Implementation of Stereo Visual Odometry Estimation for Ground Vehicles

Pavan Kumar U*, Sahul M.P.V** and B.T Venkatesh Murthy***
*Electronics & Communication Department, Siddaganga Institute of Technology
pavankumaru93@gmail.com
**Data Handling and Storage group, ISRO Satellite centre(ISAC)
sahul@isac.gov.in
***Electronics & Communication Department, Siddaganga Institute of Technology
btv_murthy@rediffmail.com

Abstract: Keeping track of vehicles location is a challenging aspect in recent days. Visual Odometry (VO) is the process of estimating the orientation and position of a vehicle or a robot by analyzing the associated camera's input. In this paper a stereo camera based visual odometry system is presented in which stereo cameras have been rigidly attached to the vehicle and motion of the vehicle is estimated using only the input coming from these stereo cameras. No other sensors are needed. The corner features are extracted in both left and right images using Harris corner detector algorithm, descriptor for each feature is extracted using Scale Invariant Feature Transform (SIFT) algorithm, feature descriptors are matched between left and right images using K-nearest neighbor match. Those matched feature correspondences are triangulated to get 3-D points. Two sets of 3-D points are obtained at time steps 't' and 't+1'. Then, the motion is estimated using least squares fitting of these two 3-D point sets.

Keywords: Visual odometry, SIFT descriptors, feature matching, KITTI dataset, stereo vision, triangulation.

Introduction

Visual Odometry (VO) is a process in which vehicle's motion is estimated using only the input of stereo cameras attached to the vehicle. Goal of VO is to determine the global orientation and position of the vehicle at every time instant. VO works by finding interest points in the images and matching them between the left and right images at every time instant. Matched interest points are triangulated to obtain 3-D points. Robust methods are then used to estimate the camera motion from these 3-D points. VO can be applied in the field of automotive, wearable computing, robotics and augmented reality. [1]

The term 'Visual Odometry' was selected for its resemblance to wheel odometry. In wheel odometry the trajectory of a vehicle is estimated by computing the number of turns of its wheels over time. Similarly, In VO the position of the vehicle is estimated with the changes that motion induces on the images obtained from the stereo cameras which is attached on the vehicle. For VO to work in an efficient manner, there must be enough amount of light illumination in the environment and cameras should capture consecutive frames ensuring that there is enough amount of scene overlap. VO has most importance at environments like aerial and underwater.

Literature survey

Visual Odometry (VO) for estimating the orientation and position of a vehicle or a robot has demonstrated some important breakthroughs over the last decade. VO is considered as a sub problem of visual simultaneous localization and mapping (SLAM) problem in robotics. Huge amount of visual odometry algorithms have been developed using monocular, stereo and multi-camera configurations.

NASA's two rovers spirit and opportunity used for mars exploration were equipped with VO system to estimate the onboard position and attitude (role, pitch and yaw) of the rovers using stereo cameras [14].

As line segments are less sensitive to lighting variations and abundant, a line segment based indoor VO system using RGB-D cameras have been developed in [4], and is robust to lighting variations.

Autonomous Underwater Vehicles (AUV) are effective when working in industrial field but are still in development state. An image processing based VO system has been implemented in [6], to estimate the AUV's ego motion and changes in the orientation. The algorithm was deployed on Raspberry pi2.

Feature matching is a difficult step in the computation of VO and many other computer vision applications. A new feature descriptor called Synthetic Basis (SYBA) descriptor have been developed in [9] along with VO, to obtain more precised feature matching results and reduce the VO computation errors.

Proposed Methodology

Stereo Vision

Stereo vision uses two or more cameras, but the condition is that cameras must be coplanar and parallel to each other. If the condition is not satisfied rectification is done. An object in 3-D space can be found using stereo vision[3].

An example for stereo vision is the human visual system. Each person has two eyes that see two slightly different views of the observer's environment. An object seen by the right eye is in a slightly different position in the observer's field of view than an object seen by the left eye.

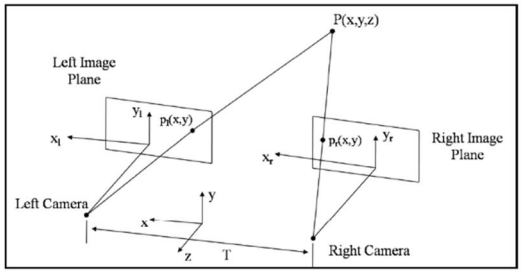

Figure1. The geometry of stereo vision

Fig.1,[3] shows the geometry of stereo system with two pinhole cameras. The right and left image planes are parallel to each other and coplanar, and represented as IR and IL respectively, Pr (x, y) and Pl (x, y) are the 2-D feature points on right and left image planes respectively and P (x, y, z) is the triangulated 3-D point in space.

Feature detection

A feature is defined as an interest point in an image. The first step in VO is to detect the features in the images captured by the stereo cameras. For VO, point features like corners and blobs are important because, one can measure the position of the corner or a blob in an image accurately. A corner is a point where two or more edges intersect. A blob is a pattern that differs from its immediate neighborhood in terms of color, texture and intensity. Corner detectors are less distinctive but are fast to compute, whereas blob detectors are slower to compute but are more distinctive. Therefore, for VO corners are best suited.

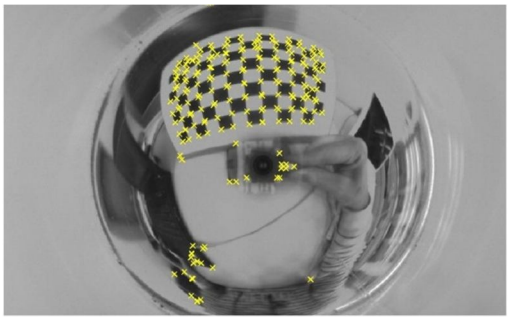

Figure2. Harris corner detector

A popular and commonly used feature detection algorithm called Harris corner detector algorithm is used to detect the corner features, the basic idea (Fig.3)[12] behind this algorithm is that if a pixel point is in a flat area, there will be no variation in the gray scale between the center pixel and the pixels surrounding to it in any direction. If the pixel point is in an edge area, there will be no variation in the gray scale between the center pixel and the pixels surrounding to it in the edge direction. If a pixel point is a corner, then there will be noticeable variation in the gray scale between the center pixel and the pixels surrounding to it when the window move in any direction [12].

Harris corner algorithm represented by equation (1) is used to determine if a pixel point is a corner or an edge. The features extracted by Harris algorithm are affine invariant and rotation invariant, but are not scale invariant. Harris equation is given by,

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \quad (1)$$

Where, w (x, y) is the window function and I (x, y) is the intensity value of the centered pixel.

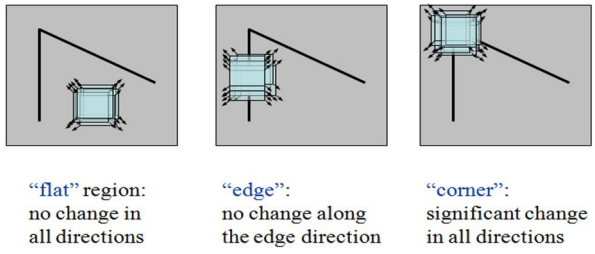

Figure3. The basic idea of Harris corner detector

If the centered pixel is in a flat region or an edge region the term "$[I(x+u, y+v)-I(x,y)]^2$" in (1) will be almost equal to zero, where as if the pixel is in a corner region this will be larger, Therefore when the value of E(u,v) is large, the pixel is considered as a corner.

Descriptor extraction

In feature description, the pixels surrounding each corner feature point is converted into a compact descriptor, so that it can be matched against the feature descriptors in the other image. The most popular and commonly used descriptors for corner features are the SIFT (Scale Invariant Feature Transform) descriptors [13].

Basically, the SIFT descriptor is a histogram for local gradient orientations. The region surrounding each feature point is decomposed into 4x4 blocks. For each block, a histogram of eight orientations weighted by magnitude and Gaussian window is built. All these histograms of 4x4 blocks are concatenated to form 128 element vector. The descriptor vector is then normalized to unit length, to reduce the effect of light illumination changes.

The magnitude and orientations for each quadrant are given by,

$$M(x,y) = \sqrt{\left(\left(L(x+1,y) - L(x-1,y)\right)^2 + \left(L(x,y+1) - L(x,y-1)\right)^2\right)} \quad (2)$$

and

$$\boldsymbol{\theta(x,y)} = \boldsymbol{tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)} \quad \boldsymbol{(3)}$$

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \quad (4)$$

Where, I(x,y) is the input image and G(x,y,σ) is the Gaussian function with scale (σ).

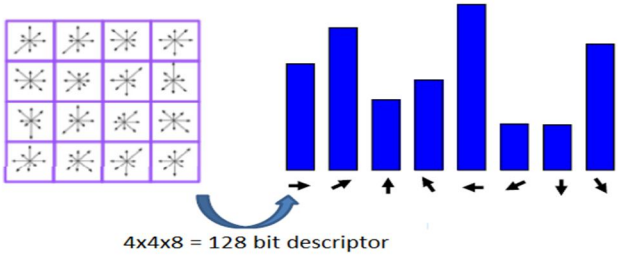

Figure4. The idea of SIFT descriptor

Feature Matching

In feature matching, the features in left image should be matched to the corresponding features in the right image. The simplest way to match feature points between any two images is to compare all the feature descriptors in one image with all other feature descriptors in the other image. The descriptors are compared using a similarity measure. For SIFT descriptors, this is Euclidean difference between two descriptors.

After comparing all the feature descriptors between two images, the best correspondence of a feature in the second image is chosen as that with the closest descriptor i.e. the correspondence with least Euclidean difference.

The Euclidean difference is given by,

$$D = \sqrt{\sum_{i=1}^{n} \left(x_i^L - x_i^R\right)^2} \tag{5}$$

Where, x^L is the descriptors of left image and x^R is the descriptors of right image.

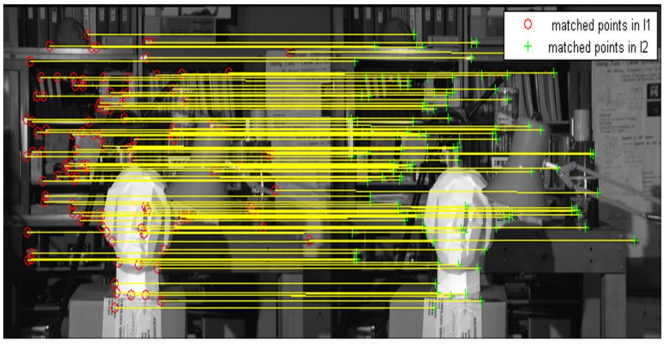

Figure5. Matched features between left and right images

Outlier Removal

From feature matching we have obtained feature point matches (correspondences) between left and right images. Some of the matches even after thresholding may be wrong. The wrong matches are considered as 'Outliers' and the correct ones as 'Inliers'. The causes for mismatches are blur, occlusions, sudden changes in the view point and image noise. Outliers must be removed, for trajectory to be estimated accurately.

RANSAC (RANdom SAmple Consenous) is commonly used and popular algorithm to remove outliers. The basic idea behind RANSAC is to take any two correspondences and fit a line between them. Then compute distance of remaining points from this line. Consider the points which are having distance below a threshold as inliers. Repeat this step as many number of times to obtain more number of inliers. [2]

The number of iterations N required to guarantee that a correct solution is found can be computed as,

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)} \tag{6}$$

Where, p is the probability of success, ϵ is the percentage of outliers in the data and S is the number of data points.

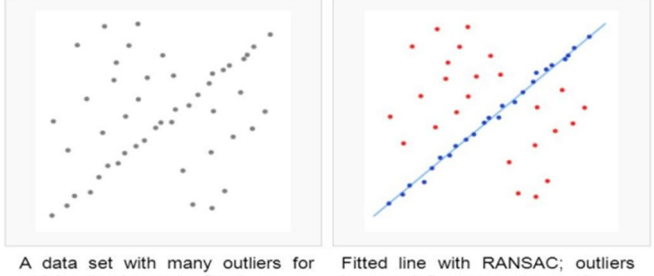

Figure6. RANSAC example

Triangulation

The method in which stereo determines the position in space of p and q in Fig.7(a) [3] is triangulation, i.e. The position of the 3-D points which are triangulated are obtained by intersecting the rays back projected from 2-D image correspondences of at least two image frames.[3].

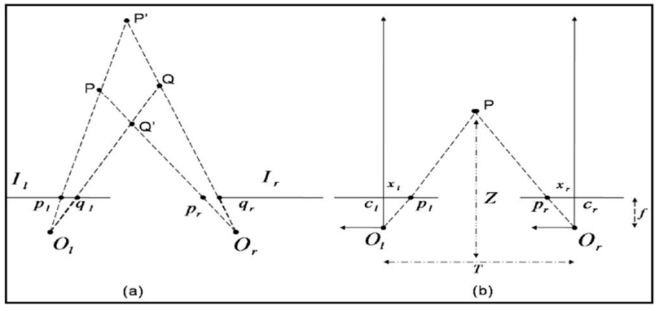

Figure7. A simple stereo system

From Fig.7(b) [3], P (X, Y, Z) is the position of single 3-D point from its projections, pl (xl, yl) and pr (xr, yr). T is the distance between the centres of projection Ol and Or, which is the baseline of the stereo system. F is the common focal length for both left and right cameras, and Z is the distance between 3-D Point P and the baseline T. From similar triangles (pl, P, pr) and (ol, P, or),

$$\frac{(T + xl) - xr}{Z - F} = \frac{T}{Z} \tag{7}$$

$$Z = F\frac{T}{d},\ X = Z\frac{xl}{F},\ Y = Z\frac{yl}{F} \tag{8}$$

Where, *d = (xl - xr)* is the disparity, which measures the difference in retinal position between the two matched corresponding feature points in the two images.

Motion Estimation

Once the sets of corresponding matched image features are obtained, these are projected to a 3-D space by triangulation between the right and left image frames. Then, with two 3-D feature points sets, the relative motion between the previous and current time steps can be estimated using least-squares fitting algorithm.

The motion is related to rotation and translation that will align the two 3-d point sets at time instants 't' and 't+1'. For the case $n \geq 2$ correspondences, one possible solution is to compute the rotation part using SVD (singular value decomposition) and the translation part as the difference between the centroids of two sets of 3-D feature points.

Let P_a and P_b be the two 3-D points datasets at time steps 't' and 't+1' respectively. Then the centroids are just the average mean point and can be calculated as follows,

$$centroid(p_a) = \frac{1}{N}\sum_{i=1}^{N} P_a^i \quad \& \quad centroid(p_b) = \frac{1}{N}\sum_{i=1}^{N} P_b^i \tag{9}$$

Where, N is the number of 3-D feature points at particular time step.

Rotation is obtained by accumulating cross covariance matrix(H) and applying SVD to it.

$$H = \sum_{i=1}^{N}\left(P_a^i - centroid(P_a^i)\right) * \left(P_b^i - centroid\left(P_b^i\right)\right)^T \tag{10}$$

$$[U, S, V] = SVD(H) \tag{11}$$

$$R = VU^T \tag{12}$$

Translation is given by,

$$t = -R * centroid(P_a^i) + centroid\left(P_b^i\right) \tag{13}$$

The transformations computed have absolute scale and thus, By concatenating the transformations, the motion of a sequence can be computed.

Initialize global rotation ($global_R$) and global translation ($global_T$) to 3x3 identity matrix and 3x1 null matrix respectively. Then repeat the following for all the images in the sequence,

$$global_R = R * global_R \quad (14)$$
$$global_T = R * global_T + t \quad (15)$$

Database

The database used in this paper is the standard KITTI odometry dataset, Karlsruhe Institute of technology, Chicago. The KITTI odometry dataset comes from real outdoor environment, and includes 9 stereo sequences saved in loss less PNG format.

Experimental Results

VO algorithm is implemented on MATLAB R2013b® and the results are presented in this section. A stereo sequence (left and right image pair) with a resolution of 1241 x 376 pixels and the baseline between the two cameras of 0.54m was employed. By using proposed algorithm, the position and orientation of the vehicle on KIITI dataset can be estimated. Due to complexity of SIFT descriptors, our VO algorithm is slower when compared to VISO2 algorithm. Our method takes about 830ms per frame and VISO2 algorithm takes 150ms per frame.

Trajectory results on KIITI sequence-00 of 430 image pairs (i.e. from image 002040.png to 002470.png) and 300 image pairs (i.e. from image 003500.png to 003800.png) are shown in Fig. 7 and Fig.8 respectively.

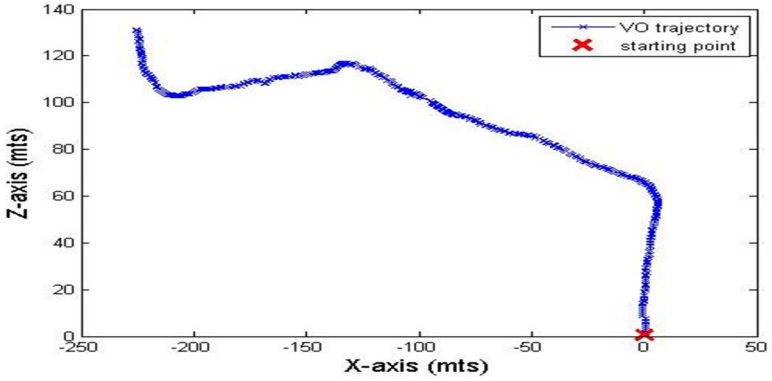

Figure8. Trajectory results on sequence-00 of 430 image pairs

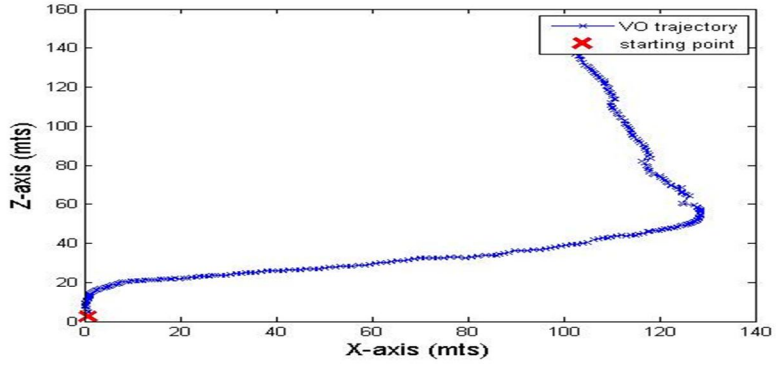

Figure9. Trajectory results on sequence-00 of 300 image pairs

Conclusion

In this paper proposed topology have implemented a stereo camera based visual odometry system with Harris corner features and SIFT descriptors are used to match features between consecutive left and right frames which improves the accuracy of motion estimation. Experiments on real outdoor data validates the performance of this algorithm.

In future we indent to decrease the computation time of the proposed algorithm. Binary descriptors may be used to speed up the performance of the algorithm and several other algorithms like bundle adjustment and local optimization may be combined with the present algorithm to reduce the drift errors, and make the algorithm work more effective and efficient.

Acknowledgment

This work is supported by ISRO satellite center (ISAC), Bangalore and Siddaganga institute of technology. We would even like to thank Karlsruhe institute of technology, Chicago for KITTI dataset.

References

[1] D. Scaramuzza, F. Fraundorfer, "Visual Odometry: Part I – The First 30 Years and Fundamentals", IEEE Robotics and Automation Magazine, vol 18, issue 4, 2011.
[2] D. Scaramuzza, F. Fraundorfer, "Visual Odometry: Part II - Matching, Robustness, and Applications", IEEE Robotics and Automation Magazine, vol 19, issue 1, 2012.
[3] Richard Hartley, "Multiple View Geometry in Computer Vision", 2nd Edition, Australian National University, Canberra, Australia and Andrew Zisserman University of Oxford, UK .
[4] Y. Lu and D. Song, "Robustness to lighting variations: An RGB-D indoor visual odometry using line segments," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 688-694.
[5] B. Zhao, T. Hu and L. Shen, "Visual odometry - A review of approaches," IEEE International Conference on Information and Automation, Lijiang, 2015, pp. 2569-2573.
[6] K. Sukvichai, K. Wongsuwan, N. Kaewnark and P. Wisanuvej, "Implementation of visual odometry estimation for underwater robot on ROS by using RaspberryPi 2," International Conference on Electronics, Information, and Communications (ICEIC), Da Nang, 2016, pp. 1-4.
[7] E. Hourdakis and M. Lourakis, "Countering drift in Visual Odometry for planetary rovers by registering boulders in ground and orbital images," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, 2015, pp. 111-116.
[8] W. Mou, H. Wang and G. Seet, "Efficient visual odometry estimation using stereo camera," 11th IEEE International Conference on Control & Automation (ICCA), Taichung, 2014, pp. 1399-1403.
[9] A. Desai and D. J. Lee, "Visual Odometry Drift Reduction Using SYBA Descriptor and Feature Transformation," IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 7, 2016, pp. 1839-1851.
[10] T. Mouats, N. Aouf, A. D. Sappa, C. Aguilera and R. Toledo, "Multispectral Stereo Odometry," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 3, 2015, pp. 1210-1224.
[11] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting oftwo 3-d point sets,” IEEE Trans. Pattern Anal. Machine Intell., vol. 9, no. 5, 1987, pp. 698–700.
[12] Harris algorithm[online] available from: http://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf
[13] SIFT algorithm[online] available from: https://www.inf.fuberlin.de/lehre/SS09/CV/uebungen/SIFT.pdf
[14] Yang Cheng, Mark Maimone and Larry Matthies, "Visual odometry on the Mars Exploration Rovers," IEEE International Conference on Systems, Man and Cybernetics, 2005, pp. 903-910.