Probablistic Models for Visual Odometry

Boyang Zhang

boyangzhang06@gmail.com

***Abstract*—Visual odometry is a process to estimate the position and orientation using information obtained from a camera. We test a popular open source implementation of visual odometry SVO, and use unsupervised learning to evaluate its performance.**

Keywords—visual odometry, unsupervised learning.

I. Introduction

Visual odometry is the process of estimating position and orientation using images from an onboard camera. It is used in robotics for navigation, and can complement or even supplement the GPS and IMU module. GPS provides accurate position, but suffers from jamming, spoofing, multipath error, and indoor environments because it depends on wireless signals from satellites. MEMS IMUs provides acceleration and rotation in three axes, but suffers from inherent bias and drift due to environmental factors such as temperature, humidity, and vibration. In contrast, visual odometry provides position without depending on wireless communications, and orientation with less degrading due to environmental factors. By providing a redundant source of position and orientatin to the navigation system, the robot is more robust to failures and will be able to operate in more challenging environments.

The experiment uses SVO [6], an open source visual odometry software package, and a motion capture system with millimeter level accuracy. The camera images are fed into SVO to obtain the position and orientation estimate. A tracking device is attached to a monocular camera, and the motion capture system tracks the dongle to report the position and pose estimate. The camera and dongle is manually moved in the motion capture system. At the beginning of the test, the SVO and the motion capture system are initialized to the same initial point. Since the motion capture system is the most accurate, its estimates are regarded as the truth. The difference between the SVO estimate and the motion capture estimate is designated as the SVO error. In addition, SVO produces a number of status indicators, and the most relevant one is the number of observations. We propose using unsupervised learning to correlate the internal indicator of accuracy with the true accuracy to evaluate the effectiveness of visual odometry.

The input to the machine learning algorithm is the position error, the orientation error, and the number of observations. Mixture of Gaussian is used to find four latency multivariate Gaussians that maximizes the probability that the data lies in one of them. Since there are three different types of data, the multivariate Gaussian have three dimensions. The output of the algorithm is the mean and covariance matrix for each of the four multivariate Gaussians.

II. Related Work

A. Feature Based Visual Odometry

Feature based visual odometry correlate changes in features to the camera motion. Features from camera images are often extracted using difference of Gaussian algorithms, and popular implementations include SURF, Harris, or FAST. From the set of features, epipolar geometry algorithms such as RANSAC are used to find the change in position and orientation. On one hand, feature based visual odometry is accurate in the presence of large numbers of features and handles rapid movements. On the other hand, feature based visual odometry is slow because of the large amount of computations needed for feature detection and tracking. It could be fast enough for ground vehicles, where weight is not a constraint, and some groups have implemented this on vehicles travelling 5 meters per second [1]. However, this method is not conductive towards flying vehicles, which have weight constraints as well as real time constraints. Position and orientation estimates often need to be updated greater than once per second in order to stabilize a flying vehicle.

B. Stereo Visual Odometry

Stereo visual odometry differs from monocular visual odometry by using multiple cameras taking images of the same object. By triangulating features that appear in both camera, the distance and bearing to that feature can be calculated. As the feature moves from frame to frame, the motion of the camera can be calculated. The disadvantage to stereo visual odometry is that the distance between the two cameras is one the same order of magnitude as the distance to the feature. If the distance to the feature is an orders of magnitude or more larger than the distance between the two cameras, it degenerates to monocular visual odometry [2]. In the case of aircraft based systems, cameras may be mounted several feet or tens of feet apart, but it is expected to fly at an altitude of several hundred or thousands of feet above ground. Consequently, stereo visual odometry is not the right solution to drones. SVO uses monocular visual odometry.

C. SLAM

Highly related to visual odometry is Simultaneous Localization and Mapping [3]. SLAM utilizes onboard cameras to map the vehicle environment, and place the vehicle in the environment. While similar equipment and algorithms are used in both VO and SLAM, they are significantly different in a number of areas. Visual odometry is concerned with finding the current position and orientation, and builds a map of the surrounding in order to aid estimation of the position and orientation. On the other hand, SLAM is interested in mapping the surrounding environment, and may depend on the camera

travelling in a loop in order to accurately place the camera location by completing the loop closure. For robotics, SLAM is useful for navigation and guidance because it builds a complete map of the surrounding, and aids the vehicle in deciding where to go next. Aspects of visual odometry are used to determine the relative position of the vehicle to the environment. On the other hand, visual odometry is useful for navigation and controls because it produces the position and orientation, which is used by the aircraft to change the roll, pitch, and yaw in order to obtain the desired position and orientation.

D. Semi-Dense Visual Odometry for a Monocular Camera

An alternative to feature based visual odometry is density tracking. In density tracking, movements are tracked over time by finding the position and orientation change that minimizes the pixel difference between frames. In a recent paper [4], a proposal was put forth to use a stereo camera to capture the depth of prominent structures and use the change in the pixel locations on the depth maps to calculate position and orientation. While previous papers have proposed using the depth map of the entire image, the innovation here is to use a subset of the entire image, namely the features with structures that can be easily tracked across frames, hence the name semi-dense. By using only a subset of the density map and by restricting the computation to pixel comparisons, the algorithm can run in real time on a CPU, which improves upon previous versions that have required a dedicated GPU or were not real time. The biggest weakness with using depth map is that as the camera moves further away from the surrounding, the less accurate the depth map becomes. When the camera is mounted on an aircraft several hundred or thousand feet above the ground, the depth map will resemble white noise unless advanced imaging equipment such as lidar is used.

E. Robust Odometry Estimation for RGB-D Cameras

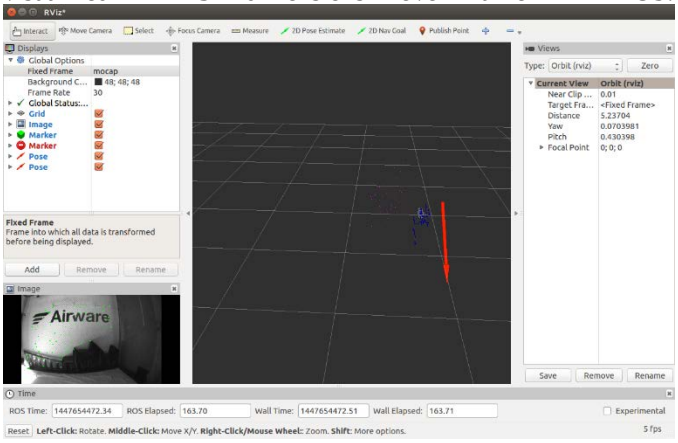
Similar to the algorithm above, this paper [5] proposes uses an RBG-D camera instead of a depth only camera. The camera continuously captures the color and distance, and the algorithm finds the motion that induced the change from one frame to the subsequent frame. This is done by taking the difference between two consecutive frames, which results in the photogrammetric error, and using that to update the state estimator. While the algorithm using the RGB-D data is expected to perform better than the one above, one remaining problem I foresee is when large scale bias is introduced in the image. For example, if cloud movement partially obscures direct sunlight on half of the image frame, the majority of the photometric error results from the shade instead of the movement of the camera.

III. DATASET AND FEATURES

The data was collected from two source. The first source is from a monocular camera, which is denoted as the raw image. The second source is from a motion capture system, which produces the position in the x, y, and z dimensions, and the orientation as a quaternion. Robot Operating System is used both to organize and store data, as well as to operate on data by applying an algorithm to a data stream.

SVO is installed as a ROS package, and used to operate on the raw image. The output of SVO are the raw image overlaid with green dots that denote the position of observations, the count of the number of observations, the number of keyframes, the tracking quality, the tracking stage, the position, the orientation, and the timestamp. Of these sets of information, the tracking quality, the count of the number of observations, the position, the orientation, and the timestamp were used as features.

Here is a screenshot of the position/orientation data visualized in 3 dimensions over time in ROS:

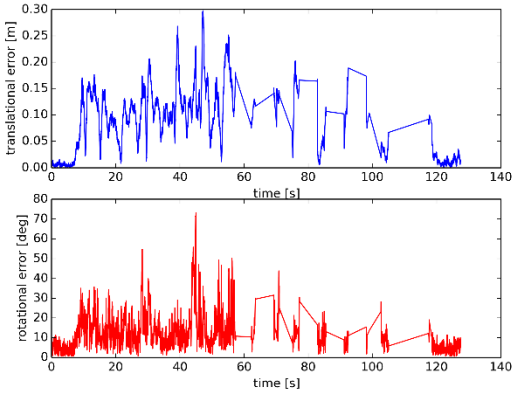


One task was to correlate each SVO position/orientation estimate with a MOCAP position/orientation estimate based on timestamp. This is needed because the two are independent systems and therefore the data overlap in time, but do not have a one to one correspondence. Two approaches were explored. The first approach is to take the weighted average of the first preceding MOCAP timestamp and the first succeeding MOCAP timestamp according to the following formula:

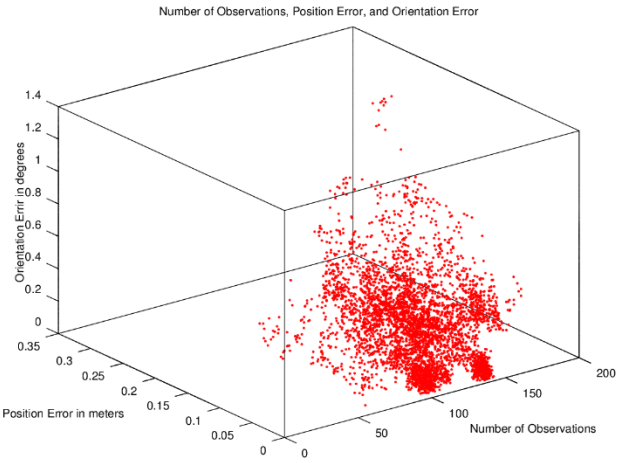
$$for\ t_a < t < t_b$$

$$x(t) = ((t - t_b) * x(t_a) + (t - t_a) * x(t_b))/(t_b - t_a)$$

The second approach is to take the nearest MOCAP timestamped data. While the second approach is simpler, it is also more accurate because there are times when the SVO stops outputting data because there is insufficient information to make a position/orientation estimate, and the last estimate could be from several seconds ago and completely inaccurate.



Finally, we obtain the position error, the orientation error, and the number of observations at each valid timestamp. The position error is the Euclidian distance between the SVO and MOCAP estimate. The orientation error is the angular distance between the SVO and MOCAP estimate which was obtained by converting the quaternion estimate back into angles along the three dimensions. These are plotted in a 3d graph below.



IV. METHODS

The machine learning algorithm used on the data is mixture of Gaussians [7]. The reason this is used is to find the latent Gaussians because we see above that the data is organized into several clusters. By finding the latent Gaussians, a model can be built to find the likelihood that the combination of the number of observations, position error, and orientation error is a valid data point, or an outlier. Our goal is to be able to find $p(x;\ \theta)$ where x is a vector of size three that represents the three inputs, and θ is the mean vector, probability vector, and covariance matrix for a three dimensional multivariate Gaussian distribution.

The mixture of Gaussians is a specialization of the EM algorithm. We would like to maximize the likelihood that the parameters with respect to the training data

$$\ell(\theta) = \sum_{i=1}^{m} \log(p(x;\theta))$$

However, explicitly finding the parameters by taking the partial derivative in the general case is difficult. Instead, we can iteratively find the local maximum using the EM algorithm.

$$\sum_{i=1}^{m} \log(p(x;\theta)) = \sum_{i=1}^{m} log \sum_{z^i} Q_i(z^i) * p(x^i, z^i;\theta)/Q_i(z^i)$$

Using Jensen's inequality, this becomes:

$$\geq \sum_{i=1}^{m} \sum_{z^i} Q_i(z^i) * log\ (p(x^i, z^i;\theta)/Q_i(z^i))$$

This becomes an equality if the distribution is constant, which is the case if we set

$$Q_i(z^i) = p(x^i, z^i;\theta)/\sum_{z} p(x^i, z^i;\theta)$$

$$Q_i(z^i) = p(x^i, z^i;\theta)/\ p(x^i;\theta)$$

$$Q_i(z^i) = p(z^i|\ x^i;\theta)$$

This is the basis for the E Step, and the M step is simply

$$argmax_\theta \sum_{i=1}^{m} \sum_{z^i} Q_i(z^i) * log\ (p(x^i, z^i;\theta)/Q_i(z^i))$$

Which can be solved by taking the derivative with respect to θ

In the case that the Qs are Gaussian random variables, this reduce to the following:

E Step:

$$w_j^i = p(z^i = j\ |x^i;\phi,\mu,\Sigma)$$

$$w_j^i = \frac{p(x^i|\ z^i = j;\mu,\Sigma) * p(|\ z^i = j;\phi)}{\sum_{l=1}^{k} p(x^i|\ z^i = l;\mu,\Sigma) * p(|\ z^i = l;\phi)}$$

$$p(x^i|\ z^i = j;\mu,\Sigma) = \frac{1}{\sqrt{(2k)^\pi|\Sigma|}} exp\left(-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)\right)$$

M Step:

$$\phi_j = \frac{1}{m}\sum_{i=1}^{m} w_j^i$$

$$\mu_j = \frac{\sum_{i=1}^{m} w_j^i x^i}{\sum_{i=1}^{m} w_j^i}$$

$$\Sigma_j = \frac{\sum_{i=1}^{m} w_j^i\ (x-\mu_j)(x-\mu_j)'}{\sum_{i=1}^{m} w_j^i}$$

V. EXPERIMENTS/RESULTS/DISCUSSION

One of the difficulties in using the mixture of Gaussians is picking the initial values, especially the mean. This is because the w_j^i can diverge to infinite if the denominator becomes approximately zero. This happens when there are data points that are several orders of magnitude smaller from the initial mean of all latent Gaussians. The exponential used to calculate the multivariate Gaussian probability becomes arbitrarily small, and this in turn causes the mean and covariance update to become even bigger because they. This causes a positive feedback cycle which converges when everything becomes infinite.

The initial values that we chose for the four latent Gaussian are the following in order to capture the largest range of possible values and prevent a divergence. The first index represents the number of observations, the second index represents the position error, and the third index represents the orientation error.

$$\mu_1 = [70; 0; 0]$$
$$\mu_2 = [100; 0; 0]$$

$$\mu_3 = [140; 0; 0]$$
$$\mu_4 = [181; 0; 0]$$

The values for ϕ were picked based on estimates from the histogram of the number of matches.

$$\phi_1 = \frac{1}{25}$$
$$\phi_2 = \frac{1}{25}$$
$$\phi_3 = \frac{1}{25}$$
$$\phi_4 = \frac{1}{25}$$

We pick the covariance matrix to be the same for all Gaussian because the initial covariance was unknown.

$$\Sigma_j = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

After the mixture of Gaussian algorithm converged, the resulting values were the following:

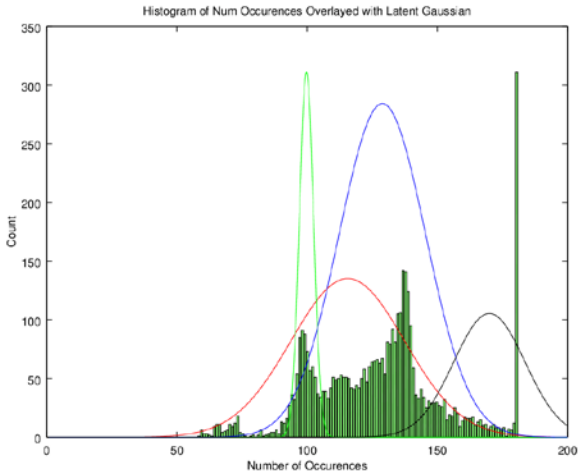
$$\mu_1 = [115.62; 0.039433; 0.10888]$$
$$\mu_2 = [99.773; 0.011172; 0.090430]$$
$$\mu_3 = [128.92; 0.11459; 0.26105]$$
$$\mu_4 = [170.02; 0.13682; 0.36728]$$
$$\phi_1 = 0.295781$$
$$\phi_2 = 0.084255$$
$$\phi_3 = 0.474947$$
$$\phi_4 = 0.145017$$

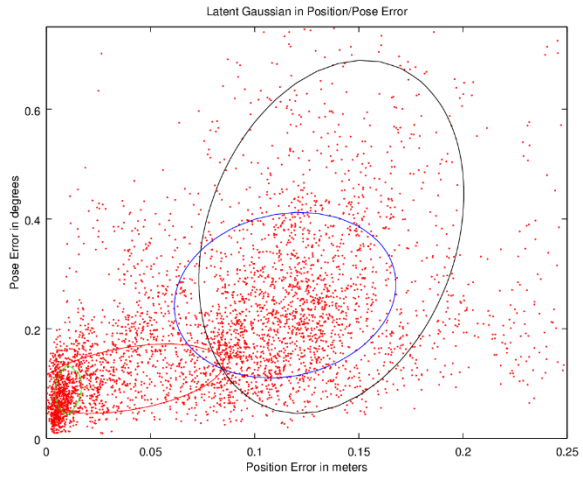
$$\Sigma_1 = \begin{matrix} 476.57 & -0.50995 & -0.45143 \\ -0.50995 & 0.0014902 & 0.00086440 \\ -0.45143 & 0.00086440 & 0.0029082 \end{matrix}$$

$$\Sigma_2 = \begin{matrix} 7.2986 & -0.0057415 & -0.016763 \\ -0.0057415 & 0.000033071 & 0.000076833 \\ -0.016763 & 0.000076833 & 0.0016774 \end{matrix}$$

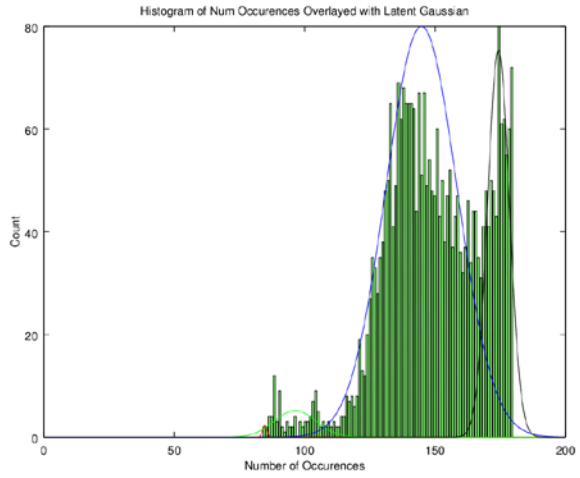
$$\Sigma_3 = \begin{matrix} 277.79 & -0.0095786 & -0.58304 \\ -0.0095786 & 0.0020218 & 0.00077565 \\ -0.58304 & 0.00077565 & 0.016319 \end{matrix}$$

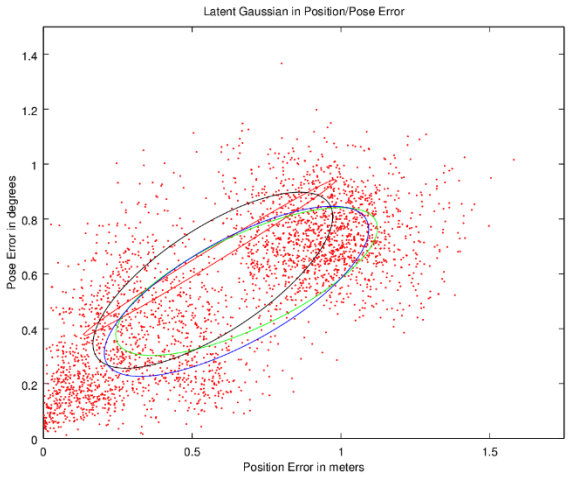
$$\Sigma_4 = \begin{matrix} 188.54 & -0.11415 & -2.9577 \\ -0.11415 & 0.0029274 & 0.0036221 \\ -2.9577 & 0.0036221 & 0.075097 \end{matrix}$$





To verify this, we collected another set of datapoint, and used the same analysis on the data. The histogram of the number of occurrences matches the overall shape of the training set. However, the position error and orientation error has been scaled up by a factor of 5. Additionally, there are two distinct clumps of data, one with low error, and one with high error, just like in the training data. However, the test example had difficulties correlating the number of occurrences with a clump of data, as it did in the test set.





$$\mu_1 = [84.75219; 0.55864; 0.65864]$$
$$\mu_2 = [96.41319; 0.68283; 0.57185]$$
$$\mu_3 = [144.81366; 0.53611; 0.53611]$$
$$\mu_4 = [174.34948; 0.56950; 0.57669]$$

$$\phi_1 = 0.0013048$$
$$\phi_2 = 0.0283962$$
$$\phi_3 = 0.7536366$$
$$\phi_4 = 0.2166623$$

$$\Sigma_1 = \begin{matrix} 0.693302 & -0.109364 & -0.053949 \\ -0.109364 & 0.133485 & 0.088921 \\ -0.053949 & 0.088921 & 0.059885 \end{matrix}$$

$$\Sigma_2 = \begin{matrix} 62.968265 & -1.136013 & -0.403983 \\ -0.136013 & 0.000033071 & 0.061765 \\ -0.403983 & 0.061765 & 0.052326 \end{matrix}$$

$$\Sigma_3 = \begin{matrix} 186.81. & -0.23617 & -0.13961 \\ -0.23617 & 0.14236 & 0.072208 \\ -0.13961 & 0.072208 & 0.069324 \end{matrix}$$

$$\Sigma_4 = \begin{matrix} 17.402843 & -0.041761 & -0.130936 \\ -0.041761 & 0.117023 & 0.0036221 \\ -0.130936 & 0.0036221 & 0.074230 \end{matrix}$$

VI. CONCLUSION/FUTURE WORK

As seen from the unsupervised learning algorithm, there is a positive correlation between the position error in meters and orientation error in degrees. This is what we had expected, because as the vehicle made difficult maneuvers, the error in both position and orientation would increase. Additionally, there is negative correlation between the number of occurrences, which is an SVO indicator for its estimated accuracy, and the error in both position and orientation. The reason why the correlation is negative is because the number of occurrences represents the number of depth estimates. Depth estimates occur when the algorithm has a poor estimate, but as the estimate improves with time, the depth estimate turns into a 3D point, and the number of occurrences becomes reduced. Therefore, we conclude that we were successful in evaluating the SVO algorithm and software package using unsupervised learning.

VII. ACKNOWLEDGMENT

I would like to thank Dr. Hui Li for her assistance and guidance.

REFERENCES

[1] Konolige, Kurt, Motilal Agrawal, and Joan Sola. "Large-scale visual odometry for rough terrain." Robotics Research. Springer Berlin Heidelberg, 2011. 201-212.

[2] Nistér, David, Oleg Naroditsky, and James Bergen. "Visual odometry." Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. Vol. 1. IEEE, 2004.

[3] Scaramuzza, David, and Friedrich Fraundorfer. "Visual Odometry Part 1: The First 30 Years and Fundamentals." IEEE Robotics & Automation Magazine, December 2011.

[4] Engel, Jakob, Jurgen Sturm, and Daniel Cremers. "Semi-dense visual odometry for a monocular camera." Computer Vision (ICCV), 2013 IEEE International Conference on. IEEE, 2013.G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[5] Kerl, Christian, Jurgen Sturm, and Daniel Cremers. "Robust odometry estimation for RGB-D cameras." Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013.

[6] Forster, Christian, Matia Pizzoli, and Davide Scaramuzza. "SVO: Fast semi-direct monocular visual odometry." Robotics and Automation (ICRA), 2014 IEEE International Conference on. IEEE, 2014.

[7] Ng, Andrew. "CS229 Lecture Notes.