TSET-UTC 212 Final report: Visual Navigation with Android Tablets

Kostas Daniilidis
University of Pennsyvlania

We implemented our monocular visual inertial odometry for several mobile platforms and made it available in `https://github.com/daniilidis-group/msckf_mono` . Our implementation was tested in a 3rd party paper (A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots. Demerico and Scaramuzza, ICRA 2017) and performed best regarding platform universality.

We described in the following the filter and outlier rejection implemented.

1. State Estimation

Algorithm 1 State Estimation

Input
sensor state s_i, features $\{f\}$
IMU values $\mathcal{I}$ for $t \in [T_i, T_i + dt_i]$

Filter
Propagate the sensor state mean (1) and covariance (2)
Augment a new camera state
for each filter track to be marginalized **do**
Remove inconsistent observations
Triangulate the feature using GN Optimization
Compute the uncorrelated residuals $r_0^{(j)}$ (3)
Stack all of the $r_0^{(j)}$
Perform QR decomposition to get the final residual (4)
Update the state and state covariance

To estimate the 3D pose of the camera over time, we employ an Extended Kalman Filter with a structureless vision model, as first developed in [4]. For compactness, we do not expand on the fine details of the filter, and instead refer interested readers to [3] and [4]. At time T_i, the filter tracks the current sensor state (**??**) as well as all past camera poses that observed a feature that is currently being tracked. The full state, then, is:

$$S_i := S(T_i) = \begin{bmatrix} s_i^T & \bar{q}(T_{i-n})^T & p(T_{i-n})^T & \dots & \bar{q}(T_i)^T & p(T_i)^T \end{bmatrix}^T$$

where n is the length of the oldest tracked feature.

1.1. Prediction Step

Between update steps, the prediction for the sensor state is propagated using the IMU measurements that fall in between the update steps. Note that, due to the high temporal resolution of the event based camera, there may be multiple update steps in between each IMU measurement. In that case, we use the last IMU measurement to perform the propagation.

Given linear acceleration a_k and angular velocity ω_k measurements, the sensor state is propagated using 5th order Runge-Kutta integration:

$$\begin{aligned} \dot{\bar{q}}(\tau_k) &= \frac{1}{2}\Omega(\omega_k - \hat{b}_g(\tau_k))\bar{q}(\tau_k) & \dot{b}_a(\tau_k) &= 0 \\ \dot{p}(\tau_k) &= v(\tau_k) & \dot{b}_g(\tau_k) &= 0 \\ \dot{v}(\tau_k) &= R(\bar{q}(\tau_k))^T(a_k - \hat{b}_a(\tau_k)) + g \end{aligned} \tag{1}$$

To perform the covariance propagation, we adopt the discrete time model presented in [2]. The IMU error covariance is propagated with the discrete-time state transition matrix Φ_k and the discrete-time system noise covariance Q_k:

$$\begin{aligned} P_{k+1|k} =& \Phi_k P_{k|k} \Phi_k^T + Q_k \\ \Phi_k =& \exp(F \cdot (\tau_{k+1} - \tau_k)) \\ Q_k =& \Phi_k G Q_k \Phi_k \cdot (\tau_{k+1} - \tau_k) \end{aligned} \tag{2}$$

where the linearization of the sensor error state $\tilde{s}$ is:

$$\dot{\tilde{s}} = F\tilde{s} + Gn_s$$

1.2. Update Step

When an update from the tracker arrives, we augment the state with a new camera pose at the current time, and update the covariance using the Jacobian that maps the IMU state to the camera state.

We then process any features that need to be marginalized. For any such feature f_j, its past observations $\{f_j(T_i)\}$ and the camera poses at times $\{T_i\}$ for each observation can be used to estimate the 3D position of the feature $\hat{F}_j$ using Gauss Newton optimization, assuming the camera poses are known [1]. The projection of this estimate into a given camera pose $C_i := \begin{bmatrix}\bar{q}(T_i)^T & p(T_i)^T\end{bmatrix}^T$ can then be computed. The residual, $r^{(j)}$, computed for each feature at each camera pose is then simply the difference between the observed feature position and the estimated position. We then left multiply $r^{(j)}$ by the left null space, A, of the feature Jacobian, H_F, as in [4], to eliminate the feature position up to a first order approximation:

$$\begin{aligned} r_0^{(j)} =& A^T r^{(j)} \\ \approx& A^T H_S^{(j)} \tilde{S} + A^T H_F^{(j)} \tilde{F}_j + A^T n^{(j)} := H_0^{(j)} \tilde{S} + n_0^{(j)} \end{aligned} \tag{3}$$

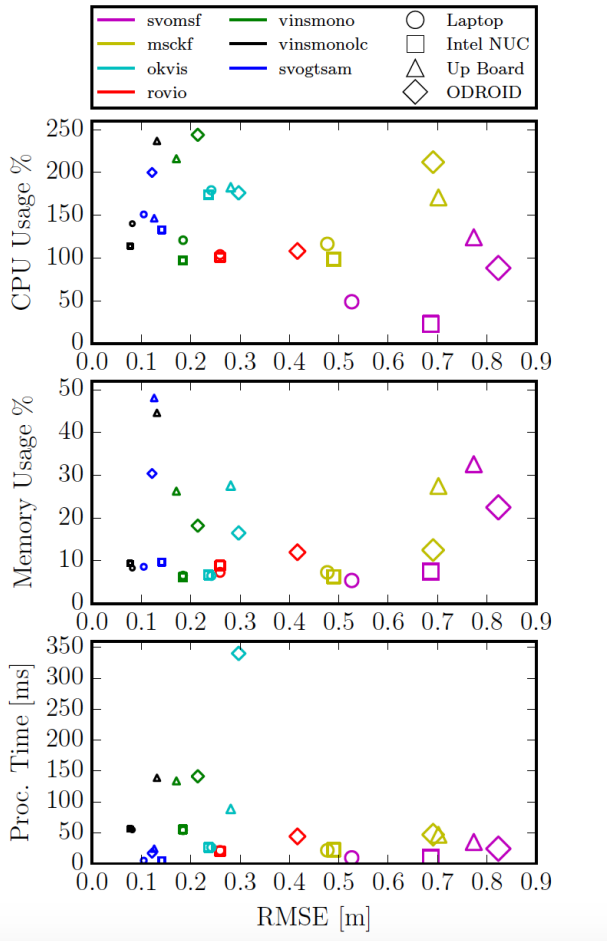
The elimination procedure is performed for all features, and the remaining uncorrelated residuals, $r_0^{(j)}$ are stacked to obtain the final residual r_0. As in [4], we perform one final step to reduce the dimensionality of the above residual. Taking the QR decomposition of the matrix H_0, we can eliminate a large part of the residual:

$$\begin{aligned} r_n =& Q_1^T r_0 \\ H_0 =& \begin{bmatrix}Q_1 & Q_2\end{bmatrix} \begin{bmatrix}T_H \\ 0\end{bmatrix} \end{aligned} \tag{4}$$

The EKF update step is then $\Delta S = Kr_n$.

When a feature track is to be marginalized, we apply a second RANSAC step to find the largest set of inliers that project to the same point in space, based on reprojection error. This removes moving objects and other erroneous measurements from the track.

You can see below the performance of our system (denoted msckf) compared with other competing algorithms over several mobile platforms as presented in Demerico and Scaramuzza.



References

[1] L. Clement, V. Peretroukhin, J. Lambert, and J. Kelly. The battle for filter supremacy: A comparative study of the multi-state constraint kalman filter and the sliding window filter. In *Computer and Robot Vision (CRV), 2015 12th Conference on*, pages 23–30, 2015. 2

[2] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis. Observability-constrained vision-aided inertial navigation. *University of Minnesota, Dept. of Comp. Sci. & Eng., MARS Lab, Tech. Rep*, 1, 2012. 2

[3] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. 2006. 1

[4] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. Technical report, 2007. 1, 2