A

Project Report On

Trajectory (Motion) Estimation Of Autonomously Guided
Vehicle Using Visual Odometry

By

Ashish Kumar, Group -12, Roll No. 14104023
M.Tech, EE, (2014-1015), IIT Kanpur
Artificial Intelligence (CS365A)
Guide- Prof. Amitabha Mukharjee

Abstract: Visual odometry is a technique to
determine coordinates of a vehicle or any
object by using surrounding visual
information. The approach is revolutionary
for small autonomously guided vehicles.
Because in indoors performance is very poor.
And IMUs at small scale are very sensitive to
noise. And IMUs can also give unexpected
results which may lead to erroneous motion
estimation. Technique utilizes some state of
the art techniques like feature detection,
tracking and some mathematics of camera
theory.

I. INTRODUCTION

Odometry is a process of estimating
motion parameters using various kinds of
sensors like IMUs, GPS. When we do this using
visual information (Images), it is called visual
odometry. The term Visual Odometry was first
used by David D. Nister in 2004[1]. Which was
named so because it is similar to Wheel
odometry, in which we integrate the angle of the
wheels over time. In Visual Odometry we
integrate images over time. Problem with wheel
odometry is “wheel slip”. Which if not
incorporated in the estimation, can lead to
highly erroneous results.

Visual odometry was first used in
NASA’s Mars rover “Phoenix” and
“Opportunity”. And today a lot of research is
going on in this field.

So precisely in this particular
application of visual odometry, we estimate
trajectory covered by autonomous ground
vehicles (UGV-unmanned ground vehicles)[1],
we’ll be given some images captured from
camera mounted on the UGV and I need to
precess them through visual odometry system.
Visual odometry system involves a various step
process which in turn includes feature
detection, feature matching or tracking,
RANSAC, motion estimation and offline
adjustment. Odometry can be done by a
monocular camera or stereo camera. Stereo
camera often provides better results due to less
reconstruction errors involved in estimating
motion.

II. A BRIEF INTRODUCTION
TO WHAT I HAVE DONE

I Downloaded data set of 443 images
from Karlsruhe Institute of technology,
Chicago, which is technological institute of
TOYOTA in area of autonomous vehicles
research[6].

I went through the maths of the system
given in [1],[2],[3],[4],[5].Actually there is a
lot of maths involved in this project but it is
quite interesting. The maths is from two
different parts. One from images processing and
another from camera theory. Combining these
two gives birth to this beautiful project.

Then I wrote the whole program in
MATLAB to test the algorithm. Used Some
inbuilt functions of MATLAB like feature
detection, matching, because these are highly
optimized function.

After this I wrote the whole code in
Visual basic .NET for real time implementation
using EmguCV (.NET wrapper of
OpenCV )[7]. Then I Interfaced MATLAB with
Visual basic to plot trajectory graphs.

I have also implemented code using
feature matching and feature tracking. The later
is giving very good results.

III. PROBLEM FORMULATION

Aim: To estimate camera poses from set of
images taken at discrete interval
Indirectly: We have to find a Transormation
matrix which relates two image frames i.e. how
the two frames are rotated and translated from
each other.
Referring to [1], let a set of images be \( \{I_0, I_1, I_2, \ldots, I_{k-1}, I_k\} \), camera poses be \( \{C_0, C_1, C_2, \ldots, C_{k-1}, C_k\} \), and transformation matrix is given by
\[
T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}
\]
(1)

Here \( T_{k,k-1} \) is homogenous transformation matrix between images \( I_k \) and \( I_{k-1} \). \( R_{k,k-1} \) and \( t_{k,k-1} \) are rotation and translation matrix between images \( I_k \) and \( I_{k-1} \).

Let us suppose vehicle started \((0,0,0)\) of real world coordinate, then position of vehicle at time \( t_k \) is given by last column of matrix \( T_{k,k-1} \) whose last element is 1.

### IV. PREREQUISITES

- Camera Theory.
- Strong Knowledge of Camera intrinsic and extrinsic parameters.
- Spatial Transformation
- Image Basics
- Image Features

Above are some topics which are required you to gone through, because these terms will be used frequently in subsequent explanations.

For camera theory you can refer to [4], which is known as Bible of camera theory. And for Spatial transormations chapter 2 of [8] is enough.

Image Basics involves some basic knowledge and working with multidimensional matrices and RGB, Gray scale images. For this a lot of online tutorials are available.

For image features also, you can find best tutorials on internet. But the feature I have used will be described in a bit lesser detail.

As we move on I’ll keep on introducing new things and their details.

### V. ALGORITHM

- Feature detection (SIFT/SURF/FAST/Shi-Tomasi)
- Feature Matching / Tracking
- Outlier Removal using RANSAC
- Motion Estimation Using Essential Matrix
- Optional Windowed Bundle Adjustment [X]

Fig 2 shows poses of a stereo camera at different time instants. \( C_{k-1} \) and \( C_k \) are camera poses at time instants \( t_{k-1}, t_k \) respectively. For more information on transformation refer to chapter 2 of [4].

In the above figure, take \( O_1, O_r \) as position of camera at time \( t_1, t_2 \), which can be referred as image_1 and image_2. As you can see image_1 has been translated by a translation vector of ‘T’ and rotated by rotational matrix ‘R’.

‘R’ is a matrix of 3x3, ‘T’ is 3x1 vector which represents translation in X, Y, Z direction respectively and \( T_{k,k-1} \) 4x4 matrix to combine the effect of ‘R’ and ‘T’.

The camera pose at time \( t_k \) is given by
\[
C_k = T_{k,k-1} \ast C_{k-1}
\]
(2)

Fig 3

Feature detection (SIFT/SURF/FAST/Shi-Tomasi)
Feature Matching / Tracking
Outlier Removal using RANSAC
Motion Estimation Using Essential Matrix
Optional Windowed Bundle Adjustment [X]

Fig 3 Algorithm
**a. Feature detection**

A feature in an image nothing but only a point of interest. We apply mathematical operations on those points only rather whole image.

A lot of feature detection algorithms are available. We can choose any of them but for this particular application I have tested SIFT, FAST, FREAK, Harris, SURF, Shi-Tomasi.

Motion estimation using SIFT, SURF, FREAK takes a lot of computational time. Whereas Shi-Tomasi corners are computed in lesser time and for motion estimation these doesn’t eat much time.

I used SIFT, SURF, FREAK for feature matching and Shi-Tomasi, Harris for feature tracking.

Feature tracking provided very good results then feature matching with drastic improve in computation time.

**Functions used (Objects in language):**

MATLAB:
1. detectSIFTFeatures
2. detectSURFFeatures
3. detectFASTFeatures

OpenCV:
1. SIFTDetector
2. SURFDetector
3. GFTTDetector

For more reference documentation of EmguCV [7].

**b. (i) Feature Matching**

Suppose you have detected features 50 and 60 features in two consecutive images im1 and im2 respectively. Feature matching means finding an approximate match of 50 features in im1 to 60 features in im2.

Precisely we want to know where these 50 features of image-1 are in image-2 (Camera-1 ,Camera-3). These matches can be found using radial match or k-nearest neighbor match. I have used K-NN.

In K-NN we find distance of each of 50 features of image-1 from 60 feature of image-2. Then we only retain the points (matches) which are having distance between them below a threshold. Threshold is set heuristically.

For feature matching we need additional functions. But these are not required in feature tracking.

**Functions used (Objects in language):**

MATLAB:
1. extractFeatures
2. matchFeatures

OpenCV:
1. ComputeDescriptorsRaw
2. knnMatch (FLANN class OpenCV)

**b. (ii) Feature Tracking**

Difference between feature matching and feature tracking is as follows. Feature matching means independently finding features in two frames and matching them. Whereas feature tracking means finding features in one image only and tracking them in subsequent images[2]. It may happen that some feature may get lost due to occlusion or out of field of view of camera. So you have to add some new best strong points. I have used feature tracking only in OpenCV due to time constraint.

For tracking first we detect best feature called Shi-Tomasi features or GoodFeaturesToTrack[10] and then use sparse version of Kannade-Lucas-Tomasi tracker using optical flow. For KLT refer to the original implementation [9].

**Functions used (Objects in language):**

OpenCV:
1. GFTTDetector
2. cvCalcOpticalFlowPyrLK

**c. Outlier Removal using RANSAC**

From feature matching (tracking depending what you have used), we have obtained point correspondences (matches). Some of the matches even after thresholding may be wrong. The wrong matches are called ‘Outliers’ and correct ones are called ‘Inliers’.

The outliers can greatly diminish performance of Visual Odometry system. So before motion estimation we have to remove these outliers. Popular method is to use RANSAC (Random Sampling And Consensus).

As referred by [1], In brief RANSAC works like this. Take any two correspondents from the matches and fit a line between them. Compute distance of remaining point from this line. Save the points which are having distance below a threshold. These are inliers. Keep on repeating this method to obtain as many inliers as you want.

RANSAC is a non-deterministic technique to obtain inliers. No of loops required is given by

\[ N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} \]  

(3)

\( p \) = probability of success
\( \varepsilon \) = % of outliers in data
\( s \) = no of points by which model can be instantiated
So after RANSAC we only have inliers.

Let Point correspondents are denoted by $x, x'$ in 1st and 2nd image respectively. $x, x'$ are 3x1 homogenous vector i.e. 3rd element is equal to 1.

**d. Motion Estimation**

Before we proceed to motion estimation, it is required to get familiar with some of camera concepts like its intrinsic and extrinsic parameters. For Maths involved in Motion estimation we’ll refer to [4]. Most of maths is from Part II of [4].

Intrinsic Parameters of camera is the calibration matrix of camera which give you the perspective projection. It is a 3x3 matrix denoted by ‘$K$’.

$$K = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$  (4)

$\alpha_u, \alpha_v$ are the scale factor in x,y direction due to perspective projection. $u_0, v_0$ are center of image.

Next I’ll describe about various frames of references i.e. world coordinate frame, camera frame.

The Image Frame may be rotated w.r.t. world coordinate system (C). Let the rotation be ‘R’ and translation be ‘t’. then a matrix $P=[R|t]$ is called extrinsic parameters matrix. So let ‘X’ be a point in image plane w.r.t frame of ref (I) and ‘X’ be a point w.r.t. world coordinate system then

$$x = KPX$$  (5)

$x$ is 3x1 homogenous vector, $K$ is 3x3 calibration matrix. $P$ is 3x4 matrix and $X$ is 4x1 homogenous vector.

So from here you can understand that we have to estimate ‘R’ and ‘t’ w.r.t. a frame of reference given by

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$  (6)

Refer to [4] for more detail on this.
(i) Estimating ‘R’ and ‘t’

There is something called essential matrix which is combination of ‘R’ and ‘t’. and is given by

\[
E = \begin{bmatrix}
0 & -t_z & t_y \\
t_z & 0 & -t_x \\
-t_y & t_x & 0
\end{bmatrix}
\]

(7)

So now you may have understood the point that we have to estimate this essential matrix and then break into ‘R’ and ‘t’.

To estimate ‘E’ matrix we have a lot of algorithms available. For example Normalized 8 point algorithm, 7 point algorithm, Nister’s 5 point algorithm, RANSAC-1 point algorithm.

A condition of epipolar geometry is satisfied by this ‘E’ matrix. Which is called epipolar constraint and is given by

\[
\text{From RANSAC output we have } x, x'.
\]

\[
\text{We put } x, x' \text{ in above equation and estimate } E.
\]

\[
\text{You might be thinking that I have points to solve the equation, then why I am saying that “we are estimating ‘E’”? This is because RANSAC output is non deterministic. So we have only an estimate of correct correspondences.}
\]

I have used Normalized 8 point algorithm.

Functions used (Objects in language):
MATLAB:
1. estimateFundamentalMatrix
OpenCV
2. cvFindFundamentalMat

Let’s imagine that first camera pose is denoted by P and second by P’. Where ‘P’ is of the form given by (6) and P’ = [R|t], where ‘R’ and ‘t’ are axis rotation and translation vector by which camera-2 has been transformed w.r.t. camera-1.

Aim is to estimate ‘R’ and ‘t’ from ‘E’ matrix. So for this we take SVD of ‘E’ matrix. There is special method to take SVD of essential matrix. For which you can refer to appendix B refer to [3].

As given in [1]. Let SVD of ‘E’ matrix is given by UDVᵀ and a matrix ‘W’ is given by then we get four solutions given by

\[
W^T = \begin{bmatrix}
0 & \pm 1 & 0 \\
\mp 1 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

(9)

Out of these four solutions only one corresponds to true configuration. And these four solutions corresponds to configurations in space as given below

Now we have to select only one correct configuration out of these four. To do this we use a variant of direct linear transform as given in chapter 12 of [4] on page no 312.

DLT is used to triangulate a point. Means finding x,y,z of an image point in world coordinate system. After triangulation we put a constraint on the triangulated point that it must be in front of both of the cameras. This is chiral constraint given in [3].

Since this constraint will be fulfilled by only one configuration , so now we have ‘R’ and ‘t’ which relates camera-1 and camera-2. The process is repeated for subsequent images and at any time we can tell X, Y, Z coordinate of the vehicle w.r.t. the point from it started.

Now we concatenate all the \( T_{k,k-1} \) matrices to obtain position on \( N^{th} \) frame w.r.t. the point from where the vehicle started. So position of \( N^{th} \) frame is w.r.t. starting point is given by

\[
0 = \begin{bmatrix} 0 & 1 & 2 & \cdots & N^{-2} & N^{-1} \end{bmatrix}
\]

(11)

\[
N^{-1}T = n^{-1}T^{-1}
\]

(12)

So let us take starting point be (0,0,0). And ‘R’ and ‘t’ be the correct configuration of camera-2 w.r.t. camera-1 . Then location of camera-2 w.r.t. camera-1 is given by

\[
X = [R|t]^{-1} [0,0,0,1]^T
\]

‘X’ is (X,Y,Z) of camera-2 w.r.t frame of reference of camera-1
b. Optional Bundle Adjustment

As name suggests it is optional to use. Although I haven’t used it but I’ll explain in short what it is. For more details on it please refer to [2].

Bundle means set of transformation matrices. We take previous ‘m’ transformation matrix to find next transformation matrix. This eliminates disparity problem if frames are too close.

As described in [2], in fig 9 uncertainty of pose $C_k$ is combination of uncertainty in $C_{k-1}$ and $T_{k,k-1}$.

And we are done. Now it’s time to have a look on my program’s results. I used DATA SET from KITTI [7], Karlsruhe Institute of Technology, Chicago. Which has color as well as grayscale images in separate folders. They also have provided ground truth data (IMU data, GPS data), calibration matrix. Ground truth is used to compare results. I’ll show my results both using feature matching and feature tracking.

**Results**

Car Used In Capturing Images

From fig 10 you can see that according to camera (Red) forward direction is ‘Z’ axis, left is ‘X’ and ‘Y’ is pointing downwards. And for IMU (Green) ‘X’ is pointing forward, ‘Y’ is leftward and ‘Z’ is pointing upward. So according to ground truth ‘X’ components will be the direction of motion.
Some SnapShots of my .NET application

Fig 12. Feature matching Technique using FAST detection and SURF extraction on DATA SET images

Fig 13. Feature Tracking Technique using Shi-Tomasi features detection and Sparse version of Kannade-Lucas-Tomasi Tracking on DATA SET images

Fig 14. Feature Tracking Technique using Shi-Tomasi features detection and Sparse version of Kannade-Lucas-Tomasi Tracking on DATA SET images
Fig 15. Feature Tracking Technique using Shi-Tomasi features detection and Sparse version of Kannade-Lucas-Tomasi Tracking on Live camera

Ground truth

Fig 16  Ground Truth provided by IMU and GPS mounted on the Car

Ax,Vx - Acceleration, velocity in x direction.
Ay,Vy - Acceleration, velocity in y direction.
Az,Vz - Acceleration, velocity in z direction.
X    - X coordinate of Vehicle
Y    - Y coordinate of Vehicle
Z    - Z coordinate of Vehicle
Difference between the results of OpenCV for matching and tracking is the part where car stops for a couple of seconds. In feature matching, I was getting a non-zero translation even the car was stopped. But Using feature tracking I got a nearly zero translation when car was stopped. You can notice the difference in the results of OpenCV by the arrow tip. In feature matching results, you can see some pixels outside of trajectory. Those are wrong results. But in Feature tracking this is removed.

Data Set
1. Karlsruhe institute of Technology, Chicago (Technogical research institute of TOYOTA for Autonoumous vehicles)
2. Raw 443 unrectified gray scale images of size 1392 x 512 of .png format. 3. Images are captured in City.

Softwares Used
1. MATLAB 2013, MathWorks.
3. EmguCV, a .NET wrapper of OpenCV Binaries.

References:
[3]. David Nister, Member, IEEE, “An Efficient Solution to the Five-Point Relative Pose Problem”, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE.VOL. 26, NO. 6, JUNE 2004
[4]. Multiple View Geometry in Computer Vision 2nd Edition by Richard Hartley Australian National University, Canberra, Australia and Andrew Zisserman University of Oxford, UK
[5]. H.C. Longuet, Higgins “A computer algorithm for reconstructing a scene from two projections”.
[9]. https://www.ces.clemson.edu/~stb/klt/
[10]. Jianbo Shi, Carlo Tomasi, “Good Features to track” in IEEE international conference on computer vision and. pattern recognition, Seattle, June 1994