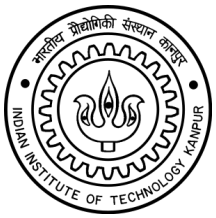
Visual Odometry In Self Driving Car



Indian Institute of Technology, Kanpur

Undergraduate Course Project

K.V.Sameer Raja(12332)

Mentor - Prof. Gaurav Pandey
Co-advisor - Prof. Subhajit Dutta

Abstract

This is the documentation of the work we have done to implement visual odometry in the self driving car project. Currently we have tested monocular visual odometry on the Ford Campus Vision and Lidar dataset. The implementation is in C++ using Opencv library and works in real time. We used only the camera information to predict the motion of car without any prior knowledge of surrounding and other sensor data. We have used FAST and SIFT to detect features. KLT tracker with RANSAC for outlier rejection is employed for tracking features.

Contents

1 Introduction

Visual Odometry is the process of estimating the vehicle's trajectory using a single or multiple camera rigidly attached to the vehicle. Similar to wheel odometry, visual odometry incrementally estimates the position of the vehicle without taking into account of the past inputs. In Monocular Visual Odometry, the path obtained is a scaled version of the ground truth. Visual Odometry was first used by NASA in Rovers Mars.

2 Motivation

In self driving car, the challenge is to solve the problem of braking, steering and accelerating which are done automatically without driver attention. All of the three paradigms require correct absolute path prediction. Other techniques which can be employed for same task are :

- Wheel odometry - It has slipping problem in uneven terrain
- Global Positioning System(GPS) - It is erroneous and not always available
- Inertial Measurement Unit(IMU) - I.M.U's which generate accurate path are generally too costly. There are some I.M.U's which are available at comparatively moderate cost but the path generated by them is often erroneous.

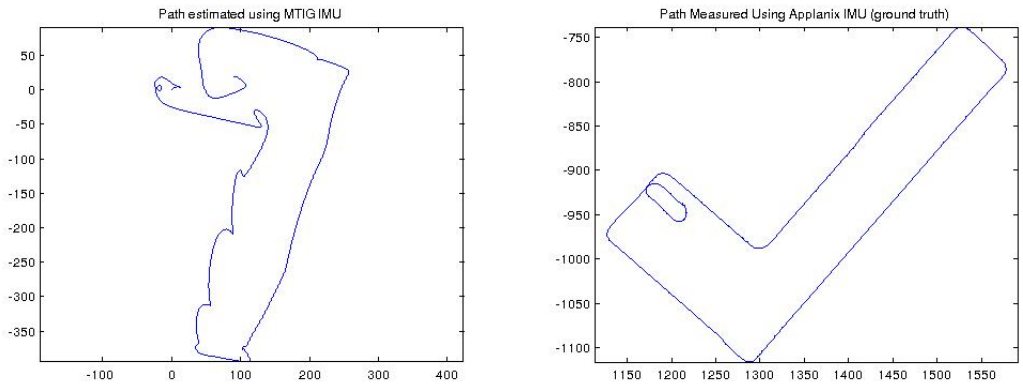


Figure 1: Normal I.M.U. and high cost I.M.U.

Visual odometry promises appropriate results in almost all kind of environments with relatively low cost apparatus(cheap sensors in comparison to other sensors). This has resulted in increasing focus of research in the area. V.O works correctly when we have sufficient illumination and enough number of interesting points (features) in the frames. Also the consecutive frames should have overlap of common features, which enables us to track those features. The aim of this undergraduate project is to solve the problem of predicting trajectory in self driving car using visual inputs alone in for monocular system.

3 Problem Formulation

The camera is mounted on the moving car and takes images with certain fixed frames per second to have sufficient overlap of scenes in two consecutive frames. Let the set of images be $I_0, I_1, ...I_n$. The aim is to estimate the transformation matrices relating

the consecutive camera poses which will be utilised to estimate the trajectory of the car. Without loss of generality we can assume that camera co-ordinates is same as vehicle's co-ordinate frame except for some translation. Let's denote the camera pose as $C_0, C_1, ...C_n$. The two consecutive camera poses are related as

$$C_n = C_{n-1} T_n$$

where

$$T_k = \begin{pmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{pmatrix}$$

is the transformation matrix consisting of rotation matrix $R_{k,k-1}$ and translation vector $t_{k,k-1}$ between instants k and k-1. C_0 is the initial camera pose . If we are able to find $T_1, ..T_n$ then any camera pose C_n can be computed by concatenating $C_0, T_1, T_2, ...T_n$ and thus we can find the trajectory by finding all the camera poses. We find some keypoints(discussed later) p_k^i and p_{k-1}^i and utilize those to find transformation matrix T_k. T_k is found by minimizing the L2 norm between the 2D feature sets p_k^i and p_{k-1}^i and solve the following objective function [8]

$$T_k = \arg\min_{T_k} \sum_i ||p_k^i - T_k p_k - 1^i|| \tag{1}$$

4 Fundamentals

In order to arrive at the correct transformation matrices we use the perspective camera model to map 3d points in universe to 2d points in camera pixels.

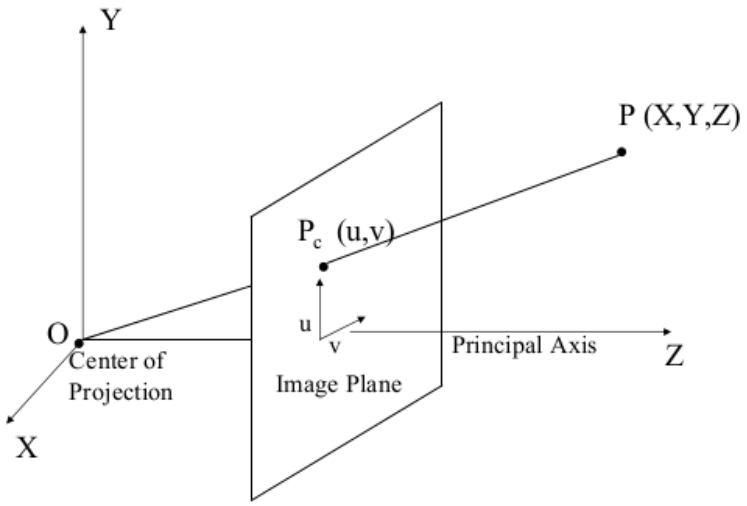


Figure 2: Image Projection Model

src : https://www.ics.uci.edu/ majumder/vispercep/cameracalib.pdf

4.1 Camera Calibration

Figure 2 shows the pinhole model of a camera. O is called the center of projection of camera and the image plane is always at a distance of f (focal length) from O. A point in

3D P(X,Y,Z) (in camera's frame) is viewed in image plane as $P_c(u,v)$ (here we assume that image plane origin ($O^{'}$) is the point(α) where principal axis intersects image plane). By the property of similar triangles,

$$\frac{f}{Z} = \frac{u}{X} = \frac{v}{Y}$$
$$u = \frac{fX}{Z}, \quad v = \frac{fY}{Z}$$

The above equations can be represented in homogeneous coordinates as

$$P_h = \begin{pmatrix} u^{'} \\ v^{'} \\ w \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = KP$$

$$P_c = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{u^{'}}{w} \\ \frac{v^{'}}{w} \end{pmatrix}$$

If the origin($O^{'}$) of image plane does not coincide with α then we need to adjust P_c accordingly by

$$u = \frac{fX}{Z} + c_x, \quad v = \frac{fY}{Z} + c_y$$

where (c_x, c_y) is the position of α (called principal point) from $O^{'}$. notice that P_c we derived until now is not exactly the point in the image because u v are in inches (or MKS system). So we need to correct that factor by multiplying m_x m_y which are pixels/inch in x and y directions respectively. So the overall rectified equations now looks like

$$u = m_x \frac{fX}{Z} + m_x c_x, \quad v = m_y \frac{fY}{Z} + m_y c_y$$

In matrix form it is represented as

$$P_h = \begin{pmatrix} u^{'} \\ v^{'} \\ w \end{pmatrix} = \begin{pmatrix} m_x f & 0 & m_x c_x \\ 0 & m_y f & m_y c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = KP$$

This K matrix is called intrinsic camera calibration matrix. If the camera's coordinate frame is not aligned with vehicles coordinate frame then we have to perform rotation and translation to align the above two frames. These matrices are called extrinsic calibration matrix. In this project, we are not making use of the extrinsic calibration matrix as there is only translation vector required to align camera frame with vehicle frame which donot effect our generated path.

5 Methodology

We were provided images which are adjusted for distortion. A brief outline of the algorithm followed is :

- **Feature detection**- Detect a considerable amount of features in the images. If number of features fall below a certain threshold a redetection is triggered.
- **Feature tracking**- Track the detected features in consecutive images I_t and I_{t-1}.

- **Epipolar Geometry**- With the pairs of tracked points obtained from above step, use Nister's five point algorithm with RANSAC to find essential matrix.
- **Estimating Motion Trajectory**- Extract rotation matrix and translation vector from the essential matrix and concatenate it to find the trajectory of motion.

5.1 Feature Detection

Instead of tracking all of the pixels we rather focus on features which are interesting parts of image that differ from immediate neighbourhood. To employ the repeatability of features we need invariance(Translational, Rotational, Scale and intensity invariance) and robustness of the features. Feature detectors can be broadly divided into two types as follows:-

- **Corner detectors** such as Harris (based on eigen values of second moment matrix) and FAST detector, because corners are relatively invariant to change of view. They are translational and rotational invariant but not scale invariant.
- **Scale Invariant Detectors** such as SIFT.

5.1.1 SIFT Feature Detector and descriptor

- **SIFT Detector**: Outline of SIFT detector:
  - **Construct a scale space**:- To achieve scale invariance, SIFT finds best scale to detect feature. To create a scale space, the resized images are progressively "blurred" out. Mathematically, "blurring" is done by the convolution of Gaussian Kernel with image

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{2}$$

    where L is blurred image,I is image,x and y are pixels,σ is scale and $G(x,y,\sigma)$ is the gaussian kernel as:-

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{3}$$

  - **LoG and DoG operations**:- The Laplacian of Gaussain(LoG) operation calculates second derivatives of gaussian which detects corners and edges, the potential points for features. LoG is given by :

$$LoG(x,y) = \nabla^2 L(x,y) = L_{xx} + L_{yy} \tag{4}$$

    But as we see, L is dependent on scale factor σ. So, to account for this, we multiply LoG(x,y) by σ^2 to produce scale-normalised LoG given by :

$$LoG(x,y,\sigma) = \sigma^2 \nabla^2 L(x,y) = \sigma^2 (L_{xx} + L_{yy}) \tag{5}$$

    But since second derivative is computationally expensive, LoG can be approximated with Difference of Gaaussian(DoG). DoG is obtained by difference of blurrings at different scales depicted in the figure below:-

$$DoG(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma) \tag{6}$$

LoG and DoG are related by:

$$(k-1)LoG(x,y,\sigma) \approx DoG(x,y,\sigma) \tag{7}$$

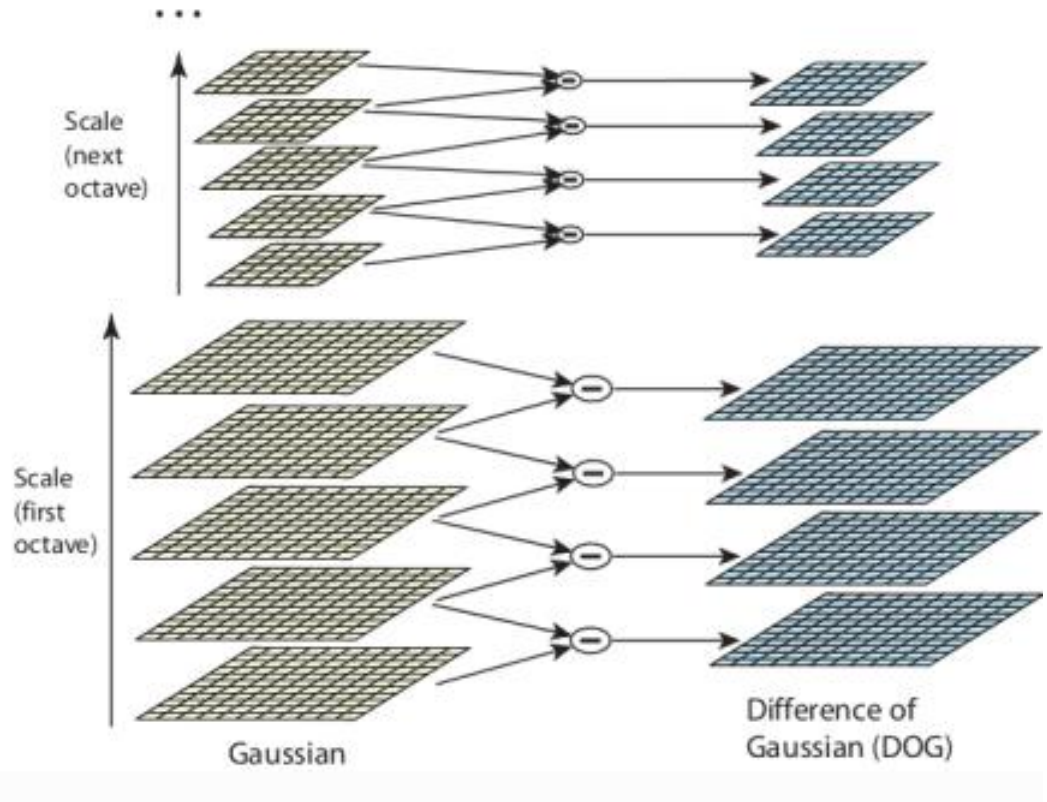


Figure 3: Difference of Gaussian (src : [4])

– **Detect keypoints**:- After finding Difference of Gaussian(DoG), we search for extrema over scale space by considering a neighborhood window around a point(say p_1) in a certain scale (say σ_1) and same sized window's in preceding and succeeding scales. If the intensity value at this point turns out to be an extrema then σ_1 is the optimal scale and that point(p_1) is feature point. The low contrast features and flat region are removed by examining the eigenvalues of Hessian matrix.

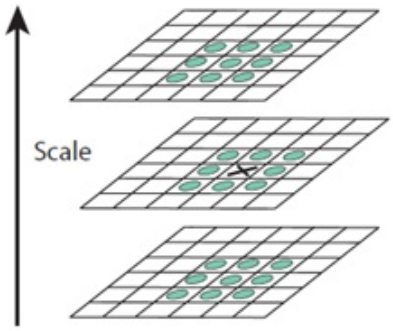


Figure 4: Extrema detection (src : [4])

- **SIFT Descriptor**
  For describing a SIFT feature mathematically, a 16 x 16 window around the detector is taken and divided into 4 x 4 grid cells. In each cell, the gradient orientations for all pixels calculated and are quantized to 8 bins. A frequency histogram of orientations is calculated for each of the cell. Concatenating the values of histogram frequencies in all cells gives 16 x 8 =128 dimensional vector for a SIFT descriptor. :[4]).

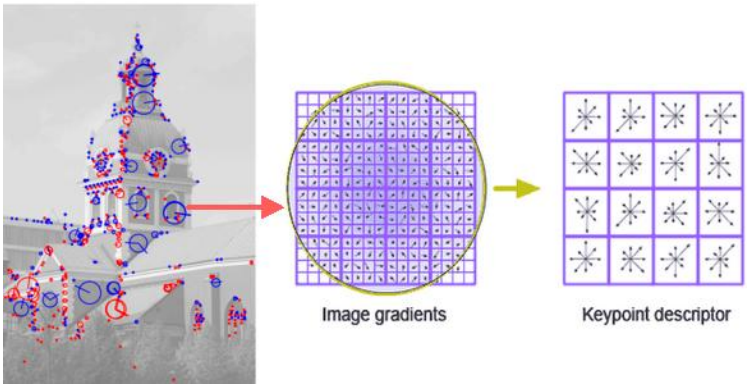


Figure 5: SIFT descriptor

src : http://www.codeproject.com/KB/recipes/619039/SIFT.JPG

5.1.2 Fast Feature Detector

SIFT and Harris corner are computationally expensive and time taking. FAST[7] detector is often used for real time video processing because of high speed performance. To find a corner it uses the heuristic that a corner will have a continuous set of brighter set of pixels or darker set of pixels than its own. It constructs a circle of 16 pixels and checks if it satisfies either of the following conditions:

- A set of N contiguous pixels which has brightness greater than the corner with at least a certain threshold.
- A set of N contiguous pixels which has brightness lesser than the corner with at least a certain threshold. The number of contiguous pixels,N and threshold are hyper parameter.

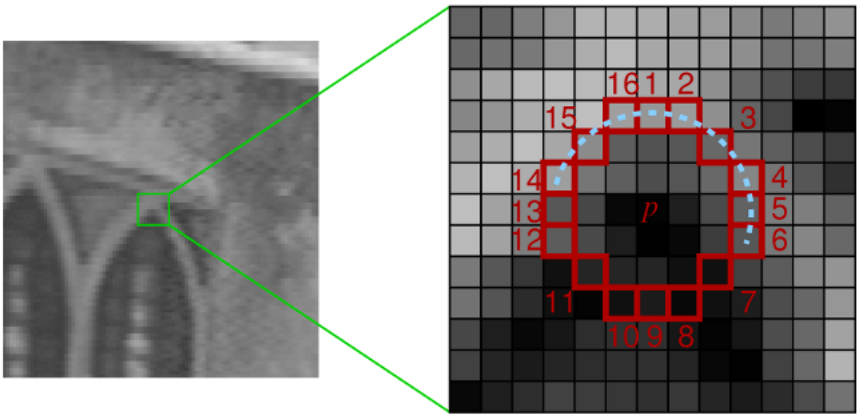


Figure 6: Fast feature detector (src : [7]

A High speed test for FAST [2] : To exclude a large number of non-corner points, it checks only for the points 1,9,5 and 13. It first checks if 1 and 9 are too brighter or darker than center pixel, if so, then it checks for 5 and 13. If the center pixel is corner then either of these pixels must be brighter or darker than certain threshold. Though it exhibits high performance but it has several weaknesses, prominent among them are:-

- It doesn't reject many candidates for $N < 12$
- The pixels selected are not optimal since corners might be tilted

Machine learning approaches are used to solve these problems.

5.2 Feature Tracking

After detecting the potential keypoints our aim is to estimate the point translation in consecutive frames. We have employed KLT tracker [9] to track the features.

5.2.1 Kanade Lucas Tomashi(KLT) Tracker

KLT tracker assumes following conditions:-

- **Brightness Constancy**- The same point projected in different frames looks same.
- **Small Motion**- The movement of point is not very far.
- **Spatial Coherence**- The motion of points are similar to their neighbour.

From brightness constancy we get

$$I(x,y,t) = I(x+u, y+u, t+1) \tag{8}$$

where x,y are pixel points,t is the time instant and u,v are displacement of the points. Assuming small motion and expanding R.H.S. using taylor series gives

$$I(x+u, y+v, t+1) \approx I(x,y,t) + I_x.u + I_y.v + I_t \tag{9}$$

Solving the above equation we get

$$\nabla I(x,y).(u,v) = 0 \tag{10}$$

($I_t \approx 0$ comes from our assumptions)
In the above equation we have two unknowns and single equation. To solve this we use spatial coherence assumption i.e. pixel's neighbour have the same (u,v) as the pixel. We take a 5 x 5 window which gives 25 equations as follows:-

$$\begin{pmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \cdots & \cdots \\ I_x(p_{25}) & I_y(p_{25}) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} I_t(p_1) \\ - \; I_t(p_2) \\ \cdots \\ I_t(p_25) \end{pmatrix}$$

The above is over-constrained linear equation which can be solved by least squares solution which is

$$\begin{pmatrix} \Sigma I_x I_x & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y I_y \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} - \; \Sigma I_x I_t \\ \Sigma I_y I_t \end{pmatrix}$$

From above equation we also solve the problem of finding which points to be tracked.If the matrix in L.H.S. has eigenvalues λ_1 and λ_2 and both of them are large,then it's a corner. If only one of them is large, then it's an edge. If none of them are large, then it's a flat region. We are concerned only about corners and edges.

5.3 Motion Estimation

The goal of visual odometry is to estimate motion which can be found from the transformation matrix by solving the first equation.The problem is a hard problem since the objective variable is a matrix . But a careful observation shows that instead of searching for a feature in whole of the next image we can constrain our search in a line called epipolar line. This is derived from epipolar constrain discussed in next section.

5.3.1 Epipolar Geometry

It uses the coplanarity condition of 3d universe point,their corresponding image point and Camera Centers. Let us denote this plane as π. Baseline is the line joining camera centers. The point where baseline intersects image plane is termed as epipole. Epipolar line is the intersection of the image plane with pi. Thus instead of searching for image features correspondence in whole of the image it searches in the epipolar line corresponding to feature [1].

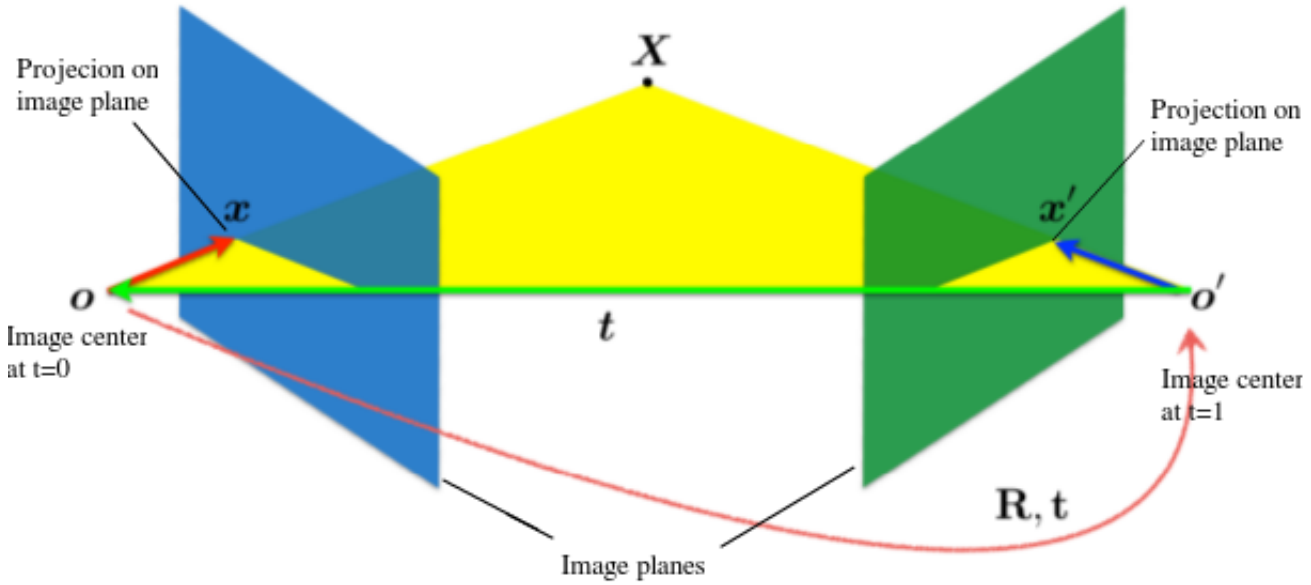


Figure 7: Epipolar Geometry (src : [1])

If we have calibrated observation,then in the figure above ,x ,x' are perpendicular to $(x \times t)$.Using rigid motion we can write

$$x' = R(x - t) \tag{11}$$

Using co-planarity condition we can write

$$(x - t)^T (x \times t) = 0 \tag{12}$$

Combining equation 6 and 7 we get

$$x'^T R(t \times x) = 0 \tag{13}$$

To represent cross-product in matrix form, $(t \times x)$ is represented as $[t_x]x$ where

$$[t_x] = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

Therefore

$$x'^T R(t \times x) = 0 \tag{15}$$

$$x'^T E x = 0 \tag{16}$$

Where E =R$[t_x]$ is called essential matrix.

5.3.2 Essential matrix computation

As discussed above once we have x and x', we can calculate essential Matrix E which satisfies equation(11). An essential matrix can be computed using Nister's 5 point algorithm. Equation (11) can be re-written as:

$$\widetilde{x}^T \widetilde{E} = 0 \tag{17}$$

where

$$\widetilde{x} = [x_1x_1' \;\; x_2x_1' \;\; x_3x_1' \;\; x_1x_2' \;\; x_2x_2' \;\; x_3x_2' \;\; x_1x_3' \;\; x_2x_3' \;\; x_3x_3']^T \tag{18}$$

$$\widetilde{E} = [E_{11} \;\; E_{12} \;\; E_{13} \;\; E_{21} \;\; E_{22} \;\; E_{23} \;\; E_{31} \;\; E_{32} \;\; E_{33}] \tag{19}$$

As there are five unknowns in E matrix (3 angles and 2 translation elements), we stack five such $\widetilde{x}$ points and form a 5×9 matrix (say $\widetilde{X}$). Nister[5] proposed a mathematical solution for the equation

$$\widetilde{X}^T \widetilde{E} = 0 \tag{20}$$

which gives the solution matrix E. It requires minimal 5 points which is better than previous 8 point algorithm. It involves calculating coefficients and finding roots of 10^th order polynomials. But from feature tracking we have so many pairs of points (x and x') which may also contain some outliers (due to least squares solution described above). So, we need to choose a set of 5 pairs from them. This selection is done using RANSAC.

- **RANSAC** : Feature correspondences using KLT tracker are often erroneous and there can be outliers. Least squares fitting uses all of the data and can be influenced by outliers. In order to have better feature correspondence, we uses iterative Random Sample Consensus, abbreviated as RANSAC [3], to remove outliers. At each iteration RANSAC uniformly selects a subset of data samples and then Essential matrix is computed. It checks if this essential matrix satisfies the equation (15) for all point pairs. The pairs which satisfies equation(15) are called inliers. RANSAC terminates if the count of the inliers is higher than specified amount by user. If the number is satisfied then it returns this matrix as final Essential matrix, otherwise it searches for points once again until it reaches maximum number of loops specified by the user.

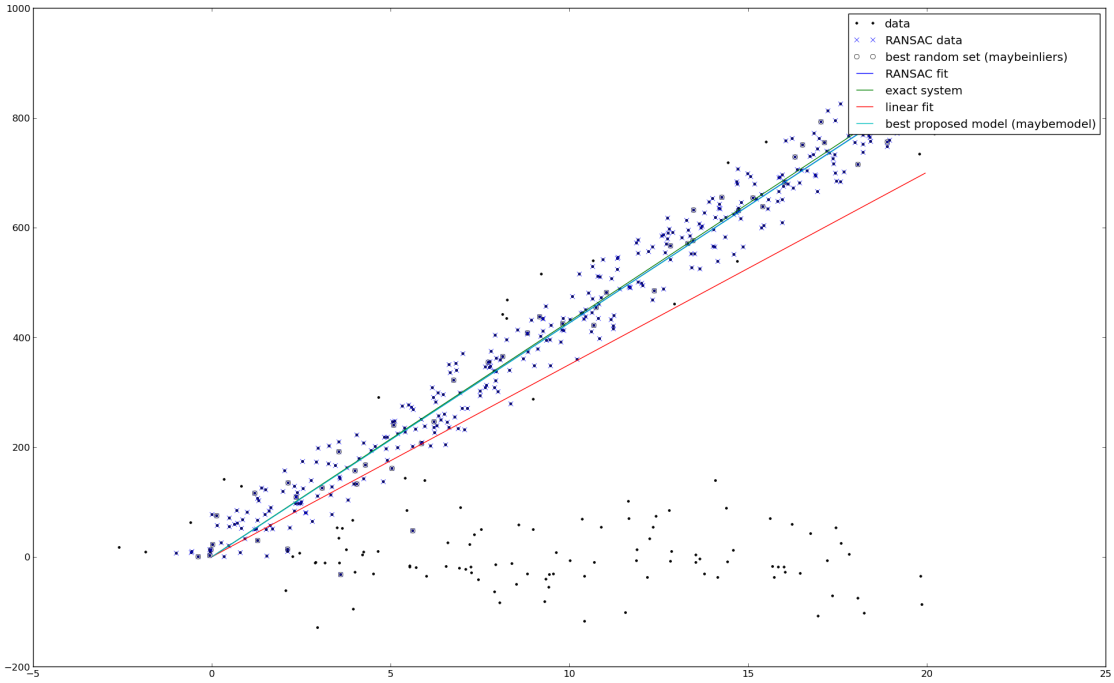


Figure 8: RANSAC Model

src http://i.stack.imgur.com/umP7k.png

5.3.3 Trajectory Estimation

At first rotation matrix and translation vector is estimated from the essential matrix. R and t can be obtained from the properties of essential matrix. The Singular Value Decomposition of E is

$$E = \left(\; U\Sigma V^T \; \right)$$

where U and V are 3 x 3 orthogonal matrices and

$$\Sigma = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Let us define

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Using skew-symmetric and orthogonal properties of t_x and R, there are two possible values of R as R_1 =UWV^T and R_2 =UW^TV^T. Also translation vector can either be t or -t, where $[t_x]$=$UW\Sigma U^T$. Thus there are four combination for rotation and translation as depicted by the figure below

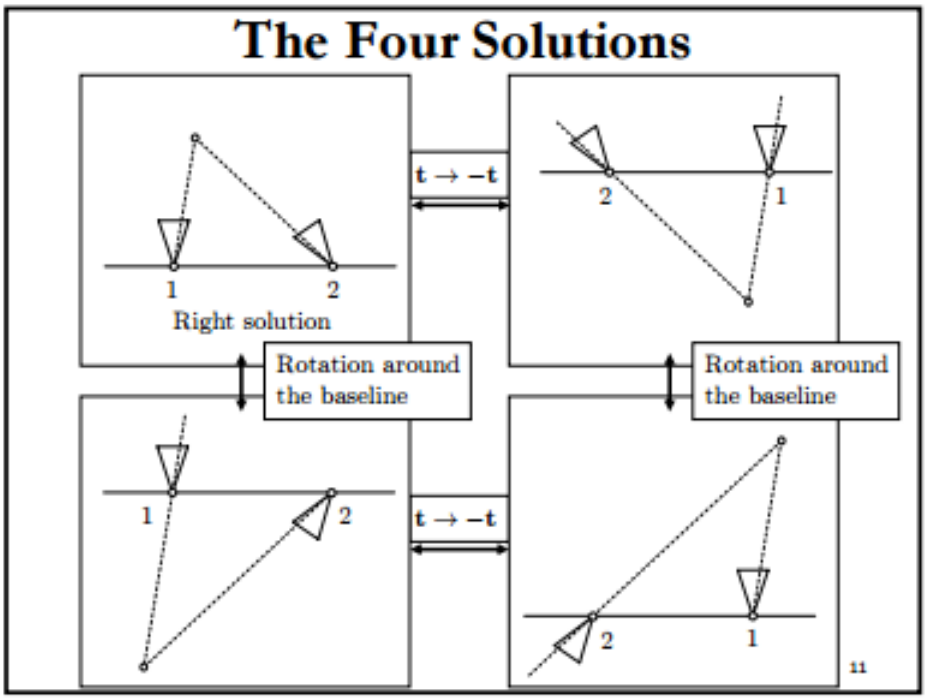


Figure 9: Four Solutions

src: http://isit.u-clermont1.fr/ ab/Classes/DIKU-3DCV2/Handouts/Lecture16.pdf

The t and -t swaps the position of the cameras. R_1 and R_2 makes a rotation of pi around the baseline. Therefore only one solution is feasible which is obtained via triangulation of a point and choosing the solution where the point is in front of both cameras. This condition is called cheirality constraint [4]. After estimating R and t, the trajectory is obtained by the equation:

$$t_{new} = t_{old} + scale.R_{new}t$$
$$R_{new} = R.R_{old}$$

- **Scale Ambiguity** There are multiple solutions for t as many possible decomposition of E and consecutively different set of U and Σ. Therefore scale ambiguity remains problem for a monocular system. We can use I.M.U. sensor data to resolve the problem.

6 Experiments and Results

6.1 Dataset

We are training our algorithm on images obtained from Ford Campus Vision and LIDAR Dataset [6] which were taken in an urban environment. The dataset was collected by an autonomous ground vehicle testbed, based upon a modified Ford F-250 pickup truck. The images were captured at the rate of 8fps so that we have sufficient overlap over images. The camera was mounted laterally on the car to get a wider environmental view.

6.2 Experimental Setup

The images obtained were adjusted for distortion. The images had a part of vehicle which resulted in considerable feature points detected on that part of vehicle. So we cropped the image to remove the unnecessary parts to ensure correct transformation matrix. This cropping doesn't effect the focal parameters but the principal point should be translated by the amount of cropping done.
As explained above, the images obtained were rotated 90° anticlockwise from actual straight view.(sample image to be attached). Since the axis of the vehicle doesn't align with the rotated camera we rotated the given image to align with the vehicle's axis. This resulted in changed intrinsic parameters which were computed by the following equations: If h and w are height and width of the images then for the rotated images principal point c_x,c_y for new rotated images are given by

$$x_{new} = h - y_{old}$$
$$y_{new} = x_{old}$$

6.3 Results

The ground truth for the dataset is:

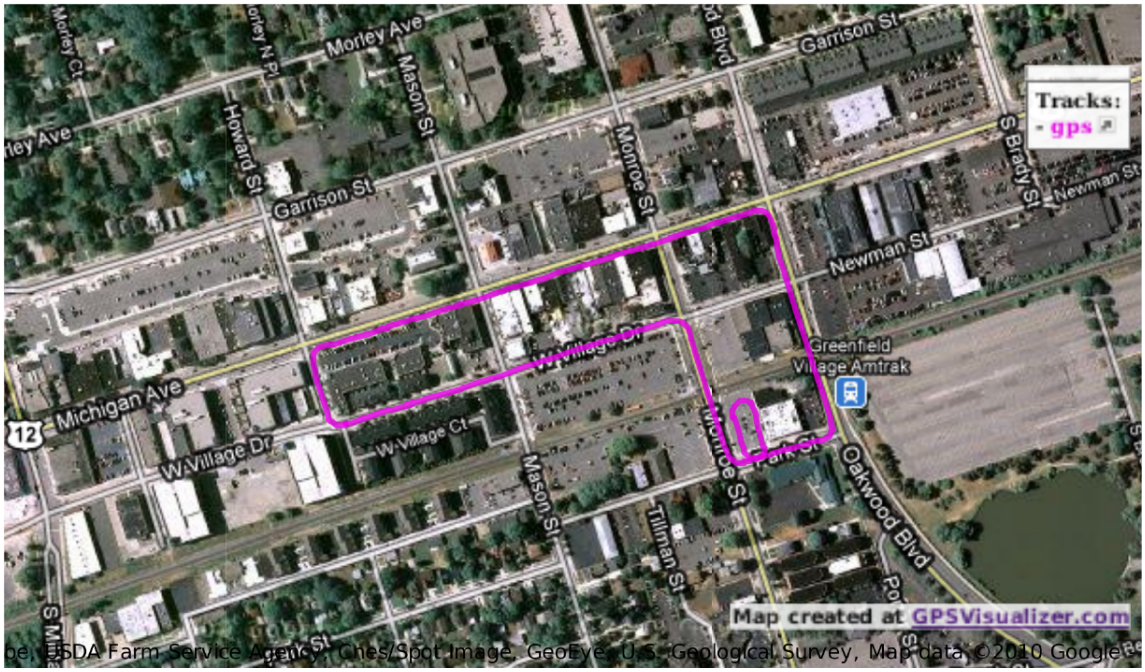


Figure 10: Ground truth of the trajectory

Tracking path using FAST feature detector and KLT tracker

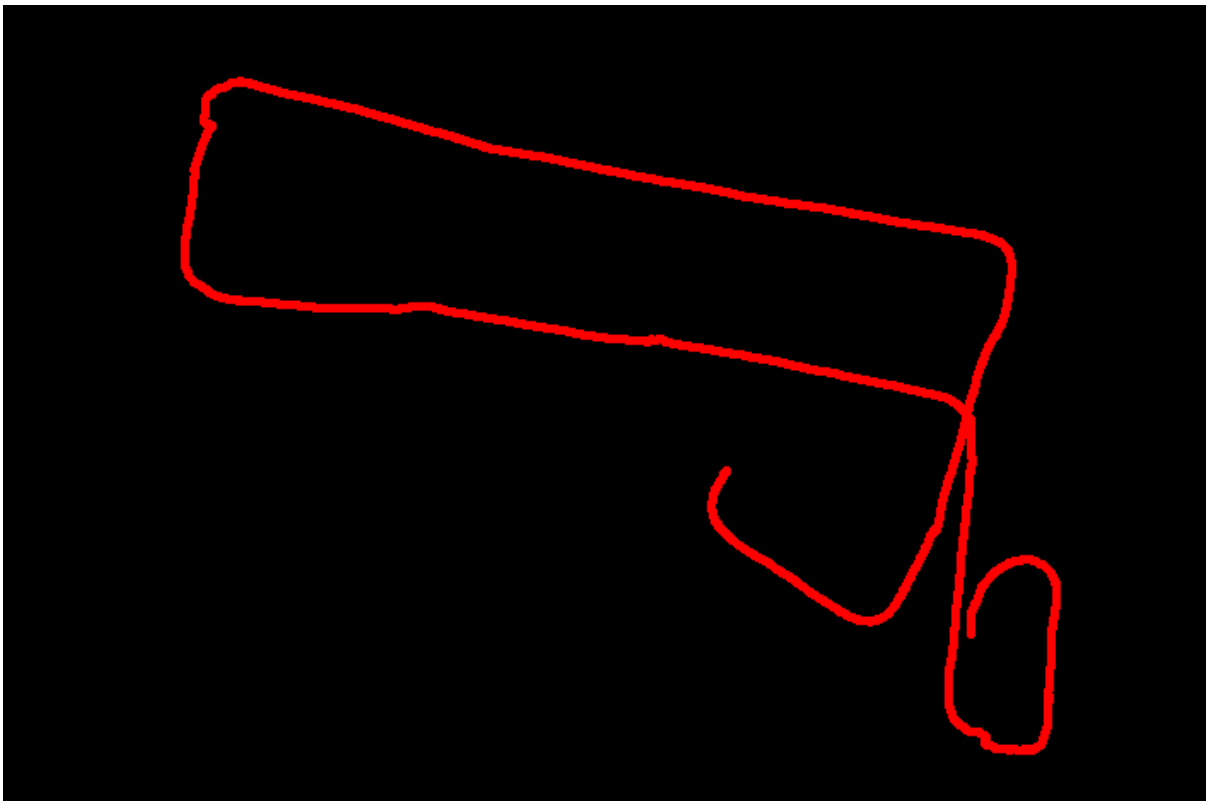


Figure 11: Estimated trajectory using FAST Detector

We also tracked using SIFT feature detector and KLT tracker:

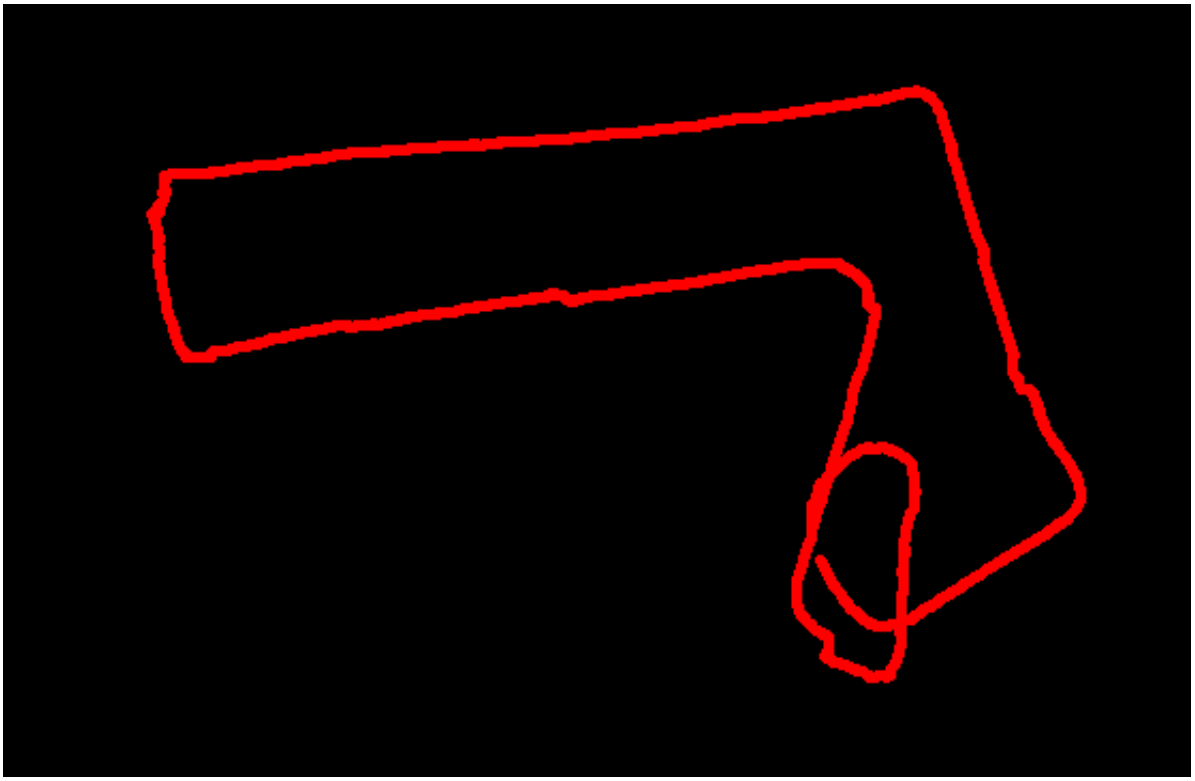


Figure 12: Estimated Trajectory using SIFT Feature Detector

The sampled points by FAST feature detector are :

Figure 13: Fast Detectors

Results of KLT tracker is below:

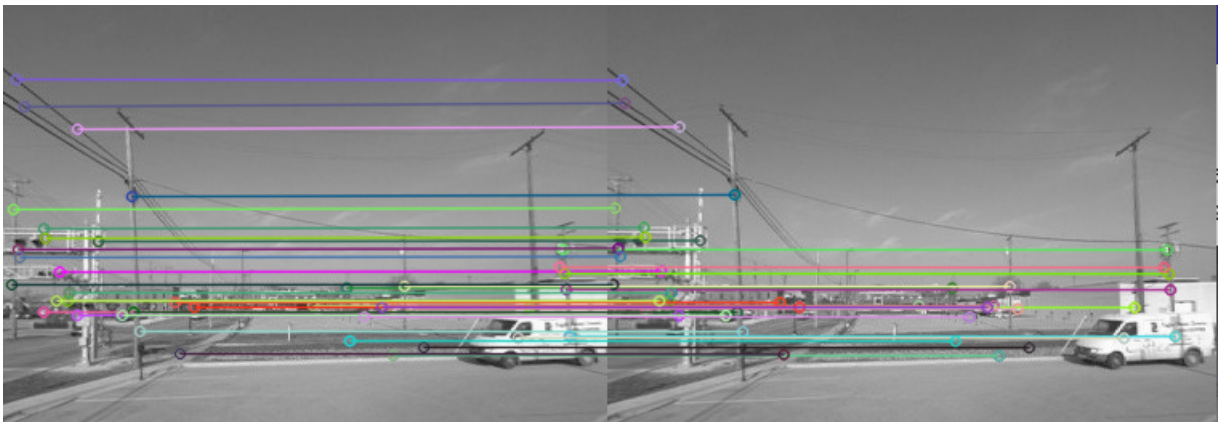


Figure 14: KLT Tracker

7 Discussion

We implemented the monocular visual odometry on Ford Campus Vision and LIDAR Dataset. We observed that when vehicle moves straight, our program generates straight path but it is tilted as shown above. This is due to the fact that we are calculating transformations relatively in visual odometry.This can be rectified using information from IMU sensors(not necessarily costly) or LIDAR sensors. The results using FAST were satisfactory. There is also a slight error in path prediction due to incorrect scale. This can be also be solved by IMU data or Bundle Adjustment. Results using sift are also satisfactory but SIFT took more time than FAST in feature detection step.

8 Conclusions and Future work

In general as tasks involving visual odometry needs to be ran in real time, we thus consider implementing FAST in our main work.

- **Resolving Scale Ambiguity**:We hope to resolve the scale ambiguity and tilt in the path using . We are trying to implement kalman filter which also takes other sensor data apart from camera images (I.M.U.) in our case.
- Another way of solving the scale ambiguity is to use Bundle Adjustment(BA) which finds scale by triangulating 3D points using previous frames and then re-projecting the points back to image to find scale factor for current timestamp. We would like to use this to resolve scale ambiguity as it helps in reducing the use of IMU's.

9 Acknowledgement

We thank Prof. Gaurav Pandey, Dept. of Electrical Engineering for his valuable support throughout the project guiding us from time to time and looking into the project when it was needed. We thank Prof. Subhajit Dutta Dept. of Mathematics and statistics for his suggestions. We also thank OpenCV community for their user friendly libraries.

References

[1] Epipolar Geometry. `http://www.cs.cmu.edu/~16385/lectures/Lecture18.pdf`.

[2] HighSpeed Test for Fast opencv documentation. `http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html`.

[3] RANSAC Tutorial. `http://image.ing.bth.se/ipl-bth/siamak.khatibi/AIPBTH13LP2/lectures/RANSAC-tutorial.pdf`.

[4] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision 60*, 2 (2004), 91–110.

[5] NISTÉR, D. An efficient solution to the five-point relative pose problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 6 (2004), 756–770.

[6] PANDEY, G., MCBRIDE, J. R., AND EUSTICE, R. M. Ford campus vision and lidar data set. *International Journal of Robotics Research 30*, 13 (2011), 1543–1552.

[7] ROSTEN, E., PORTER, R., AND DRUMMOND, T. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence 32* (2010), 105–119.

[8] SCARAMUZZA, D., AND FRAUNDORFER, F. Visual odometry [tutorial]. *Robotics & Automation Magazine, IEEE 18*, 4 (2011), 80–92.

[9] TOMASI, C., AND KANADE, T. *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ. Pittsburgh, 1991.