Autonomous Vehicle Drift Corrected Global Localization using Visual Odometry

E. Ramona Stefanescu
Stanford University
Stanford, California 94305
ramona28@stanford.edu

Abstract

For autonomous driving, robust and precise self-localization of vehicles is one of the main challenges. This is typically performed using a combination of wheel odometry and inertial sensing (gyroscopes and accelerometers). This approach has two limitations: inertial sensors are prone to drift, and wheel odometry is unreliable in rough terrain (wheels tend to slip and sink). Hence, great effort is put into the improvement of Visual Odometry (VO) methods to obtain additional localization measurements for automotive applications.

1. Introduction

Many major car manufacturers, technology companies such as Google and Apple, and universities are actively working on developing self-driving cars. Google's self-driving car project relies on a combination of lasers, radars, and cameras in the form of a roof-mounted sensor pod to navigate pre-mapped environments.

In this project, I will work on improving the limitations of using an IMU + GPS system on a localization framework by augmenting it with a visual odometry module. To overcome some of the hardware/software challenges (hardware synchronization, different coordinate frame etc). I will use ROS as a middleware ecosystem.

1.1. Related work

In VO the goal is to recover the full trajectory of one camera or a camera system from images. This is incrementally done by estimating the relative transformation between the camera positions at two time steps and accumulating all transformations over time to recover the full trajectory. Given sequences of images recorded from a camera, one can estimate the ego-motion of the car from feature matches between the camera images.

During the past fifteen years, considerable research effort has already been devoted to visual odometry in computer vision and robotics. Most of previous visual odometry methods can be typically divided into three categories. The first group is based on the expensive LiDAR sensor ([9]), which registers scans between the different time stamps. The second one relies on local feature matching across video frames([1, 10]), while the third group minimized the photometric error between the current frame and reference keyframe [5]. The feature-based visual odometry approaches are currently very popular. The key is to first estimate the camera poses from the salient point correspondences through a robust estimator like RANSAC ([7, 2, 5]) where the local feature extraction algorithm plays a very important role. Then, bundle adjustment is employed to simultaneously refine the camera poses. Most direct visual odometry methods are generally based on the Lucas-Kanade framework, which is one of the most widely used techniques in computer vision. These approaches directly find the optimal geometric transformation by minimizing the photometric error between the input and the warped reference frame. The major limitation of these methods is that they tend to become stuck at a local optimum and hence require a good initialization.

1.2. Monocular visual odometry

Monocular visual odometry methods can recover the motion only up to a scale factor. The absolute scale can then be determined by computing the size of objects in the scene, from motion constraints, or integration with other sensors. Also, the incremental approach greatly suffers from drift caused by the accumulation of estimation errors of the individual transformations. It is usually addressed with an iterative refinement over the last images. This is done by reprojecting images points into 3D by triangulation and minimizing the sum of squared reprojection errors (sliding window bundle adjustment). [8] investigated the second order statistics of the essential matrix to reduce the estimation error with the eight-point method. While [4] with a ground plane estimation in a real-time SfM system deal with scale-drift. They combine multiple cues with learned models to adaptively weight per-frame observation covariances for ground plane estimation.

2. Technical Approach

There are few approaches for localization, which has been traditionally solved using Monte Carlo methods. Here, the Markov assumption is used to maintain a particle-based posterior representation of an agent's pose, integrating visual observations. Typically, these methods operate only in a small environment without providing any global (geographic) positioning information. Furthermore, and most significantly, they rely on more specific measurement sources (e.g., depth measurements, sonar) and are restricted to small-scale environments (e.g., parking lots) where accurate 2D plans are available.

- *Feature extraction* - as far as we know, all the image based localization methods use certain image features as the representations of the input images or video sequences. Directly using all the original image pixels could cause both huge storage problems and large communication bandwidth problems. Robust and distinctive features can be used to represent input images in some particular applications.

- *Global features* - are one way of representing input images of a scene. The features summarize the global information of an image, and usually, the feature representation is in the form a single vector. A global feature based method treats an input image as a whole signal, and extracts representative information in either the spatial or frequency domain of this signal.

- *Local features* - extracting global features surely will lose some of the important information in images, therefore local and descriptive features of these images can be extracted. Popular features include scale-invariant feature transform (SIFT), Speeded-Up Robust Features (SURF), etc.

The way the pose is modeled is application-specific, but for rigid mobile vehicles it usually has three degrees of freedom (two for position plus one for orientation) in the planar case and six degrees of freedom (three for position plus three for orientation) in the 3-dimensional case. Most of the localization methods use a variation of the Kalman filter or a particle filter to estimate the vehicle pose. Following the work of [3] for real-time visual odometry and improved by [10] and [5], I will implement a visual odometry pipeline that coupled with a GPS/RTK and IMU system, will provide a stable and robust global localization system.

Input A stream of gray scale images coming from a camera (initially I will test the algorithm on a mono setting and if available I will extended to a stereo system), angular rates and longitudinal accelerations from IMU and lat, lon GPS coordinates.

Figure 1: Example frame - Stanford entry circle

Output For every pair of images, we need to find the rotation matrix R and the translation vector t and integrate them with the other sensors output.

2.1. Algorithm Outline

- Capture images: I_t, I_{t+1}
- Undistort the above images
- Use a feature detection algorithm (FAST, ORB) to detect features in I_t, and track those features to I_{t+1}. A new detection is triggered if the number of features drop below a certain threshold
- Use Nister's 5-point algorithm with RANSAC to compute the essential matrix
- Estimate R, t from the essential matrix that was computed in the previous step
- Use scale information from synchronized GPS and concatenate the translation vectors, and rotation matrices.

Image capture - The camera used in the data collection was a Point Grey Blackfly Mono, 1.3 Mpx, at a 10fps. The route was the Stanford University entry circle as seen in Figure 1 and more busy neighborhood as seen in Figure 6.

Image undistortion - calibration of the camera was performed and intrinsic parameters along with distortion coeficients were determined.

Feature Detection - The main challenges in VO systems are mainly related to computational cost and light and imaging conditions. For VO to work efficiently, sufficient illumination and a static scene with enough texture should be presented in the environment to allow apparent motion to be extracted. In areas that have a smooth and low-textured surface floor, directional sun light and lighting conditions are highly considered, leading to non-uniform scene lighting.

Figure 2: FAST feature: a) I = 20 and b) I =50

Figure 3: ORB features

FAST [6] assumes that there is a point P which we want to test if it is a corner or not. We draw a circle around this point, and for every pixel which lies on the circumference of this circle, we see if there exits a continuous set of pixels whose intensity exceed the intensity of the original pixel by a certain factor I. Figure 2 shows the FAST features detected with a I threshold of 20 and 50, respectively.

ORB is a fusion of the FAST key point detector and BRIEF descriptor with some modifications. Initially to determine the key points, it uses FAST. Then a Harris corner measure is applied to find top N points. FAST does not compute the orientation and is rotation variant. It computes the intensity weighted centroid of the patch with located corner at center. The direction of the vector from this corner point to centroid gives the orientation. Moments are computed to improve the rotation invariance. The descriptor BRIEF poorly performs if there is an in-plane riotation. In ORB, a rotation matrix is computed using the orientation of patch and then the BRIEF descriptors are steered according to the orientation. The features detected are presented in Figure 3.

Feature Tracking - Kanade-Lucas-Tomasi feature tracker is used for finding sparse pixel wise correspondences. The KLT algorithm assumes that a point in the nearby space, and uses image gradients to find the best possible motion of the feature point. If, during the tracking procedure, the number of feature points go below a threshold (2000 in this case), then new detection is triggered.

Essential Matrix Estimation - Once we have point-correspondences, we have several techniques for the computation of an essential matrix. The essential matrix is defined as follows:

$$y_1^T R y_2 = 0 \tag{1}$$

Here, y_1, y_2 are homogenous normalized image coordinates. The Nister algorithm solves a number of non-linear equations, and requires the minimum number of points possible, since the Essential Matrix has only five degrees of freedom. If all of our point correspondences were perfect, then we would have need only five feature correspondences between two successive frames to estimate motion accurately. However, the feature tracking algorithms are not perfect, and therefore we have several erroneous correspondence. A standard technique of handling outliers when doing model estimation is RANSAC. It is an iterative algorithm. At every iteration, it randomly samples five points from the set of correspondences, estimates the Essential Matrix, and then checks if the other points are inliers when using this essential matrix. The algorithm terminates after a fixed number of iterations, and the Essential matrix with with the maximum number of points agree, is used.

Computing R and t

Another definition of the Essential Matrix consistent with the definition mentioned earlier is the following:

$$E = R[t]_x \tag{2}$$

Where, R is the rotation, while $[t]_x$ is the matrix representation of a cross product with t. Taking the SVD of the

Figure 4: Trajectory using FAST features for: a) I=20 and b) I=50

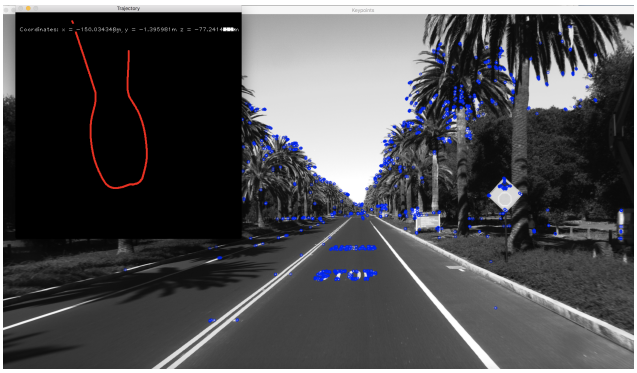


Figure 5: Trajectory using ORB features

essential matrix, and then exploiting the constraints on the rotation matrix, we get the following:

$$E = U\Sigma V^T \tag{3}$$

$$[t]_x = VW\Sigma V^T \tag{4}$$

$$R = UW^{-1}V^T \tag{5}$$

Constructing Trajectory - Let the pose of the camera be denoted by R_{pos}, t_{pos}. We can then track the trajectory using the following equation:

$$R_{pos} = RR_{pos}t_{pos} = t_{pos} + tR_{pos} \tag{6}$$

The scale information of the translation vector t has to be obtained from some other source (IMU) before concatenating. The resulted trajectory is presented in Figure 4 and Figure 5.

One of the ViO drawback it is its dependency on various camera settings and feature detection parameters. In Figure 6 and Figure 7, I show that having different setting for the exposure of the camera, keeping everything else the same (FAST features with I=20) will lead to different trajectory in camera coordinate system. However, these trajectories are smooth and coupled with GPS and IMU data can

Figure 6: Drive with different exposure settings: upper left corner exposure = 2500, upper right corner exposure = 1000, bottom exposure = 1500

overcame the global positioning system limitations: slow-convergence and noisy signal (Fig. 8).

3. Bayesian filtering

The integration of a local frame VO with a global position estimator (GPS+IMU) is done using a Bayesian filter. The filtering problem can be expressed as estimating the state x of the dynamic discrete system, given:

- the analytical knowledge of the state transition function f_t and the statistical knowledge of the state noise w_t
- the analytical knowledge of the output function h_t and the statistical knowledge of the observation noise v_t
- a realization of the system output $z_{0:t}$ up to time t.

A probabilistic filter for a system is a mathematical tool whose goal is to estimate system state history given the

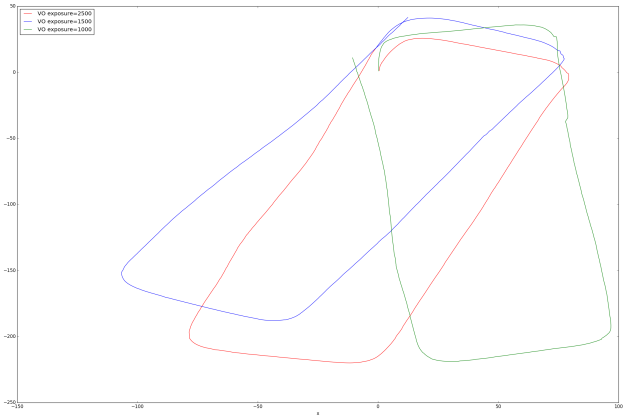


Figure 7: VO trajectories in camera coordinate system

measurements, that is estimating:

$$p(x_{0:t}|z_{0:t}) \tag{7}$$

Often it is interesting to evaluate only the marginal distribution of the current state:

$$p(x_t|z_{0:t}) \tag{8}$$

given the observations.

Many approaches for computing the state estimate have been proposed - many of them rely on the assumption that the process being observed is Markov. A process is Markov if the current measurement is independent from the past ones, given the current state: $p(z_t|z_{0:t-1}, x_t) = p(z_t|x_t)$.

Extended Kalman Fiter is the first and most direct application of the Kalman filter to non-linear systems. This filter model attempts to allow for system non-linearities by linearizing the system through the use of Jacobian matrices. This takes the for of a minimum mean-square-error estimator based on the first-order Taylor series expansion. The EKF filter equations are as follows:

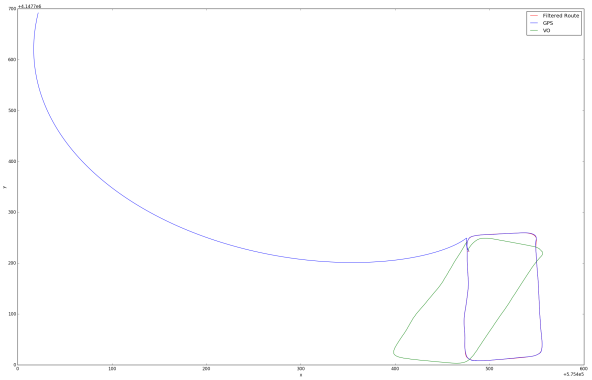
- *Prediction Phase*:

$$\hat{x}_t^- = f(\hat{x}_{t-1}, u_t) \tag{9}$$
$$P_t^- = A_k P_{k-1} A_k^T + W_t Q_t W_t^T \tag{10}$$

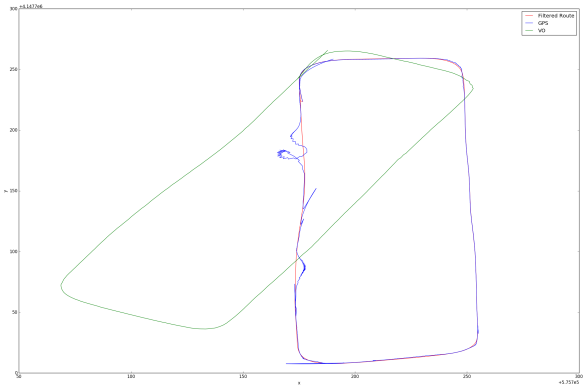
- *Update Phase*:

$$\begin{aligned} K_t &= P_t^- H_t^T (H_t P_t^- H_t^T + V_t R_t V_t^T)^{-1} \\ \hat{x}_t &= \hat{x}_t^- + K_t(z_t - h(\hat{x}_t^-)) \\ P_t &= (I - K_t H_t) P_t^- \end{aligned} \tag{11}$$

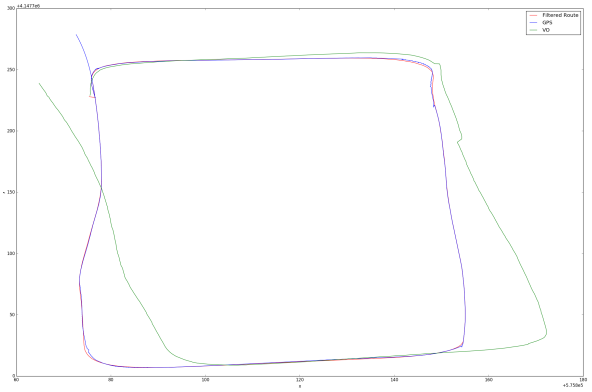
Where h, f: the non-linear measurement functions, $\hat{x}$: estimated state, $\hat{x}_k^-$: current state prediction, u: control



(a) GPS slow position convergence - cold start



(b) Noisy GPS signal



(c) Both slow position convergence and noisy GPS signal

Figure 8: VO trajectories in camera coordinate system

variables, P: state variance matrix (i.e., error of estimation), Q: Process variance matrix (i.e., error due to process), z: measurement variables, K: Kalman gain, R: measurement variance matrix (i.e., error from measurements), A: the Jacobian matrix of partial derivatives of f with respect to x, W: the Jacobian matrix of partial derivatives of f with respect to w, H: the Jacobian matrix of partial derivatives of h with respect to x, V: The Jacobian matrix of partial derivatives of h with respect to v. Subscripts are as follows:

k current time period, $k-1$ previous time period.

The available sensor data are: vehicle speed (v) in heading direction (ϕ) and a yaw rate ($\dot{\phi}$) which both have to be fused with the position (x and y) from a GPS sensor.

$$x_k = \begin{bmatrix} x \\ y \\ \phi \\ v \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} position\ X \\ position\ Y \\ heading \\ velocity \\ yaw\ rate \end{bmatrix} \tag{12}$$

The process matrix for a constant turn and constant velocity model which is used in this case has the following form:

$$g(x) = \begin{bmatrix} x + \frac{v}{\dot{\phi}}(-sin(\phi) + sin(dt * \dot{\phi} + \phi)) \\ y + \frac{v}{\dot{\phi}}(cos(\phi) - cos(dt * \dot{\phi} + \phi)) \\ dt * \dot{\phi} + \phi \\ v \\ \dot{\phi} \end{bmatrix} \tag{13}$$

The Jacobian of the process matrix J_A with respect to the state vector is defined as:

$$J_A = \begin{bmatrix} 1.0 & 0.0 & h_{13} & h_{14} & h_{15} \\ 0.0 & 1.0 & h_{23} & h_{24} & h_{25} \\ 0.0 & 0.0 & 1.0 & 0.0 & dt \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{14}$$

where:

$$h_{13} = \frac{v}{\dot{\phi}}(-cos(\phi) + cos(dt * \dot{\phi} + \phi)) \tag{15}$$

$$h_{14} = \frac{1}{\dot{\phi}}(-sin(\phi) + sin(dt * \dot{\phi} + \phi)) \tag{16}$$

$$h_{15} = \frac{dt * v}{\dot{\phi}}(cos(dt * \dot{\phi} + \phi) - \frac{v}{\dot{\phi}^2}(sin(dt * \dot{\phi} + \phi)) \tag{17}$$

$$h_{23} = \frac{v}{\dot{\phi}}(sin(dt * \dot{\phi} + \phi) - sin(\phi)) \tag{18}$$

$$J_A = \begin{bmatrix} 1.0 & 0.0 & h_{13} & h_{14} & h_{15} \\ 0.0 & 1.0 & h_{23} & h_{24} & h_{25} \\ 0.0 & 0.0 & 1.0 & 0.0 & dt \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{19}$$

$$h_{24} = \frac{1}{\dot{\phi}} * (-cos(dt * \dot{\phi} + \phi) + cos(\phi) \tag{20}$$

$$h_{25} = \frac{dt * v}{\dot{\phi}} * sin(dt * \dot{\phi} + \phi) - \frac{v}{\dot{\phi}^2}(-cos(dt * \dot{\phi} + \phi)) \tag{21}$$

Figure 9: Drive with different exposure settings: upper left corner exposure = 2500, upper right corner exposure = 1000, bottom exposure = 1500

If a GPS measurement is available, the following function maps the state to the measurement.

$$h = \begin{bmatrix} x \\ y \\ v \\ \dot{\phi} \end{bmatrix} \tag{22}$$

The matrix J_H is the Jacobian of the Measurement function h with respect to the state. Function h can be used to compute the predicted measurement from the predicted state. GPS measurements are available the J_H matrix is defined as:

$$J_H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

Otherwise, if no GPS measurements are available, then the J_H matrix is equal to:

$$J_H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

Figure 8 shows the integration of VO with GPS+IMU using an Extended Kalman filter.

4. Validation and conclusions

Due to the fact that there was no accurate GNSS data available, the validation of the integration was done by overlaying the integrated route on a Google Earth image. It can be observed in Figure 9 that the filtered route matches the road and also the driven route pretty accurately. It needs to

be mentioned however, that the EKF required multiple parameters tuning, including covariance information regarding both the measurement and process noise, as well as the covariance of some sensors (IMU in particular) for which the specs were not matching the collected data.

Advanced Driver Assistance Systems (ADAS) and systems for autonomous vehicles fuse together a rich, diverse sensor suite using compute vision (CV), LIDAR, radar, GPS, and other sensors for perception and positioning which enable increased safety and autonomy. This paper has shown that using Visual Odometry, GPS and IMU we can get good results in terms of translation and rotation of the vehicle during a drive in day time.

The code is available at: https://github.com/ramonastef28/ProjectCS231a.

References

[1] H. Badino, A. Yamamoto, and T. Kanade. Visual odometry by multi-frame feature integration. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 222–229. IEEE, 2013.

[2] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

[3] A. Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3946–3952. IEEE, 2008.

[4] M. H. Mirabdollah and B. Mertsching. Fast techniques for monocular visual odometry. In *German Conference on Pattern Recognition*, pages 297–307. Springer, 2015.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[6] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[7] D. Scaramuzza and F. Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

[8] S. Song and M. Chandraker. Robust scale estimation in real-time monocular sfm for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1566–1573, 2014.

[9] J. Zhang and S. Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2174–2181. IEEE, 2015.

[10] J. Ziegler, H. Lategahn, M. Schreiber, C. G. Keller, C. Knoppel, J. Hipp, M. Haueis, and C. Stiller. Video based localization for bertha. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 1231–1238. IEEE, 2014.