafer [10], [11] improvised Moravec’s algorithm by utilizing the features error covariance matrix for motion estimation. This extra information demonstrated superior results in pose estimation and motion correction for rovers used in space exploration. Olson et al. [12], [48] approached the problem with a separate hardware sensor to measure the orientation of the camera sensor and used Forester corner detector for feature detection as they are much faster over Moravec’s operator. They described issues with egomotion estimation and the problem of error accumulation over time. This error from each estimation process, however small it may be, over time gets accumulated and would completely corrupt the position information.

Lacroix et al. [14] described the importance of the key points in his implementation of stereo visual odometry for planetary exploration rovers. They used a dense stereo matching approach to cluster regions with similar depth and to track the motion of this region. The idea behind this approach was that the background can be classified into regions like buildings and trees and then tracking these regions would result in better accuracies. Features were clustered by their depth with the neighboring pixels as in [15], [34] as the shape of the correlation curve and the standard deviation of features depth are directly proportional. Cheng et al. [17], [18] implemented visual odometry onboard the Mars rovers, utilizing the same approach. The approach worked better as more information of the feature pertaining to its correlation function was utilized and the use of RANdom SAmples Consensus (RANSAC) [6] for outlier rejection. Milella and Siegwart [13] proposed a different approach using the Shi-Tomasi approach [19] for corner detection. This approach weighted features based on a score which depicted the robustness and reliability of the feature in predicting motion estimation. Using least squares, motion estimation was solved and then the Iterative Closest Point (ICP) algorithm [20] was used for pose refinement.

Visual Odometry was termed by Nister et al. [3]. He proposed real time implementation of motion estimation with robust outlier rejection algorithm. In this approach features were not tracked over consecutive frames rather they are detected for every stereo pair. Their approach estimated the camera pose as a 3-D-to-two-dimensional (2-D) problem and rejected outliers using RANSAC.

Kerl et al. [23] developed a dense visual odometry approach with an assumption that the cameras will have no intensity variations between frames. The approach uses segmented regions from an image, to estimate visual odometry, by tracking the regions rather than tracking individual features. This approach helps in
reducing computation time and speeds the estimation process. One key assumption that is considered in this approach is that the regions segmented in the image have a uniform motion, which may not always be true. Also this approach fails to work for scenes with a lot of regions like densely crowded city streets.

Huang et al. [24] developed Fast Visual odometry from Vision which is very similar to the approach proposed in this thesis but the process of estimation motion uses the sum of squared pixel error between frames. Frames in real-time are prone to exposure, white balance and many other illumination changes. This approach assumes that the image from two consecutive time instances will have the same intensity values shifted by a pose constant. The approach tracks pixels to estimate visual odometry. Since this approach assumes the pixel intensity to be its feature descriptor, feature matching will be inefficient as the intensity values change over time with varying pose.

Pomerleau and Magnenat [25] published another approach named point matcher. Though the process is modular and efficient for real-time videos, the approach lacks reliability as many of the error minimizers and parameters are hard coded. This approach is similar to approaches described above, in terms of feature registration and tracking. The visual odometry estimation process involves a lot of hard coded functions for selecting inliers and outliers. These hardcoded regions from where the features are selected are kept constant throughout the process and works well for select databases. Such restrictions cannot be applied to real-time visual odometry estimation process as the environmental conditions vary and the approach must be adaptive to the environment. For real-time visual odometry, methods should be independent, reliable and robust.

In all these approaches, the key assumption is that the background is static and all the features move with respect to the camera (no independent motion), which is typically not true in automotive applications. In automotive applications, cameras look into the road where every object has its own motion. During such instances, the outlier rejection process has to be strong along with feature detection. A finite balance has to be established in real-time between the number of inliers and the number of outliers.
Chapter 3  : Datasets
The process of estimating egomotion in this thesis uses stereo images captured from a stereo camera setup. The setup has to meet the stereo camera setup requirements. The datasets used for this research are the KITTI dataset and the New Collage dataset.

![Sequence path traced in KITTI dataset](image1.png)

Figure 3-1 Sequence path traced in KITTI dataset [47].

The KITTI dataset was formed by students from Karlsruhe Institute of technology in collaboration with Toyota Technological Institute, Chicago. The dataset was acquired with in the streets of Karlsruhe, in a modified car as shown in the image below. The dataset consists of stereo along with Velodyne laser data of up to 165GB. The dataset also consists of precise geographical locations of every image being captured. The modified car is equipped with two stereo cameras each for color and gray scale images with matched intrinsic and extrinsic parameters in a lossless PNG format.

![Setup used for data collection in KITTI dataset](image2.png)

Figure 3-2 Setup used for data collection in KITTI dataset [47].

The dataset consists of around 22 paths equipped with color and gray scale stereo image sets and 3D point cloud data for every image set. In the dataset 11 paths (00-10) have ground truth data and can be used for training and validating the
algorithm. 11 paths (11-21) do not have ground truth and are used for testing.

The New College Vision and Laser Dataset from Oxford contain 30Gb of data that is aimed at researchers working on outdoor 6 D.O.F navigation and mapping. The ground truth data is constructed using information from Global Positioning System (GPS) and Inertial Measuring Unit (IMU). The robot used for capturing the stereo and laser data along with the path traversed is shown in Figure 3.3.

Figure 3-3 Path traced by the robot in New college dataset [46].

Figure 3-4 Robot used for new College Dataset [46].
Chapter 4 : Methodology

We assume the stereo camera rig consists of two identical cameras, and that the images from these cameras are calibrated to an epipolar plane. The input is a sequence of gray scale frames, taken over fixed intervals of time. Left and right frames, captured at time $t$ and $t+1$ is referred as $L_t, L_{(t+1)}, R_t$ and $R_{(t+1)}$. These frames are the input to the algorithm and the motion trajectory between the $t$ and $t+1$ frame is expected as the output. Each and every feature is weighted for its contribution of information to infer this result, so that when the same feature is tracked to future frames, its correctness can be validated by their previous predictions.

Figure 4-1 Block diagram of the proposed approach.
4.1. Proposed Algorithm

The stereo image sets are rectified to satisfy epipolar geometry and the images are converted to gray scale for faster processing. Since the feature detection is only intensity level based, gray scale images provide sufficient information.

1. If the stereo image set is the first in its sequence, then the image is only used to generate a 3D feature set as shown in figure 4.1. Initial Feature generation stage is also performed if the tracking information is lost. In this stage,
   a. The image is first divided into segments by windowing the image.
   b. Each window will have an initial Fast Threshold value, which will be adaptively updated based on the number of features generated in that window. Using the Adaptive Fast Threshold value, generate fast features in each window separately as described in section 4.4.
   c. Match these features from left image to the right image in the Image stereo set to get feature correspondence and to generate the feature depth using (4.1) and (4.4) also described in section 4.6. Their location is made precise by using sub pixel interpolation. With the features location and depth, it becomes a three-dimensional feature.

\[
\text{disparity} = X_{\text{left}} - X_{\text{right}} \tag{4.1}
\]

\[
X_{\text{realworld}} = \frac{x-c_x}{\text{disparity}} * T \tag{4.2}
\]

\[
Y_{\text{realworld}} = \frac{y-c_y}{\text{disparity}} * T \tag{4.3}
\]

\[
Z = \frac{f}{p} * \frac{T}{\text{disparity}} \tag{4.4}
\]

Where: \( \text{disparity} = \text{depth of the features in Z direction.} \)

\( C_x = X \text{ axis variation of the image plane} \)

\( C_y = Y \text{ axis variation of the image plane} \)

\( f = \text{focal length of both the cameras} \)

\( T = \text{distance between teh cameras} \)

\( p = \text{distance between pixels inside the camera sensors} \)
2. If the stereo image set is not the first in its sequence then,
   a. The three dimensional features from the previous image set are
      tracked to the current stereo sets, left image using KLT optical
      flow described in section 4.5.
   b. Follow step 1c to find feature correspondence between the left
      and the right images of the stereo image sets. Only features that
      are tracked from the previous results to the current frame are
      considered.

3. By now we should have two sets of three dimensional features
   corresponding to two consecutive frames. Now the problem is much
   more simplified in way to find the orientation and translational
   changes between the three-dimensional feature set. At first we divide
   the three dimensional features into subsets and perform RANSAC
   using Horn’s Method to find out the weighted closed form solutions
   for absolute orientation. The band of results is considered to find the
   median pose.

4. After the Motion estimation step using Horn’s method, the features are
   weighted based on their contribution towards the final result.

5. The result is further corrected by using pose results of previous frames.

6. The features variance in motion from \( t-2 \) to \( t-1 \) and \( t-1 \) to \( t \) frames is
   recorded and used to predict if a feature is a good feature or not. The
   feature’s predicted motion is a continuation of its motion from the
   previous frames. The variance from its predicted motion to the actual
   motion being tracked in the current frame is used to weigh features. A
   weight of 0, is assigned to features that have huge variances in motion
   and such features are removed later.
4.2. **Lens Distortion:**

Cameras capture visual information where the amount of visual information that can be captured is limited by the aperture size of the camera. Increasing the aperture size overexposes the scene and hence is not optimal to capture more information. Wide angle lenses in conjunction with large aperture sizes are widely used these days. With the help of these lenses, the same camera with exactly the same aperture size can capture more information by wrapping the visual information into a sphere.

Though these wide angle lenses help in capturing more visual information, the transformation the visual information goes through is a nonlinear transformation. This nonlinear transformation, provides more visual information, but increases the complexity for visual odometry estimation as it destroys the epipolar geometry of the cameras. Undistorting the image will provide more visual information and also bring the images back to epipolar geometry and hence is a balanced solution generally followed today. A modified version of Brown’s model for undistortion is used. This model uses barrel distortion approach to undistort the images using the distortion center of the image sets \( X_c \). The distortion function is formulated in (4.5).

\[
X_U = X_D + L(r) \cdot (X_D - X_C)
\]

\[
L(r) = K_1 r^2 + K_2 r^4 \ldots
\]

\[
r = \sqrt{(x_D-x_C)^2 + (y_D-y_C)^2}
\]

where :
- \( X_D(x_D, y_D) \) = Distorted image points
- \( X_C(x_c, y_c) \) = Image’s distortion center
- \( X_U(x_U, y_U) \) = Undistorted image points

As formulated above in equation, the image is unwrapped into a new planar space, and the process to perform this operation is described in the procedure below.
1. Offline stage:
   a. Modelling and estimation of distortion parameters using equation 4.5 and checker board image sets (make use of the stereo properties to correct the checker board pattern [Figure 4.2] to have straight lines).
   b. As the distortion function is nonlinear, the unwrapping of pixels into the new location in the image has to be done manually (calculating the new pixel location). Hence a look-up table for the new pixel location is calculated to reduce the computation load and execution time in real-time.

2. Online stage:
   a. Use the look-up table created offline, to undistort each frame as it arrives.
4.3. **Rectification/ Calibration**:

The stereo camera consists of two cameras mounted rigidly with a known baseline distance between them. These cameras may not be perfectly aligned to each other. This alignment is very necessary as the epipolar geometry that results from the camera alignment simplifies stereo matching and other complex processes.

Aligning cameras perfectly using fixtures or hardware mounts is more complex than performing software optimizations. One commonly used alternative is to correct one frame from either of the cameras so that images would depict perfect alignment of the cameras. The process of calculating the alignment parameters is called calibration and then correcting the images based on the calibration parameters is called rectification.

Epipolar geometry simplifies the problem of depth as the search for feature matching is reduced drastically. In Figure 4.4, the solid rectangular box shows an image plane that’s aligned to each other, and the dotted rectangle shows an image plane that is not planer. Finding a feature correspondence from the left to the right image in a stereo image set becomes a two-dimensional search if both the images are not planar and is computationally expensive. Transforming the images from plane H1 to H is done by an affine transform H2. This fix, reduces the search of feature correspondence to a one-dimensional search.
After the transformation is made, the feature correspondence from the left to the right image is usually only in the x axis with an error of ± 1 pixels. A feature match after the rectification process is shown in Figure 4.5. The calibration process to identify the $H_2$ projection matrix to bring both the images into the same plane is a complex problem, and uses the pattern in the checker board image. For calibration, the checker board pattern is matched between the stereo image sets, to find out the projection matrix by minimizing the distortion caused due to the separation of cameras in the stereo camera rig.

Let $H$ be a projection transform that transforms the image to lie on the epipole $e$. Then the transformation $H_2$ for the second camera might be chosen so as to minimize the sum of squared distances $\sum_i d(H_2x_i, Hx_i')^2$. 

Figure 4-4 Stereo camera pose rectification.

Figure 4-5 Feature matching in the stereo pair
The procedure to find the transformation matrix $H_2$ to transform the images to an epipole is summarized below.

1. Find the correspondence features from left to the right images in the checker board pattern. This step searches for correspondence in two dimensions.
2. Compute the transformation matrix for these correspondence feature sets.
3. Computer the transformation matrix $H_2$ that maps the feature to the epipole $e$.
4. Iterate through all the features to get matrix $H_2$, to minimize the distortion due to the transformation.
5. Transform the first image to the H plane using matrix $H_2$ and the second image to H plane using matrix $H_2$.

This procedure is performed more than once by changing the orientation of the checkerboard to so that an optimal transformation matrix is obtained as shown in Figure 4.6.

![Figure 4-6 Multiple orientations of the checkerboard to estimate camera calibration parameters.](image)
4.4. Feature Detection

Rosten and Drummond [27] proposed an intensity based interest point detection algorithm for images called Features from Accelerated Segment Test (FAST). Features in an image are interest points which characterizes an image and can be uniquely identified. Features are rich in local information and hence this information should be traceable in consecutive frames. Hence features are widely used in applications like image matching, object recognition, tracking etc. As discussed earlier, feature detection was first conceptualized by the early computer vision research on Moravec corners. Harris corner [49] and SUSAN corner detectors are few amongst the early interest point detection algorithms. Though these algorithms were very successful in detecting key interest points, they are time consuming and were not optimal solutions for real-time applications. Thus FAST was introduced which was robust, intensity based, and less time consuming.

Figure 4-7 Image showing the interest point under test and the 16 pixels on the circle [27].
The FAST algorithm is explained below:

1. Consider a pixel $p$, at any given location $x, y$ in the image plane with intensity $IP$. Pixel $p$ is to be identified as an interest point or not. (Refer to Figure 4.7)
2. Assume a threshold intensity value (generally 20% of $IP$) to be $T$.
3. Consider all the neighboring 16 pixels that lie in a circular fashion around pixel $p$ (Bresenham circle [4] of radius 3).
4. For pixel $p$ to be considered as an interest point, $N$ neighboring pixels need to be either above $IP+T$ or below $IP-T$ (in [27] $N=12$) of the 16.
5. To speed up the process, first priority is given to pixels $I1, I5, I9$ and $I13$ of the circle. They are compared with $IP$. If at least three out of four pixels satisfy the condition, only then the procedure is continued for the other 16 pixels. Else the pixel $p$ is rejected as a possible interest point.
6. Repeat the procedure for all the pixels in the image.

This algorithm will not work well if $N<12$, as in this case the number of possible interest points will increase drastically. Also the speed of the algorithm is determined by the orientation in which the 16 pixels are queried.

To make the algorithm faster, a machine learning approach was proposed in [3] [5].

$$S_{p \to x} = \begin{cases} 
    d, & I_{p \to x} \leq I_p - T \text{ (darker)} \\
    s, & I_p - T < I_{p \to x} < I_p + T \text{ (similar)} \\
    b, & I_p + T \leq I_{p \to x} \text{ (brighter)} 
\end{cases}$$

(4.8)

where:

$S_{p \to x}$ = is the state

$I_{p \to x}$ = is the intensity of the pixel $x$

$T$ = is a threshold
The machine learning approach speeds up the process by training on asset of images. The process involves considering a set of pixels’ $p$ and a vector of neighboring intensities $P$ for every pixel $p$. Each neighboring pixel in the vector $P$ can take one of the three states, i.e. brighter, darker or same intensity as $IP$. By using this information as training data and the ground truth being the decision whether the pixel was a key point or not, a decision tree classifier (ID3 algorithm) is trained.

Another major drawback in early corner detection algorithms was that corners were detected close to each other and were coagulated near high intensity variations. This was later resolved by using a Non Maximal Suppression for removing adjacent corners [4]. This approach scored each corner with a score function $V$ for each detected corner. The score function was the sum of absolute differences between the intensity $IP$ and the intensities of the neighboring pixels in the arc. Corners adjacent to each other were scanned and the ones with lower $V$ scores were discarded.

$$V = \max \begin{cases} \sum (\text{pixel values} - p) & \text{if } \text{value} - p > T \\ \sum (p - \text{pixel values}) & \text{if } p - \text{value} > T \end{cases}$$

(4.9)

where $p$ = pixel of interest (center pixel)

$T$ = threshold used for detection

$\text{pixel values}$ = intensity values of neighboring pixels
Fast feature detection with non-maximal suppression is robust and provided reliable features at lesser computation time as compared to Scale Invariant Feature Transform (SIFT) [28] or Histogram of Oriented Gradients [29]. Feature detectors operating on the entire image generates coughed features in the highlights than in shadows. This is due to the dynamic variance of the brightness in the image. For Visual Odometry, Interest points should be spread throughout the image to track the egomotion with respect to every corner in the image. This issue is clearly shown in image figure 4.11.

As it is clearly evident that on applying fast feature detection algorithm even with non-maximal suppression, the interest points are not spread across the frames and are highly concentrated towards regions with very high intensity variations. In figure 4.11, the features are concentrated on the tree line where there is a high contrast in intensity values. To overcome this problem, this thesis introduces an adaptive featuring technique which is described below.

Figure 4-9 Fast features, green dots show the Non-maximally suppressed corners [5].
1. Divide the image into windows of equal sizes as shown in figure 4.12.
2. Treat each window in an image as separate image.
3. Using FAST feature with non-maximal suppression generate interest points.
4. If the number of features in a window decreases below a threshold $t$, increase the Fast feature detector threshold $T$, else if the number of features in window is more than the threshold $t$, reduce the Fast feature detector threshold $T$.
5. Always check for the Fast detector threshold $T$ to be within the limit ($T_{max}$ and $T_{min}$).

The resultant is a feature set that’s spread across the image and not cluttered. The other advantages of this approach are the visual odometry algorithm will have a
constant stream of robust features and feature count is nearly constant.

Figure 4-12  Features generated from adaptive feature generation.

Figure 4-13 Graph showing no. of features generated by using fixed FAST thresholding.

Figure 4-14 Graph showing no. of features generated by using adaptive FAST thresholding.
4.5. **Feature Description and Matching**

Feature tracking provides key information about the motion of features between time intervals. This information after rejecting outliers is what is used to estimate visual odometry. The traceability of a feature over multiple frames provides an overall picture of how the background is moving with respect to the object as shown in Figure 4.16.

These features have $x,y$ and $z$ information in them and hence are 3D points.

For the approach proposed in this thesis, Kanade–Lucas–Tomasi feature tracker (KLT tracker) [30] was used to track features from frame $t-1$ to $t$. Feature tracking is also referred to as optical flow, and makes some assumptions which are summarized below.

- The intensity invariance. “Image intensities in small regions will remain the same although their location may change”. This can be expressed in (4.10), where $I$ is the pixel intensity at $x,y,z$ position.

\[
I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \quad (4.10)
\]
• The spatial coherence. “Neighboring pixels belongs to same surface and hence have similar motions”.
• The temporal persistence: “Image motion of a surface patch change gradually over time”.

Smaller movements can be liberalized using the Taylor series as shown in (4.11).

\[
I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t + \text{H.O.T.} \quad (4.11)
\]

Where:
- \(I(x + \delta x, y + \delta y, t + \delta t)\) = Intensities of the new image.
- \(I(x, y, t)\) = Intensities of the previous images.

The higher order terms in (4.11) can be neglected when we consider the intensity invariance and the temporal persistence assumption and see that the pixel intensities remain the same with variation in the pixel location (The differential location term). With all the assumptions, (4.11) reduces to (4.12).

\[
\frac{\partial I}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial I}{\partial y} \frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0 \quad \text{with optical flow } d = \begin{bmatrix} \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix} \quad (4.12)
\]

\[
E(\delta x, \delta y) = \sum_{x=\pm w_x}^{u_x} \sum_{y=\pm w_y}^{u_y} (I(x, y) - I(x + \delta x, y + \delta y, t + \delta t))^2 \quad (4.13)
\]

The KLT tracker uses (4.13) to estimate the optical flow vector. Optical flow only tracks features. So, at some point in time, features are lost, because they move out of the window. To keep a constant number of features in the image, the feature list needs to be updated from time to time. Also features are lost due to intensity variations when the camera gets over exposed or underexposed as the intensity of the features change. In the approach proposed in this thesis, only features with good FAST scores are tracked. Further, once features with good FAST scores are lost, we search for the same feature in the next five frames before it is dropped. The idea is that like the human eye, the camera’s auto exposure needs some time adapt to the changes in the light coming in.

Additionally, by dividing the image into multiple windows, and by using adaptive FAST thresholding for every window to detect sparse features, the feature count is kept constant. Minimum distance rule is enforced to avoid features being tracked which are near to each other.
After all the optimization, the features that are tracked are shown below in Figure 4.18. The green dots indicate where the features were located in the previous frame and the end of the arrow is where the features are in the current frame.

Figure 4-16 Optical flow features being captures for $t$ and $t-1$ time instances.
4.6. **Depth Computation**

Given a pair of images from a stereo camera, the disparity map is the measure of motion at pixel level with respect to change in perspective of the camera. The motion of the pixels is relative to its depth in world coordinates, and hence features near the camera have large movement, while features far from the camera have little or no motion. This characteristic property of interest points is exploited to measure the depth in real world co-ordinate system. Depth information provides that extra dimensionality to the data for visual odometry and is the key distinguishing factor between monocular and stereo visual odometry. In cases where ratio of baseline distance to the distance from the scene is low, the same extra dimensionality will not add any information, as the motion of these pixels is too small to be registered.

Figure 4.19 shows a conceptual view of the right and the left images of a stereo camera overlapped and its calculated disparity. This is a dense disparity map as the depth measurement is done to the whole frame. Dense disparity- depth measurements are suited for applications that rely on blocks of neighboring pixels for gathering information, like detecting moving objects in a scene in 3D. Dense disparity – depth measurements have been used for visual odometry estimation, by considering motion of blocks in the disparity. These approaches would produce good results in static scenes where the background has no other moving objects. For our approach we relied on sparse disparity – depth measurement for motion estimation, as the motion estimation information can be gathered in bits and pieces from selective features throughout the image. Sparse disparity-depth calculation is also less time consuming as the process of disparity and depth measurement is only done on selective features pixel location and not on the complete frame.

Figure 4-17 Stereo images overlaid from KITTI dataset, notice the feature matches are along parallel (horizontal) lines[50].
This concept can be used to compute the depth of the frame being captured. A feature in the real world coordinate system bearing \( X_1 \) coordinates along \( X \) axis in the left image and the same feature if found in the right image at \( X_2 \) location, the \( Z \) distance can be computed using (4.14). The \( Y \) values remains constant due to epipolar geometry.

Disparity is only computed for sparse features considered for motion estimation, as the approach is execution time critical. The spatial disparity map only computes the disparity of the interest points. For this process a four stage Difference of Gradient (DOG) pyramid [40] the stereo pair is used. Each interest point traverses through the pyramid in a top-down approach to refine the location of the interest point using Sum-of-Absolute-Differences (SAD) and Normalised Cross Correlation (NCC). Later using normalized cross correlation, the local variation of +/- 1-3 horizontal rows is computed. Later the pixel location is refined by using sinusoidal sub pixel
interpolation. The same location is traversed to lower layers of the pyramid to make a more precise measure of the location of the pixel in the right image.

The difference in the horizontal location gives the disparity value of the particular interest point. This disparity value, when combined with the camera base line distance and its focal length, results in actual depth, providing each interest point with a 3rd dimension, \( z \).

![Figure 4-20 Feature tracking through DoG [40] pyramid.](image)
Figure 4-21 Feature matching from left to right pyramid.

Figure 4.23 shows the region that’s being covered during the search for the correspondence feature. Assuming that the image is of size $20 \times 20$, then their corresponding pyramid sizes will be $10 \times 10$, $5 \times 5$ and $3 \times 3$ respectively. Let’s assume that there is a feature in $(10,10)$, of the left image, then in the top of its pyramid the feature will be in $(1.25,1.25)$. Assuming that the feature shows very less variance in its location at the top level of the pyramid, the feature is first assumed to be in the exact same location in the top level of the right pyramid. Due to epipolar geometry, the feature would lie on the same row, but would vary along the column index. Searching for three pixels around the known location would actually be a full image search as the three columns of the same row depicts a compressed version of the whole image. If the image size increases to $100 \times 100$, then the $\pm 3$-pixel search on the top would result in 56-pixel search, which is half the image. The blue region in the image shows a projection of the search area from its upper pyramid levels. The black window shows the current search window.

It would take 100-pixel search operations for a brute force feature matching even within the epipolar line. But by using SAD/NCC, the number of search operations will be 28 ($7 \times 4$). This is constant even if the size of the image increases.
and will be able to search feature matching pair with half the image size variance while the brute force operation count keeps increasing. Hence the computation time for finding the feature correspondence, is much smaller and the efficiency is much higher. This efficiency can be further enhanced by using sub-pixel interpolation.

Assuming that the cameras are calibrated and the images from those cameras are rectified; defining the disparity at an exact pixel location is characterized by the motion of these pixels for the change in perspective. In Figure 4.23 $P1$ and $P2$ are two interest points in a scene. If $P1$ is closer to the stereo camera setup than $P2$, then the motion of these points in the stereo image set will be inversely proportional to their real world distance. The farthest points will show smaller distance and the closer points show huge distance.

Figure 4-22 Sinusoidal Sub pixel interpolation.

Figure 4-23 Motion of a pixel w.r.t to its depth.
To find out the mathematical relation between the motion of the pixel to the real
world depth at every feature, let’s assume a stereo camera setup as shown in Figure
4.24.

![Stereo Camera Setup](image)

Figure 4-24 Geometrical representation of stereo camera setup.

As clearly indicated in the image $x_L$ and $x_R$ are the horizontal distances of the interest
point. It can be seen that two congruent triangles are formed, and that the ratio of
their side will be equal.

![Triangular Congruency](image)

Figure 4-25 Triangular congruency in the stereo camera setup.

Hence

\[
\frac{b}{Z} = \frac{(b + x_R) - x_L}{Z - f} \implies d = x_L - x_R = \frac{f \cdot b}{Z}
\]

(4.14)

where: $b =$ baseline distance between the cameras
\[ Z = \text{real world depth} \]
\[ (x_L - x_R) = \text{disparity of the interest point} \]
\[ f = \text{focal length of the cameras} \]
\[ x_L, x_R = \text{x distance of the features from the origin of the image plane} \]

Here it is clearly evident that the disparity is inversely proportional to the actual depth and with the focal length, baseline distance and the disparity, we could calculate the real world depth of the images. This \( Z \) information converts 2D interest points to 3D interest points.
4.7. **Pose Estimation**

By now we have two sets of 3D features, one set from the present and one set from the previous frame. Finding out the incremental change in pose of the tracked feature set from the previous frame to the current frame is the next task. The change in pose from frame $t-1$ to frame $t$ can be formulated as shown in (4.15).

$$ P_{\text{New}} = S \cdot R \cdot P_{\text{Old}} + T $$  

(4.15)

Where

- $P_{\text{New}}$ = Pose of the current frame
- $P_{\text{Old}}$ = Pose of the old frame
- $S$ = Scale
- $R$ = rotation
- $T$ = translation

Equation (4.15) shows that the new pose is a function of translation, rotation and scaling parameters, each of which need to be estimated. Horn’s method for absolute orientation [1] will be used to estimate these parameters using the point cloud information from the current previous frames. Horn’s method uses weighted least squares and quaternions to find a closed solution of absolute orientation. There are seven unknowns in this problem: scaling($S$), x-axis translation ($T_x$), y-axis translation ($T_y$), z-axis translation ($T_z$), x-axis rotation ($R_x$), y-axis rotation ($R_y$) and z-axis rotation ($R_z$). Rotation along each axis can also be called yaw, pitch and roll. To solve for the seven unknowns, we need at least three 3D points. For example, if we used three points, we have nine values to solve for our seven unknowns. More 3D features reduce error and increases accuracy.

The general approach for solving the scaling, rotation and translation values is summarized below.

1. Compute the mean of the point cloud $pc1$ and $pc2$ each for the current frame and the previous frame.
2. Compute the mean center for both $pc1$ and $pc2$.
3. Compute the co-variance matrix for both $pc1$ and $pc2$.
4. Apply singular value decomposition to the covariance matrix.
5. Calculate the rotation matrix using the SVD parameters.
6. Translation can be found using the mean values of the point cloud.
In our approach, we use a RANSAC based method to find out the mean pose for all the point cloud features. We divide the point cloud into subsets and calculate the pose matrix for each subset. The median pose matrix from various feature subsets is chosen. This step avoids any sudden motion until and unless the majority of features induce motion. This approach helps in rejecting outliers. Each feature subset will then receive a weight for their contribution to the mean pose value as shown in (4.16).

\[ w_i = \begin{cases} \frac{1}{\|c_i\|}, & \text{if } \|c_i\| < r \\ 0, & \text{otherwise} \end{cases} \]

\[ c_i = \hat{p}_{predicted,i} - p_{New,i} \]

(4.16)

where:
- \( \hat{p}_{predicted,i} \) = pose prediction from feature i for previous frame
- \( p_{New,i} \) = pose prediction from the current frame by feature i
- \( c_i \) = prediction error.
- \( w_i \) = weighting factor for feature i

Each feature can be predicted for its new pose based on the pose matrix of the previous frame. The error in the actual pose and the predicted pose of each feature can be used to know the quality of information that the feature can contribute to the overall pose. Features on static backgrounds will have less error and contribute heavily towards the overall pose. If the error is small, the 3D point cloud pair is assumed to have good quality and the weight factor becomes higher. This means, that this point pair will have more influence in the egomotion estimation. The bigger the error, the smaller the weight and the smaller the influence of the 3D point cloud pair on the pose estimation process. If the error is too big, then we remove the point pair completely by setting its weight to 0.

The point cloud obtained for \( F_t \) and \( F_{(t-1)} \) may be any corner in the scene, and may lie on moving objects like cars which contribute their own motion, distorting the vehicle motion estimation process. This induces more error in the estimation process and hence it is critical to only select inlier point cloud features and remove ones that falsify the motion estimation process. The process to eliminate such outlying features adopted in this research evaluates the information contribution of each candidate feature towards the resulting motion estimation.
Feature points with higher the information content indicate the feature lies on a static background and is within the traceable distance of the stereo camera. Such features are very important as they provide more information (more information in the context refers to the covariance of the features pose matrix to the overall pose matrix, more like K-nearest neighbors (KNN) [31] clustering of features). Features that provide less information, or that have a motion opposing the overall motion of the frame, can be considered to be outliers, as depicted in Figure 4.27.
Chapter 5  : Experiments

A variety of experiments very carried out for visual odometry estimation. These experiments are validated numerically by using ground truth provided from standard datasets and visually by implementing the algorithms to an autonomous golf cart. (The autonomous golf cart at RIT is a multi-year senior design project directed by Professor Raymond W Ptucha.) The experimentation investigates the ability of the proposed algorithms to tackle issues such as separating static and dynamic background for accurate visual odometry estimation, removing outlier feature points from moving objects in the scene, and reducing error accumulation over time without the use of loop closure, additional sensors or localization techniques.

The first set of experimentation aimed at solving issues related to obtaining a constant stream of features that could be processed. Use of adaptive feature detection along with windowing allowed the algorithm to get a constant stream of uncoagulated information in the form of features. These features were spread across the images. Figures 4.14 and 4.15 show that the proposed algorithm produced a constant number of reliable and robust features from most of the images and that these features were sparse.

Figure 4.14 shows the feature count varying drastically over consecutive frames and depicts variable flow of information to be processed for visual odometry estimation. Figure 4.15 shows that the information flow is constant for almost all the frames. This experiment was done on 28000 frames. The time for execution of each step in the process is tabulated for all the frames of KITTI dataset, to provide an analyses of the how fast is the process in Table 5-1.

The second set of experiments was done to see to increase the stereo matches and to get their accurate location by using sub pixel interpolation to find the location of matched feature pairs. The pixel location of FAST features was kept to whole numbers and then the number of features that were matched between left and right frames during depth measurement and between $t$ and $t-1$ frames via optical flow was kept track of.

The same experiment was conducted with refining the pixel location of matched pairs using a sinusoidal interpolation function to obtain a more precise location of the feature. This extra fractional information yielded an 8.6% boost in the number of features over the original feature count.
Table 5-1 Subpixel regression Statistics.

<table>
<thead>
<tr>
<th>Regression Statistics</th>
<th>Integer disparity</th>
<th>Sub-pixel disparity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple R</td>
<td>0.992927627</td>
<td>0.998857408</td>
</tr>
<tr>
<td>R Square</td>
<td>0.985905273</td>
<td>0.997716121</td>
</tr>
<tr>
<td>Adjusted R Square</td>
<td>0.984898507</td>
<td>0.997552987</td>
</tr>
<tr>
<td>Standard Error</td>
<td>2.224600023</td>
<td>0.836599709</td>
</tr>
<tr>
<td>Observations</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

The sub pixel interpolation also helps in disparity accuracy. Figure 5.1 shows a distribution of depth measurements. The black line shows the actual depth, while the blue and the red spots mark the depth measurement made using sub-pixel and the integer pixel locations. The estimated depth should lie closer to the black line and the sub pixel location’s depth calculations lie closer to the black line as the depth increases. Table 5.1 shows the same compared using the regression statistics. These different approaches show a reduction in the depth measurement errors across increasing depths.

Another set of experiments were carried out to see if the featuring selection and tracking process could cope with variation in frame exposure. The concern is that drastic over and under exposure of consecutive frames may lose many features due to variation in intensity. This is because the feature tracker assumes that the features vary in position but their intensities mostly remain the same which is not true in this case. This experiment was conducted using the RIT golf cart and the approach did exceptionally well in this approach.
Table 5-2 Execution time for each step.

<table>
<thead>
<tr>
<th>Max Nr. Features</th>
<th>1000 features</th>
<th>9600 features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectify Image</td>
<td>9.98 ms</td>
<td>9.98 ms</td>
</tr>
<tr>
<td>Image Pyramid</td>
<td>8.56 ms</td>
<td>8.56 ms</td>
</tr>
<tr>
<td>Optical flow</td>
<td>1.21 ms</td>
<td>6.86 ms</td>
</tr>
<tr>
<td>Disparity</td>
<td>2.12 ms</td>
<td>12.90 ms</td>
</tr>
<tr>
<td>Disparity to 3D</td>
<td>0.28 ms</td>
<td>1.32 ms</td>
</tr>
<tr>
<td>Horn’s method</td>
<td>0.60 ms</td>
<td>3.75 ms</td>
</tr>
<tr>
<td>Detect New features</td>
<td>1.91 ms</td>
<td>8.58 ms</td>
</tr>
<tr>
<td>Features weighs and motion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>refinement</td>
<td>0.80 ms</td>
<td>5.66 ms</td>
</tr>
<tr>
<td>Min distance enforcement</td>
<td>0.09 ms</td>
<td>2.08 ms</td>
</tr>
<tr>
<td>Complete algorithm</td>
<td>25.91 ms</td>
<td>61.2263 ms</td>
</tr>
<tr>
<td>Average fps</td>
<td>38.59 fps</td>
<td>16.33 fps</td>
</tr>
</tbody>
</table>

This was because of the adaptive feature detector was capable enough to generate features for various contrast regions in the image. And during the over exposure/under exposure situations there were enough new features generated which could be tracked more efficiently than the traditional approach. Such optimizations to the algorithm have made it less time consuming and fast enough to be real-time compliant. The execution time for each stage in the proposed algorithm is shown in table 5.2. The Image corrections and creating the DOG pyramid is more time consuming, but the overall time for execution is around 25ms. Less execution helps us process more frames per second and hence increase the sensitivity of the algorithm to small motion changes (More frames can track motion more accurately).

One of the biggest drawbacks of any odometry estimation process is the accumulation of error over time. The estimated pose from the previous to the current time step, contains error, and this error accumulates with each successive frame
processed. To minimize this error with visual information, the transformation matrix from previous \( m \) frames is used to estimate the current frames transformation matrix. This approach is depicted in Figure 5.2.

Figure 5.1 Disprity results by using sub pixel location to intergral location.

![Figure 5.1 Disprity results](image)

Figure 5.2 Transformation matrix between \( m \) frames.

In Figure 5.2, \( m \) depicts the number of frames that are considered to possess some visual information that could optimize the current transformation matrix and give more accurate results. Here, \( T_i \) represents the transformation matrix between frames. The error between the point cloud information and the transformed point cloud is to be minimized as per (5.1).
\[ \sum_{e_{ij}} \left\| F_i - T_{e_{ij}} C_j \right\|^2 \] (5.1)

The results for the visual odometry estimation process after all these optimizations and analysis are shown in Figures 5.3 and 5.4. These results were computed from 300 frames from the KITTI dataset, sequence 08. This Figures show the velocity and the angular momentum variation for these 300 frames using the methods proposed in this thesis.
Figure 5.3 Vx, Vy and Vz result comparison for KITTI dataset.
Figure 5.4 Yaw, Pitch and Roll results comparison for KITTI dataset.
The RMS error for the KITTI dataset based on the date when the data was captured can be seen in table 5-3.

Table 5-3 RMS Error for data based on date

<table>
<thead>
<tr>
<th>RMS Error</th>
<th>Vx</th>
<th>Vy</th>
<th>Vz</th>
<th>Pitch</th>
<th>Yaw</th>
<th>Roll</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-09-26</td>
<td>1.819003802</td>
<td>1.374765862</td>
<td>0.628549021</td>
<td>0.043933793</td>
<td>0.055646991</td>
<td>0.248387552</td>
</tr>
<tr>
<td>2011-09-28</td>
<td>0.083204798</td>
<td>0.034568495</td>
<td>0.043196555</td>
<td>0.001673344</td>
<td>0.003127362</td>
<td>0.001403155</td>
</tr>
<tr>
<td>2011-09-29</td>
<td>0.209080514</td>
<td>0.059380969</td>
<td>0.169566996</td>
<td>0.003253539</td>
<td>0.00775988</td>
<td>0.002787566</td>
</tr>
<tr>
<td>2011_09_30</td>
<td>0.364352081</td>
<td>0.73545988</td>
<td>0.222077774</td>
<td>0.00747232</td>
<td>0.010497656</td>
<td>0.006130403</td>
</tr>
<tr>
<td>2011_10_03</td>
<td>0.410125117</td>
<td>0.574803923</td>
<td>0.428756564</td>
<td>0.012539607</td>
<td>0.019349872</td>
<td>0.025116358</td>
</tr>
</tbody>
</table>

On close investigation, the results of data captured on 09/26/2011 have the highest error over data captured on other days. The least error is seen on 09/28/2011. These effects are because of the data length. Motion estimation gets better over consecutive frames and more the number of consecutive frames, the error in incremental motion estimation reduces. This error is not the same as the error accumulation of the overall trajectory rather is the error in the motion estimation between frames. As 09/28/2011 has more images per session, than having more sessions itself as in the case of 09/26/2011, the RMS error reduces over time. The RMS error for the same dataset segmented based on the content is tabulated in table 5-4.

Table 5-4 RMS Error for data based on content.

<table>
<thead>
<tr>
<th>RMS Error</th>
<th>Vx</th>
<th>Vy</th>
<th>Vz</th>
<th>Pitch</th>
<th>Yaw</th>
<th>Roll</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>1.502489948</td>
<td>1.25750709</td>
<td>0.37928101</td>
<td>0.010902831</td>
<td>0.014910277</td>
<td>0.007565111</td>
</tr>
<tr>
<td>Residential</td>
<td>1.096476491</td>
<td>0.946146191</td>
<td>0.464352015</td>
<td>0.032525413</td>
<td>0.041257366</td>
<td>0.183368332</td>
</tr>
<tr>
<td>Highway</td>
<td>0.628319795</td>
<td>0.551897685</td>
<td>0.526633115</td>
<td>0.015644485</td>
<td>0.02508115</td>
<td>0.035588564</td>
</tr>
<tr>
<td>Campus</td>
<td>0.09860991</td>
<td>0.028252349</td>
<td>0.058620277</td>
<td>0.00186406</td>
<td>0.004048608</td>
<td>0.001901811</td>
</tr>
<tr>
<td>Person</td>
<td>0.011367087</td>
<td>0.023763893</td>
<td>0.014772203</td>
<td>0.000469363</td>
<td>0.00080434</td>
<td>0.000363688</td>
</tr>
</tbody>
</table>

The data in the above table shows very less error for sessions containing persons in the background because there is no motion in the images. In City session data, the huge non static background motion causes more variation in motion estimation results and hence the huge RMSE error.
The results for the KITTI data set for various sequences having city and highway images are shown in Table 5-5. One key observation to be made is that on the highway image sequences, the velocity information is less accurately estimated because of the high velocity and a greater distance travelled by the car between frames. In the case of city dataset, the results are better for velocity as the translational motion is small and is sufficiently recorded in the smaller frame rate. In the city image sequence, the outliers are high when compared to the highway image sequence, but still manage to produce good results because of the outlier rejection.

For the oxford dataset

Table 5-5: Translational and rotational result for all the sequences of KITTI dataset.

<table>
<thead>
<tr>
<th>Highway</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td>$V_x$</td>
<td>68.60%</td>
</tr>
<tr>
<td>$V_y$</td>
<td>68.77%</td>
</tr>
<tr>
<td>$V_z$</td>
<td>64.90%</td>
</tr>
<tr>
<td>Pitch</td>
<td>95.20%</td>
</tr>
<tr>
<td>Yaw</td>
<td>93.45%</td>
</tr>
<tr>
<td>Roll</td>
<td>96.29%</td>
</tr>
</tbody>
</table>

The next set of results is for the New College dataset from Oxford. These results show larger variation in translation. This is because the ground truth translation results are not with respect to the change in motion, but rather change is center of gravity position. Hence we see less accuracy for the translational velocity. The results are shown in Figures 5.5 and 5.6, and are tabulated in Table 5.4 & 5.5.
Figure 5.5 Vx, Vy and Vz results for New college Dataset.
Figure 5.6 Yaw, Pitch and Roll results for New College dataset.
Table 5-6 New college dataset results for translation and rotation.

<table>
<thead>
<tr>
<th>New College Dataset</th>
<th>Motion</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vx</td>
<td>32.29%</td>
<td></td>
</tr>
<tr>
<td>Vy</td>
<td>89.03%</td>
<td></td>
</tr>
<tr>
<td>Vz</td>
<td>86.52%</td>
<td></td>
</tr>
<tr>
<td>Pitch</td>
<td>80.27%</td>
<td></td>
</tr>
<tr>
<td>Yaw</td>
<td>87.32%</td>
<td></td>
</tr>
<tr>
<td>Roll</td>
<td>83.10%</td>
<td></td>
</tr>
</tbody>
</table>

A comparison of our results with the best approaches is shown in the Table 5.7.

Table 5-7 Result comparison with state of the art approaches.

<table>
<thead>
<tr>
<th>Method</th>
<th>Approach</th>
<th>Execution time (ms)</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORB SLAM [32] (2015)</td>
<td>Tracking</td>
<td>74</td>
<td>72.33%</td>
</tr>
<tr>
<td>ORB SLAM 2 [32] [33] (2016)</td>
<td>Tracking + loop closure detection + global relocalization</td>
<td>100</td>
<td>98.85%</td>
</tr>
<tr>
<td>SOFT [26] (2016)</td>
<td>Tracking + loop closure detection + IMU integration</td>
<td>100</td>
<td>98.97%</td>
</tr>
<tr>
<td>ROOC [35] (2016)</td>
<td>Tracking</td>
<td>21</td>
<td>78.56%</td>
</tr>
<tr>
<td>Proposed approach</td>
<td>Tracking</td>
<td>25.91</td>
<td>86.2%</td>
</tr>
</tbody>
</table>
Chapter 6  : Conclusion

In the past year alone, six publications have been published describing the importance of features in visual odometry estimation. It’s also evident that motion tracking using block of regions or pixels would result is efficient detection and rejection of outliers but would also suffer from issues such as less information content (less feature count). Such approaches must use either inertial sensors or post processing techniques to obtain enough data to estimate accurate motion. Alternatively, the use of a sparse set of features is always prone to outliers and will eventually accumulate error over consecutive motion estimation processes.

A perfect balance between both approaches is proposed in this thesis. Efficient outlier rejection is achieved by feature profiling and feedback motion correction. An adaptive feature generation and windowing approach helps in generating sufficient features over time. The use of sub pixel interpolation helps the process of tracking and hence provides more accurate motion tracking.

The handling of exposure variation which may cause a large loss of useable features is only possible with the help of adaptive feature generation and execution time reduction. The execution time, or the number of frames that could be processed in a second, is directly proportional to the sensitivity of the algorithm to detect the slightest motion. Compared to other approaches, the proposed approach takes 40% less execution time and hence can process a very high frame rate of around 38 fps. Outliers on the dynamic backgrounds such as pedestrians and other moving objects on the road possess their own motion. The proposed methods are able to remove such outliers, which can be seen in the KITTI city sequence, where the opposing traffic and the pedestrians were handled very efficiently. These results indicate that this approach is suitable for visual odometry estimation in real-time for real world driving scenarios.

Algorithms used in state of the art systems such as ORB Slam and SOFT only work well under some post processing techniques or by loose coupling of visual data with other hardware sensors. The best approach for a real-time implementation is to reduce the accumulated error over time so that corrections can be done only when extra information is available. The proposed approach solves all of the aforementioned issues.
Bibliography


7.1. Stereo Camera Setup

The stereo camera setup mounted on the RIT MIL Golf cart can be seen in the picture below.

The setup was used to capture stereoscopic images inside RIT campus. The setup consisted on the two cameras and some precision measurements. The steps followed to create a stereo camera setup are outlined below.

- Two completely identical surveillance cameras were chosen which outputs data using a Local Area Network (LAN) cable.
- We gathered the camera sensor properties from the data sheet of the device. Some amongst the main parameters required are the sensor dimension, pixel dimension, ROI and focal length of these cameras.
- Using the online tool (https://nerian.com/support/resources/calculator/) , we calculated the baseline distance between the cameras to minimize the depth calculation error. The results are shown in figure 7-1.
- The baseline distance is the distance between the cameras, to mimic bionic eyes. The concept is that two identical cameras viewing an object at different perspectives can identify the displacement of the object with change in perspective. This information is later solidified into depth with the help of the camera properties that we noted down in the earlier step.
- The cameras were placed at a near baseline distance of 11.5 cm apart and were screwed to the roof firmly.
Results

Image sensor metrics
Sensor width: 4.27 mm
Sensor height: 3.20 mm
Pixel size: 6.67 μm
ROI width: 4.27 mm
ROI height: 3.20 mm

Lens Metrics
Focal length: 3.33 mm
Diagonal angle of view (ROI): 77.3°
Horizontal angle of view (ROI): 65.2°
Vertical angle of view (ROI): 51.3°

Stereo Geometry Metrics
Baseline distance: 11.1 cm (4.37 in)

<table>
<thead>
<tr>
<th>Depth (m)</th>
<th>Depth Error (cm)</th>
<th>Depth (ft)</th>
<th>Depth Error (in)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.1</td>
<td>0.5</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>2</td>
<td>1.8</td>
<td>2</td>
<td>0.7</td>
</tr>
<tr>
<td>5</td>
<td>11.5</td>
<td>5</td>
<td>4.5</td>
</tr>
<tr>
<td>10</td>
<td>47.2</td>
<td>10</td>
<td>18.6</td>
</tr>
<tr>
<td>20</td>
<td>198.0</td>
<td>20</td>
<td>78.0</td>
</tr>
<tr>
<td>50</td>
<td>1453.5</td>
<td>50</td>
<td>572.2</td>
</tr>
<tr>
<td>100</td>
<td>8196.7</td>
<td>100</td>
<td>3227.1</td>
</tr>
<tr>
<td>200</td>
<td>181818.2</td>
<td>200</td>
<td>71582.0</td>
</tr>
</tbody>
</table>

7-1 Camera baseline distance.

Figure 7-2 Stereo camera setup on golf kart.
Two configurations of the camera poses can be used to capture precise depth information as shown in fig 7-2. We use the Parallel configuration as the region of interest (ROI) is vastly spread across the image and the convergence needs to be adjusted in post processing (by shifting the zero-plane).

Figure 7-3 Stereo Camera Configuration.
7.2. Accessing images from Cameras

The cameras used for the stereo camera setup, stream images through the network and hence the network has to be configured to receive images from these cameras to a specific port on your laptop/desktop. The setup includes setting cameras to a specific IP address, and setting the port on the desktop/laptop to receive data only from these specific IP addresses to reduce latency in receiving images. The steps to perform these are listed below.

- Connect the cameras LAN cable to the laptop/desktop Ethernet ports and connect the cameras individual power cables to 12-volt power supply.
- To access the cameras web interface, and to change the setting log on to 198.162.0.64 IP address and login with username as admin and password as password as shown in figure 7-4.

![Login snapshot of Hik-Vision Camera.](image)
Once logged in, access the options shown in figures 7-5, 7-6 and 7-7 to change the camera resolution output, fps and streaming protocols with/without authentication.

Figure 7-5 Output Video config snapshot.

Figure 7-6 Output Camer ID snapshot.

Figure 7-7 Output Streaming protocol and its authentication snapshot.
• Configure the cameras to work with RTSP protocol and setup the camera to work with a specific IP. Also make sure that the camera output resolution matches your requirement. For the ease of accessing images at the fastest rate, we used 680x420 resolutions at 18 fps.

• Perform the same setup for the other camera too.

• On the receiving end, setup each Ethernet port to receive images from the camera with that specific IP. Using ifconfig command to setup the

ifconfig <Ethernet ID> <IP address> netmask <netmask for the IP>

Ex: ifconfig eth0:1 192.168.1.1 netmask 255.255.255.0

subnet mask and Ip address as shown below.

• Now using the RTSP protocol, we can access the images from the camera

• To test the setup, use a video viewer that supports RTSP and provide

  RTSP://<IP address>::<port number>/
  Ex: RTSP://192.168.1.102:554/

the IP to view in real time.

• The images can be captured using Opencv videocapture object by providing the same IP address. (Requires OpenCV3 to be built with ffmpeg libraries).

• The KGCOE Gitlab project consists of a C++ code to capture video in realtime from both the cameras with the timestamps (dataAcquisition.cpp).
7.3. Calibration of the Cameras

The calibration of these cameras is needed to adjust the small errors in alignment. By measuring these alignment error, we can project one of the image into the others plane so their camera optical axis becomes parallel. This reduces the complexity of information matching between images. The steps are clearly stated below.

- Setup the cameras to start capturing images.
- Hold the checker board pattern, as shown in figure 7-9 in front of the cameras to capture images simultaneously from both the cameras. Make small tilts and movements to the front and side of the setup, to capture different rotations and scale of the pattern.
- From the above step we get a set of images showcasing the pattern, from left and right cameras of the setup. Use the Matlabs calibration toolbox (https://www.mathworks.com/help/releases/R2013b/vision/ug/find-camera-parameters-with-the-camera-calibrator.html) to get the calibration matrix. Use of multiple pairs reduces the error in calculating the calibration matrix.
- During processing, project the right images using the calibration matrix to obtain a near perfect parallel configuration image pairs.
Figure 7-9 Checker board pattern for camera calibration.
7.4. **Compile and Debug the code:**

The code is uploaded to KGCOE-git and can be directly cloned into the project directory using git clone command. To compile and run the algorithm, in real time the best way is to setup a working environment in C++.

- Download and install any visual studio version on windows platform with visual C++ libraries.
- Download and install latest version of openCV 2.4.x.
- Create a visual studio project, and add the files from the git to the project.
- Make sure the property sheets are set and all the addition dependencies are linked to the project.
- The main.cpp file acts as the access point to the project and all the configurations can either be passed as a argument or can be hard coded in to this file.
- The main.cpp file requires you to provide the path to the data and the calibration file. It outputs the pose prediction on to a file named Motionestimation.csv. If the ground truth path is set then the project outputs prediction error in the same file.