Visual Odometry for Vehicle Navigation

Indian Institute of Technology, Kanpur

Undergraduate Course Project

Siddharth Tanwar(13699)

Mentor - Prof. Gaurav Pandey

Abstract

Following is the documentation of visual odometry to estimate motion of a monocular camera. Camera information is used to predict the motion of car without any prior knowledge of surrounding and other sensor data. Harris corner Detection followed by feature matching and geometry estimation using 5-Point Nister Algorithm and RANSAC for outlier rejection is employed for trajectory estimation. The data obtained is further fused with IMU data to improve results. Implementation is tested on Ford Campus Vision and Lidar dataset. The implementation is in C++ using Opencv library and works in real time.

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 used by NASA in space rovers.

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.

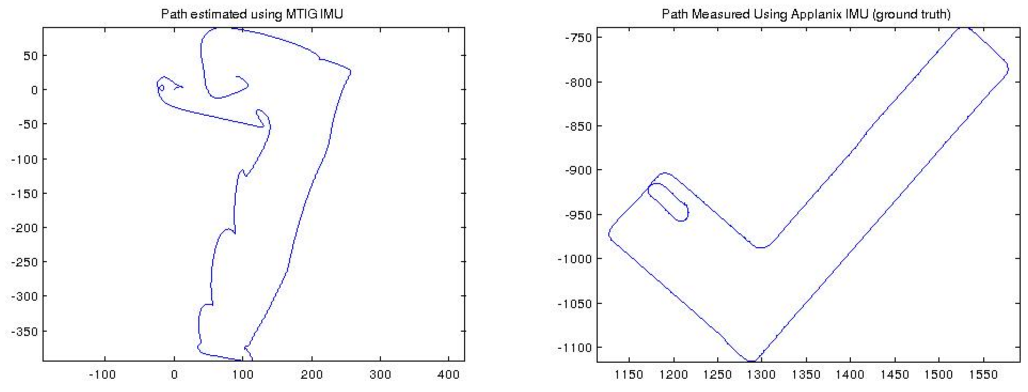

Figure 1: Normal I.M.U. and high cost I.M.U.

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.

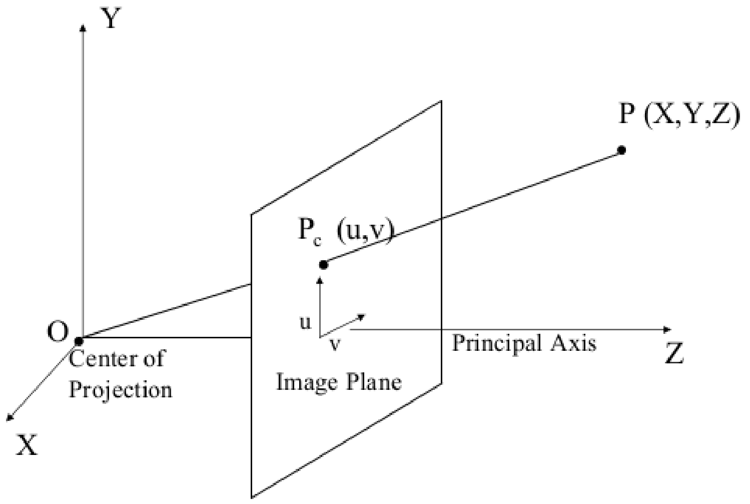

Figure 2: Image Projection Model

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.
- **Fusion with other sensors**- Fuse the obtained data with other sensors working in parallel to improve the accuracy of results and resolve scale ambiguity.

5.1 Feature Detection and Matching

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 robustness of the features. I employed Harris Corner Detection [3] because corners are relatively invariant to change of view and also given that we are using video feed consecutive images don't show much variance in view.
In each frame we detect Harris Corner Points. Harris corner points essentially represent textureness maxima, where textureness is measured using the shape of the autocorrelation function around a point. The autocorrelation function is defined by comparing the pixel values in a slightly shifted window to the pixel values of the original centered window. Instead of the conventional approach where all the points above a particular threshold are treated as corner points, we focus on windows throughout the image wherein the local maxima in a $n * n$ (n is user defined) is treated as an interest point.
We consider matching between two consecutive images only due to ease of implementation. While matching instead of comparing each point in the first image to every other point on the second image, we choose only the points within a certain disparity (30 percent of the image size in our case) for matching. The disparity depends on the surroundings. Among these points the one that has the maximum cross correlation is considered a match for image. Similar analysis is done the other way round. True match is only considered when mutual consistency condition holds ie. the points give the same result both ways [6].

5.2 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.

5.2.1 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].

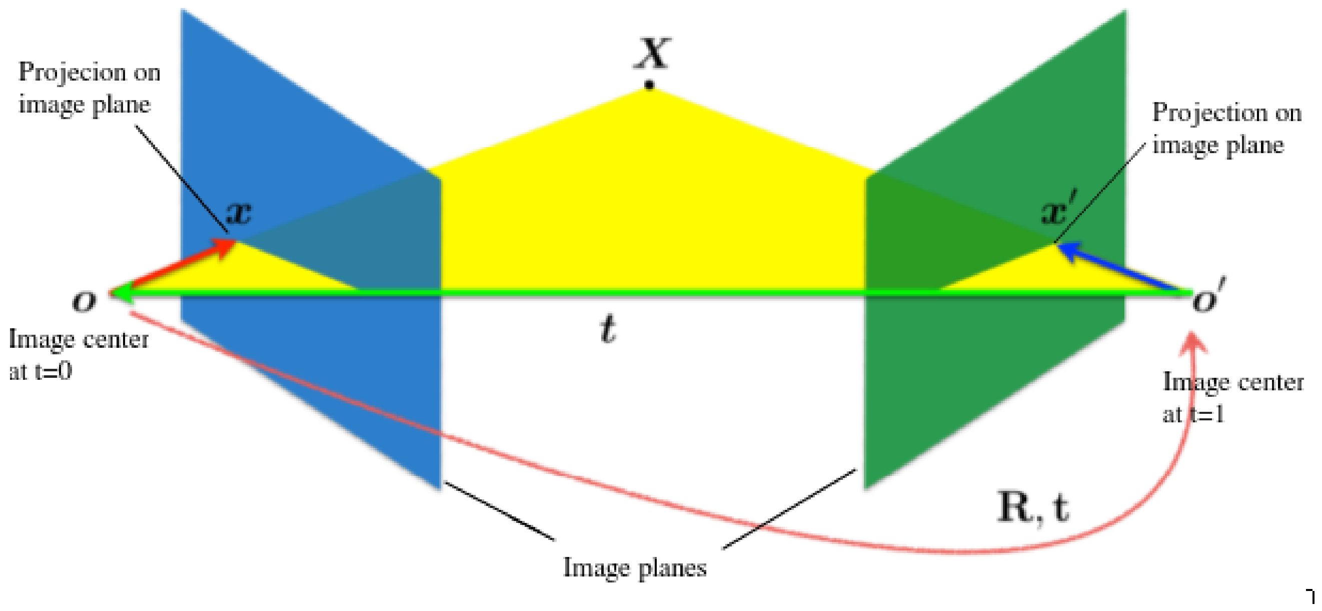

Figure 3: Epipolar Geometry (src : [1])

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

$$x' = R(x - t) \tag{2}$$

Using co-planarity condition we can write

$$(x - t)^T (x \times t) = 0 \tag{3}$$

Combining equation 6 and 7 we get

$$x'^T R(t \times x) = 0 \tag{4}$$

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

$$x'^T R(t \times x) = 0 \tag{6}$$

$$x'^T E x = 0 \tag{7}$$

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

5.2.2 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:

$$\widetilde{x}^T \widetilde{E} = 0 \tag{8}$$

where

$$\widetilde{x} = [x_1x_1^{'} \;\; x_2x_1^{'} \;\; x_3x_1^{'} \;\; x_1x_2^{'} \;\; x_2x_2^{'} \;\; x_3x_2^{'} \;\; x_1x_3^{'} \;\; x_2x_3^{'} \;\; x_3x_3^{'}]^T \tag{9}$$

$$\widetilde{E} = [E_{11} \;\; E_{12} \;\; E_{13} \;\; E_{21} \;\; E_{22} \;\; E_{23} \;\; E_{31} \;\; E_{32} \;\; E_{33}] \tag{10}$$

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

$$\widetilde{X}^T \widetilde{E} = 0 \tag{11}$$

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 $10^{t}h$ 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 [2], 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.

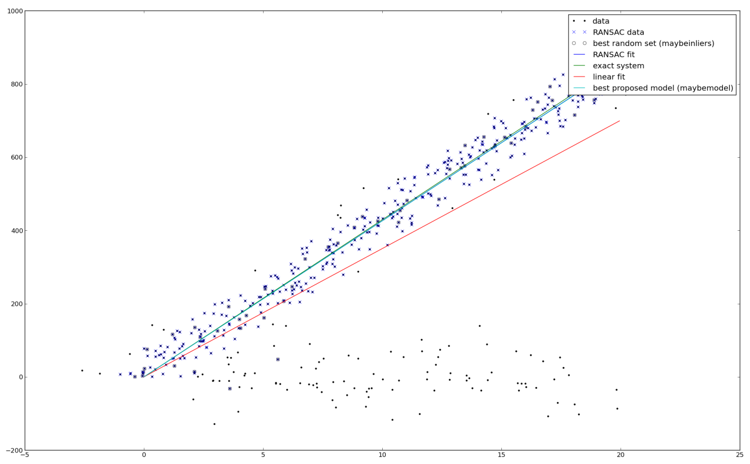

Figure 4: RANSAC Model

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

5.2.3 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

$$E = (\ U\Sigma V^T\ )$$

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$^T V^T$. 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

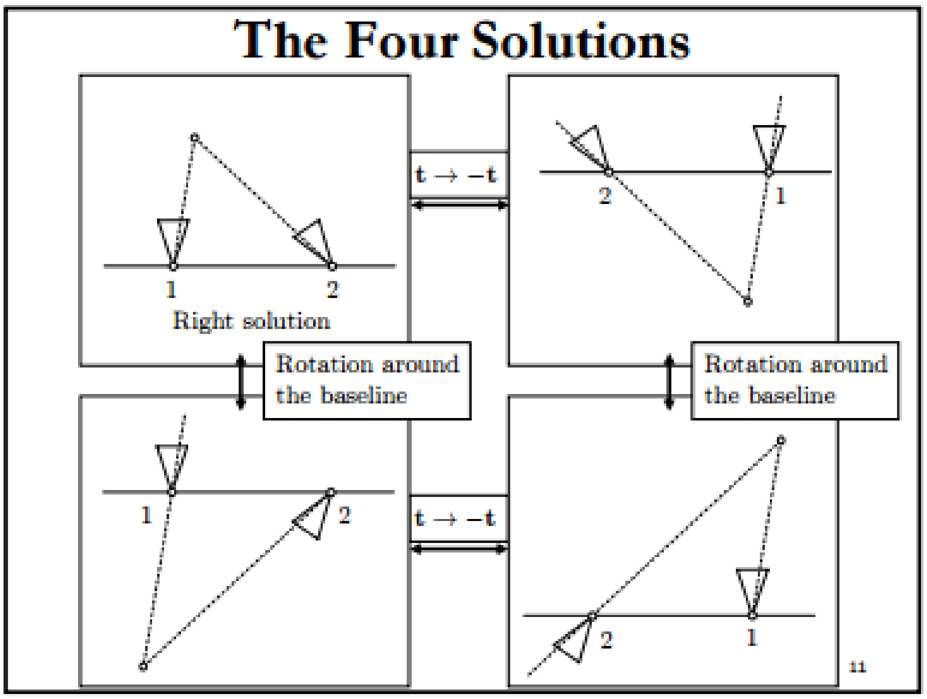

Figure 5: Four Solutions

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.

5.3 Sensory fusion

The data obtained from Visual Odometry was fused with data obtained from an IMU working in parallel but at a different rate than the VO data. Attempts were made to do so for two major purposes : to resolve scale ambiguity, to obtain more accurate results. An extended kalman filter was implemented to fuse the IMU and VO data. The filter assumed a constant velocity model and since the control signals were not known, they were assumed to be 0 and the following implementation followed.
The accelerations from the IMU were integrated prior to the Kalman filter to obtain the relative pose of the car whose 2D coordinates (x, y) were used in the Kalman filter. The path obtained from the VO is origin shifted, rotated and scaled when compared to that obtained from the IMU. EKF algorithm as follows :

Prediction Stage

$$X_t = g(X_{t-1}, u_t)$$
$$P_t = G_t P_{t-1} G_t^T + Q$$

Update Stage

$$K_t = P_t H_t^T (H_t P_t H_t^T + R)^{-1}$$
$$X_t = X_t + K_t(z_t - h(X_t))$$
$$P_t = (I - K_t H_t) P_t$$

System Model

$$X = \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \lambda_x \\ \lambda_y \\ b_x \\ b_y \\ \theta \end{pmatrix} \qquad z = \begin{pmatrix} x_{imu} \\ y_{imu} \\ x_{vo} \\ y_{vo} \end{pmatrix}$$

$$g(X_t) = \begin{pmatrix} x_t + \dot{x}_t \cdot dt \\ y_t + \dot{y}_t \cdot dt \\ \dot{x}_t \\ \dot{y}_t \\ \lambda_{x_t} \\ \lambda_{y_t} \\ b_{x_t} \\ b_{y_t} \\ \theta_t \end{pmatrix} \qquad h(X_t) = \begin{pmatrix} x_t \\ y_t \\ \lambda_{x_t}(x_t cos(\theta_t) - y_t sin(\theta_t) + b_{x_t} \\ \lambda_{y_t}(x_t sin(\theta_t) + y_t cos(\theta_t) + b_{y_t} \end{pmatrix}$$

In the above model X is the estimated state, $g()$ is the system model, $h()$ is the sensor model, z is the sensor readings, G and H are the jacobian matrices of g and h respectively, P is the prediction matrix (represents the variance in the estimated state), Q is the system noise variance matrix and R is the sensor noise. K is referred to as the gain matrix.
(x, y, θ) is the pose of the car in 2D coordinates, $(\dot{x}, \dot{y})$ is the velocity, (λ_x, λ_y) are the relative scalings in x and y direction respectively and (b_x, b_y) are the origin shifting biases in the two directions.

5.3.1 Implementation

Since the data is flowing from the two sensors at a different rate, a method to deal with this lack of synchronicity is important. The rate of incoming data from IMU is much faster than the camera and hence between two camera frames several IMU readings are obtained. So the implementation of our Kalman filter works at two levels. In the inner loop the states are predicted and updated based on solely the IMU data. As and when an image is encountered, the outer loop incorporates this data as well in the prediction and updation stages, thus combining the two sensor data.

6 Experiments and Results

6.1 Dataset

We are training our algorithm on images obtained from Ford Campus Vision and LIDAR Dataset [7] 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:

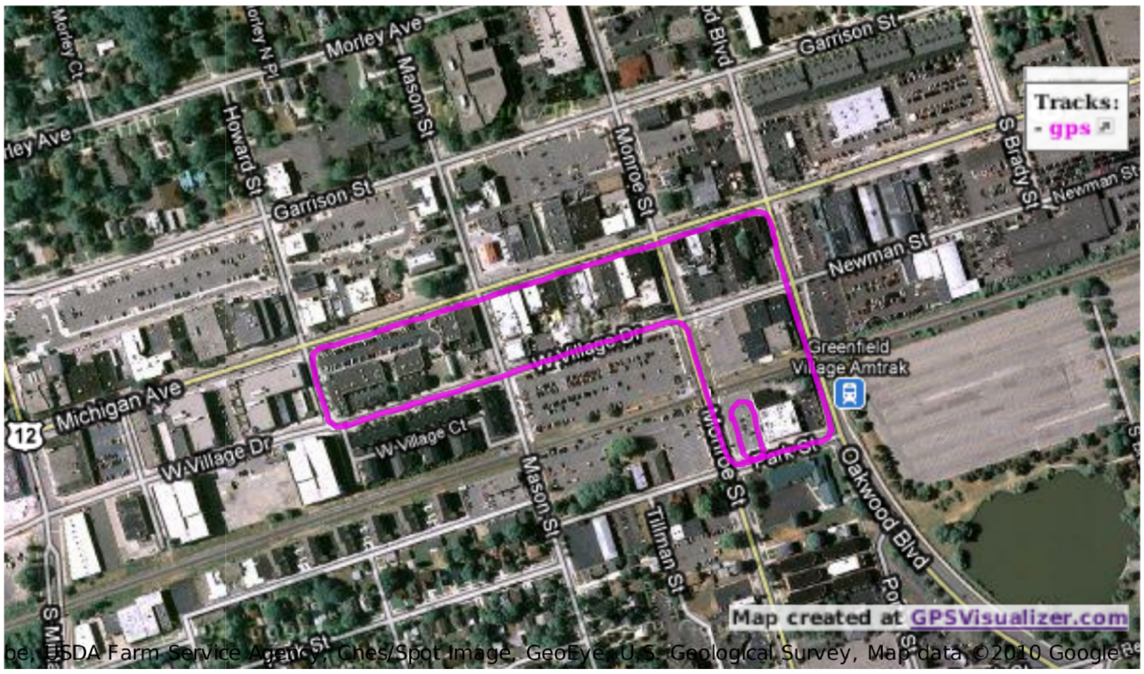

Figure 6: Ground truth of the trajectory

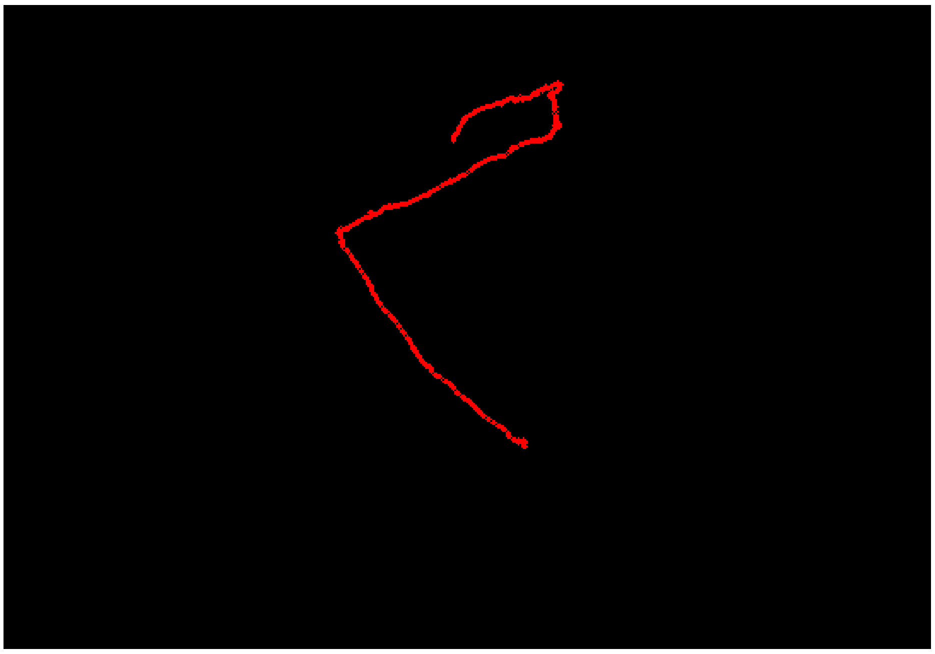

Figure 7: Estimated trajectory of a small portion of the path

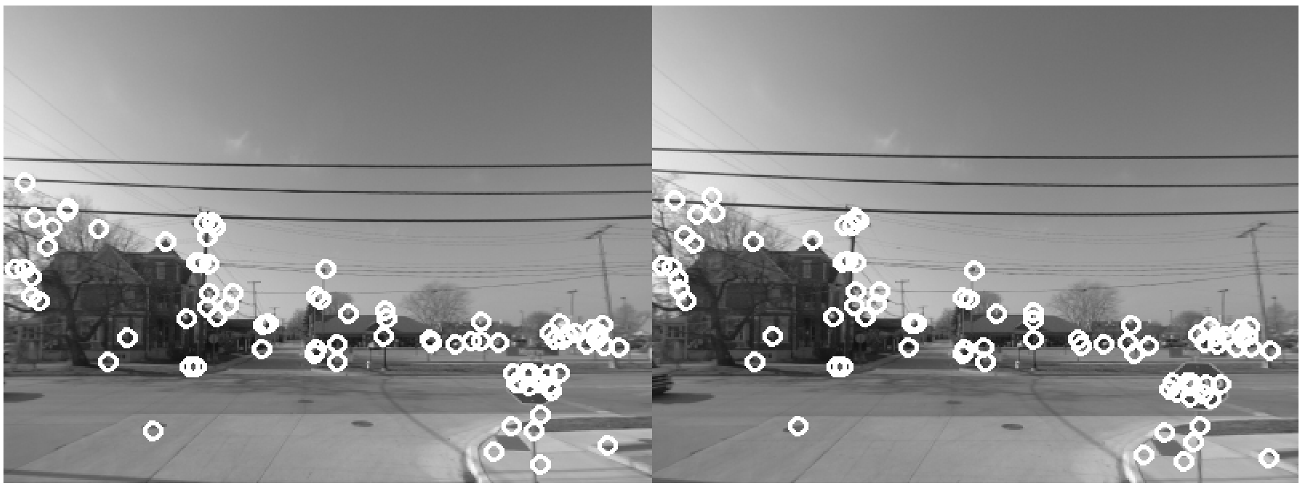

Figure 8: Sampled points and their matches between two consecutive frames

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. 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. Kalman filter didn't work as per expectation and further work is needed on it.

8 Conclusions and Future 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

I thank Prof. Gaurav Pandey, Dept. of Electrical Engineering for his valuable support throughout the project guiding me from time to time and looking into the project when it was needed. I also thank OpenCV community for their user friendly libraries.

References

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

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

[3] HARRIS, C., STEPHENS, AND M. A combined corner and edge detector proc. fourth alvey vision conference. pp. 147–151.

[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] NISTER, D., NARODITSKY, O., AND BERGEN, J. Visual odometry for ground vehicle applications. *Journal of Field Robotics 23*, 1 (2006).

[7] 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.

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