ECE 276A Project 3
Visual Inertial SLAM

Shiladitya Biswas
Dept. of Electrical and Computer Engineering
University of California,San Diego
California, USA

I. Introduction

Physical world is unpredictable and full of uncertainties. A flat floor might not be flat, a straight wall may not be straight, floor/ground friction will always vary from place to place. In the world of robotics we try to perceive and manipulate this physical world through computer generated signals which we can very well control at our own will. Similarly there will always be uncertainties, manufacturing fault etc in a robot hardware. These faults include faulty sensor, worn out rubber wheels (less friction), etc. Due to these uncertainties the robot's real state in the physical world is a function of both the computer signal(Known signal) and the environment state(completely unknown). For example, the computer sends instruction to the motor driver to rotate 5 time to travel 5m in the forward direction, but dues to worn out wheels and slightly steep floor the wheel couldn't generate the required traction and the robot only travelled 4 meters. Thus the robot controller and the real world goes out of sync i.e. the controller thinks the robot has travelled 5m forward but in reality it hasn't. This may lead to several fatal accidents (Self-Driven cars, rehabilitation robotics etc) and hence robots won't be safe to use. Thus it is evident that for a robot to manipulate/work safely in an environment, we need to take care of the unknown uncertainties and accordingly incorporate environment state estimation capabilities in the robot. Simultaneous Localization and Mapping (SLAM) is one such capability which allows a robot to simultaneously generate a Map of the unknown environment it is in and localize itself within the generated MAP. In this project we are given a set of sensor readings (Stereo camera pixels of certain landmarks) and Robot(Car) pose, using which we have to estimate the trajectory of the robot. The project consists of 3 steps: Mapping, Prediction and Update. Mapping is achieved using the Stereo camera pixel data given. Prediction and update is performed using the IMU data (Linear and Angular velocity) and the Extended Kalman filter.

II. Problem Formulation

As discussed in the previous section, we are trying to solve a mapping and localization problem i.e. SLAM. The central goal of SLAM can be stated as follows: **Given a series of control signals and sensor measurements, estimate the unknown environment/Map and its pose relative to this map.**

Mathematically speaking, given a dataset of the robot inputs $u_{0:T1}$ and observations $z_{0:T}$ we have to maximize the data likelihood conditioned on parameters Map m and state $x_{0:T}$:

$$\text{max} x_{0:T}, m \text{log p}(z_{0:T}, u_{0:T1} | x_{0:T} \text{ , m}) \tag{1}$$

Where $x_{0:T}$ is pose trajectory of the robot from t=0 to T and $z_{0:T}$ is the set of all observations from t=0 to T. and m is the map.

Other than this we will also implement a Bayesian filter that keeps track of the the probabilities given below:

$$p_{t|t}(x_t) = p(x_t | z_{0:t}, u_{0,t-1}) \tag{2}$$

$$p_{t+1|t}(x_{t+1}) = p(x_{t+1} | z_{0:t}, u_{0,t}) \tag{3}$$

where $p_{t|t}$ represents the pdf of robot state at time t conditioned on all the observations upto time t and control input upto time t-1. Similarly, $p_{t+1|t}$ represents the pdf of robot state at time t+1 given all the observations and control inputs upto time t. In this project we implement an Extended Kalman Filter (EKF) which is a type of Bayes filter to solve the localization problem.

From class we learnt that SLAM is a chicken and egg problem comprising of :

1) **Mapping:** Given Stereo Camera pose, four pixel coordinates of a landmark, transform the depth lengths from the Camera Frame to the world frame using the car's IMU data given to us. We then build the map by plotting the coordinates of the landmarks obtained in the world frame.
2) **Localization:** Given the position of the landmarks in the environment, localize the robot and estimate its trajectory $X_{0:T}$.

In order to apply the Extended Kalman Filter (EKF) to this project the following assumptions were made.

1) The prior probability is a Gaussian.
2) The motion model is affected by Gaussian noise.
3) The observation model is affected by Gaussian noise.
4) The process noise w_t and measurement noise v_t are independent of each other, of the state x_t and across time.
5) The posterior pdf is forced to be Gaussian via approximation.

I would like to thank Saurabh Mirani and Ayon Biswas for the helpful discussions on the project.

A. Motion Model

Let the motion model of the robot be defined as follows

$$x_{t+1} = f(x_t, u_t, w_t) \tag{4}$$

Applying the assumptions as stated above, we have prior of $x_t \sim \mathcal{N}(\mu_{t|t}, \Sigma_{t|t})$ and $w_t \sim \mathcal{N}(0, W)$. EKF uses first order taylor series expansion to linearize the nonlinear motion equation of the robot around the prior mean. Thus the ‘approximate equation becomes:

$$f(x_t, u_t, w_t) \approx f(\mu_{t|t}, u_t, 0) + F_t \times (x_t - \mu_{t|t}) + Q_t \times (w_t - 0) \tag{5}$$

where $F_t = \frac{df(x_t,u_t,0)}{dx}$ and $Q_t = \frac{df(x_t,u_t,0)}{dw}$. Since both the w_t and x_t are independent of each other and Gaussian distributions are stable, we can predict the pdf of x_{t+1}, i.e. it will also be a Gaussian. The predicted mean and covariance of the the next state is as follows:

$$\mu_{t+1|t} = f(\mu_{t|t}, u_t, 0) \tag{6}$$

$$\Sigma_{t+1|t} = F_t \Sigma_{t|t} F_t^T + Q_t W Q_t^T \tag{7}$$

B. Observation Model

Let the observation model of the robot be defined as:

$$z_{t+1} = h(x_{t+1}, v_{t+1}) \tag{8}$$

where $v_t \sim \mathcal{N}(0, V)$. As in the motion model, the observation model is also linearized about the predicted mean from equation 6. Hence we get,

$$\begin{aligned} h(x_{t+1}, v_{t+1}) \approx h(\mu_{t+1|t}, 0) + H_{t+1} \times (x_{t+1} - \mu_{t+1|t}) \\ + R_{t+1} \times (v_{t+1} - 0) \end{aligned} \tag{9}$$

where, $H_{t+1} = \frac{dh(\mu_{t+1|t},0)}{dx}$ and $R_{t+1} = \frac{dh(\mu_{t+1|t},0)}{dv}$. Using these the final updated mean and covariance of the robot state and landmark pose are as follows:

$$\mu_{t+1|t+1} := \mu_{t+1|t} + K_{t+1|t}(z_{t+1} - m_{t+1|t}) \tag{10}$$

$$\Sigma_{t+1|t+1} := \Sigma_{t+1|t} - K_{t+1|t} S_{t+1|t} K_{t+1|t}^T \tag{11}$$

Where,

$$K_{t+1|t} := C_{t+1|t} S_{t+1|t}^{-1} \tag{12}$$

$$m_{t+1|t} \approx h(\mu_{t+1|t}, 0) \tag{13}$$

$$S_{t+1|t} \approx H_{t+1} \Sigma_{t+1|t} H_{t+1}^T + R_{t+1} V R_{t+1}^T \tag{14}$$

$$C_{t+1|t} \approx \Sigma_{t+1|t} H_{t+1}^T \tag{15}$$

C. Calculating Depth from Stereo Camera pixel values

The stereo camera is inspired by the biological vision system which enables the robot to perceive depth. When the calibration matrix of both cameras and the baseline distance between the cameras are known, we can calculate the depth (i.e. distance from the camera) of a landmark depending upon the pixel coordinates of the landmark in both cameras. This is done as follows. Given the pixel coordinates $[u_L, v_L, u_R, v_R]$ for a given landmark $[x, y, z]$ in the optical frame, we have

$$\begin{bmatrix} u_L \\ v_L \\ u_R \\ v_R \end{bmatrix} = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ fs_u & 0 & c_u & -fs_u b \\ 0 & fs_v & c_v & 0 \end{bmatrix} \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \\ \frac{1}{z} \end{bmatrix}$$

where, f= focal length, (s_u, s_v) = Scaling from meter to pixels, (c_u, c_v) = Image center, b = Baseline distance between the 2 cameras.
Solving the above matrix equations, we get

$$Depth, z = \frac{fs_u b}{u_L - u_R} \tag{16}$$

$$x = \frac{u_L - c_u}{fs_u} \tag{17}$$

$$y = \frac{v_L - c_v}{fs_v} \tag{18}$$

These values obtained were then transformed to the world frame using the inverse of camera to IMU transformation matrix given to us with the dataset. This information was then used to predict and update landmark position and robot location means simultaneously.

III. Technical Approach

A. Prediction

We used the IMU data namely linear and rotational velocity to estimate the location of the IMU (i.e. the car) in the map. To keep things simple we worked with the discretized version of the kinematics and zero-mean perturbation kinematics. We used exponential map to transform the perturbation equations from $se(3)$ to $SE(3)$. Therefore, the predicted IMU pose is:

$$\mu_{t+1|t}^{imu} = \exp{(-\tau \hat{u}_t)} \mu_{t|t}^{imu}) \tag{19}$$

$$\Sigma_{t+1|t}^{imu} = \exp{(-\tau u_t^{\wedge})} \Sigma_{t|t}^{imu} \exp{(-\tau u_t^{\wedge})} + W \tag{20}$$

where, $\hat{u}_t$ represents hat map of velocity vector at time t, $u_t = [v_t, w_t]^T$, and $u_t^{\wedge}$ represents the twist vector defined as

$$u_t^{\wedge} = \begin{bmatrix} \hat{w}_t & \hat{v}_t \\ 0 & \hat{w}_t \end{bmatrix} \tag{21}$$

W is the covariance of noise in the motion model as defined in equation 4.

B. *Mapping*

We use the predicted mean in above subsection to update the landmark mean and covariance. We initialized the i^{th} landmarks with prior mean and covariance

$$\mu_{t|t,i}^{landmark} = [-1,-1,-1,-1]^T \tag{22}$$

$$\Sigma_{t|t,i}^{landmark} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{23}$$

Where, i goes from 1,2,3.....N. N= Total number of features at a given time t. For a given timestamp t we find the valid set of landmarks by looking at the 4xN feature matrix and searching for indices whose values are not $[-1,-1,-1,-1]^T$. These respective pixel values are then converted to the world frame (as explained in section II-C) with respect to the previously predicted IMU pose from equation 6 and the landmark means are reinitialized with these converted values. When we encounter landmarks that has already been seen before we update the mean and covariance of these landmarks using equations 10 and 11. In order to do so we build the H_t and K_t (as in equation **??** and **??**) matrix for these features.

Now, although we have made a bold update to the landmark's mean and covariance for a given timestamp in the previous paragraph, this landmark mean was obtained based on the predicted mean and covariance of the IMU pose from equation 6and 7. Hence, there is no guarantee that it will be correct. So we proceed to the next step i.e. Visual-Inertial SLAM, in which we update the pose of both the landmarks and IMU of the car at a time.

C. *Visual-Inertial SLAM/ Update*

Now that we got our updated landmarks and predicted IMU pose we compute the new predicted observation of pixels for each of the updated landmark means using the reverse of the step explained II-C. The equation is as follows:

$$\tilde{z}_{t+1,i} = M\pi(T_{CI} \times \mu_{t,i}^{Landmark}) \tag{24}$$

where, M is the stereo camera calibration matrix from equation II-C, T_{CI} is the IMU to camera frame transformation matrix, $\mu_{t,i}^{Landmark}$ is the mean pose of the ith landmark (calculated in previous subsection using equation 10) and $\pi()$ is the canonical projection function. This is done for all the N landmark points. We then compute the Jacobian of $\tilde{z}_i$ by taking the derivative of equation 24. The final Jacobian matrix obtained for a given vector $q = [q_1, q_2, q_3, q_4]$ is as follows:

$$\frac{d\pi}{dq} = \begin{bmatrix} 1 & 0 & -q_1/q_3 & 0 \\ 0 & 1 & -q_2/q_3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -q_4/q_3 & 1 \end{bmatrix} \tag{25}$$

Now, we update the pose of IMU, using equations 24, 25 and the equations below. We first build the $H_{t+1|t}$ block diagonal sparse matrix as follows:

$$H_{t+1|t,i} = M\frac{d\pi}{dq}(T_{CI}\mu_{t+1|t,i}m_i)T_{CI}((\mu_{t+1|t,i}m_i)^o) \tag{26}$$

Similarly we calculate the Kalman gain matrix from the $H_{t+1|t}$ as follows.

$$K_{t+1|t} = \Sigma_{t+1|t}^{IMU} H_{t+1|t}^T (H_{t+1|t}\Sigma_{t+1|t}H_{t+1|t}^T + V)^{-1} \tag{27}$$

Where, V is the observation noise covariance. The IMU pose update equation is as follows,

$$\mu_{t+1|t+1}^{IMU} = \exp\left(K_{t+1|t}(\hat{z}_{t+1} - \tilde{z}_{t+1})\right)\mu_{t+1|t}^{IMU} \tag{28}$$

The covaraince of the IMU pose is updated as follows:

$$\Sigma_{t+1|t+1}^{IMU} = (I - K_{t+1|t}H_{t+1|t,i})\Sigma_{t+1|t}^{IMU} \tag{29}$$

Now, using the updated IMU mean Pose, we update the landmark pose mean and covariance as per the steps discussed in section III-B. The mean of each landmark at a given timestamp t, corresponds to the world coordinates of the landmarks in the map. The updated IMU mean pose at a given timestamp t, gives us the trajectory of the IMU/ car in the world.

IV. Results and Discussion

The estimated trajectory of the robot/car for each data set is shown below. While changing V doesn't make much visible difference to the trajectory, the trajectory slightly changed when we change the W value. This shows how robust the EKF is to noise. The case wise result for each dataset is discussed and illustrated below.

The trajectory estimate for dataset no 0027 is shown in figure 1. For this I have randomly selected every 5th feature from the feature matrix for a given timestamp. When, I took every 25th sample from the feature matrix per timestamp, the

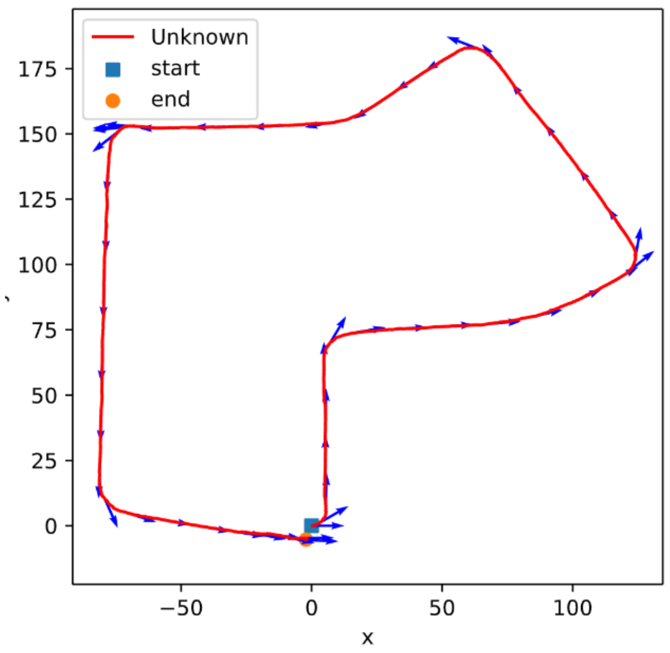

Fig. 1. Trajectory estimate by taking 790 (out of 3950) evenly sampled features per timestamp (Dataset 0027 V=15,W=0.001)

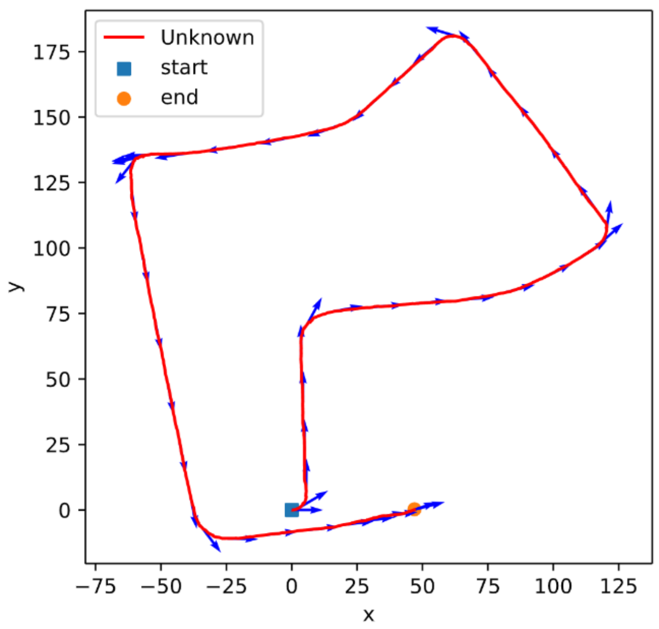

Fig. 2. Trajectory estimate by taking 158 (out of 3950) evenly sampled features per timestamp (Dataset 0027 V=15,W=0.001)

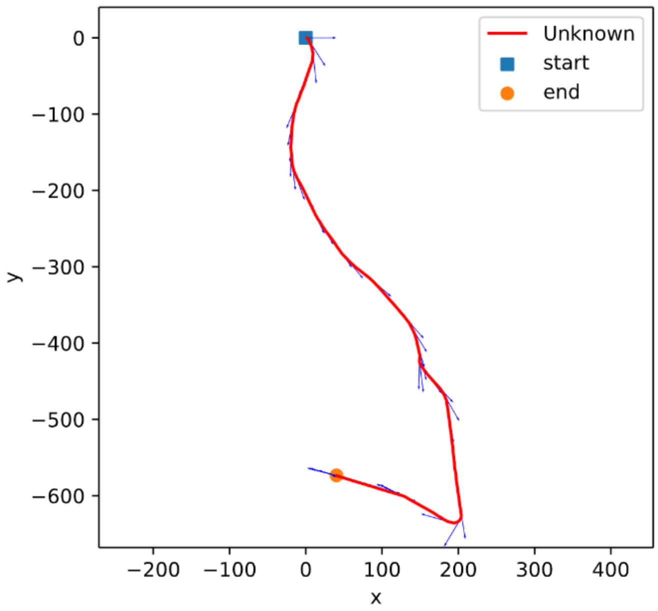

Fig. 3. Trajectory estimate by taking 963 (out of 4815) evenly sampled features per timestamp (Dataset 0034 V=15,W=0.001)

program runtime reduced. But then I wasn't able to get a loop closure. This fact is illustrated in fig 2.

From this we conclude that, good feature selection forms an essential part of Visual Inertial SLAM, especially to detect loop closures. Thus, there is a trade off between program runtime and performance. A solution to this problem is to cherry pick the most frequently visited/ seen feature, but even this has a drawback. It is an obvious fact that throught time the most least visited feature will be present at the turning points of the trajectory. Thus this solution may also lead to bad trajectory prediction. The safest way to get a good trajectory estimate is to take into consideration all the features at a give timestamp.

Trajectory estimate of Dataset no 0034 is show below in figures 3 and 4. It can be observed that both the downsampled versions i.e. 25 and 5 are almost similar, they have approximately the same starting and ending points. This is due to the fact that there is no loop closure in this dataset. As a result of which, towards the end of the trajectory, there is less previously visited features as compared to dataset 0027, thus there is less deviation at the end. On comparing the plots with the video provided to us, it is evident that figure 3 i.e. feature matrix sampled at intervals of 5 performs better as compared to fig. 4.

The trajectory estimate of dataset 0022 is shown in figures 5 and 6. As in dataset 0034, it is observed that downsampling has no effect on the initial and final postion of the car. This is again due to the fact that, there is no loop closure in this trajectory. The only visible difference is the fact that 6 dataset i.e. 25 Downsample data, predicted trajectory is a bit left drifted, this is because by downsampling, we are essentially ignoring some vital feature in every timestamp, hence the predicted

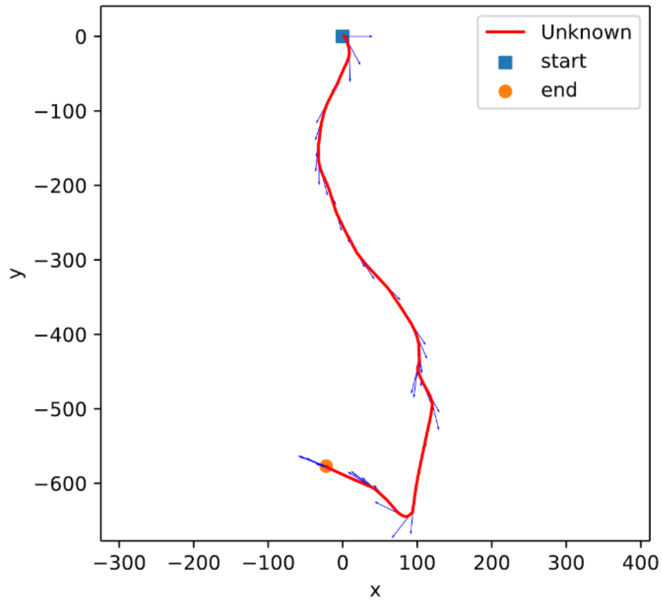

Fig. 4. Trajectory estimate by taking 193 (out of 4815) evenly sampled features per timestamp (Dataset 0034 V=15,W=0.001)

trajectory is a bit translated. This again endorses the fact that data preprocessing is a essential part of SLAM.

In conclusion we see that, stereo camera gives a good perception of depth and hence enable us to predict the trajectory of the car in the world. The IMU pose prediction step forms an essential part of Visual inertial SLAM, it helps us to roughly get the map and then ultimately update the pose of both the landmarks and the IMU in the map, thus aiding in localization. It is also observed that the Extended Kalman filter algorithm

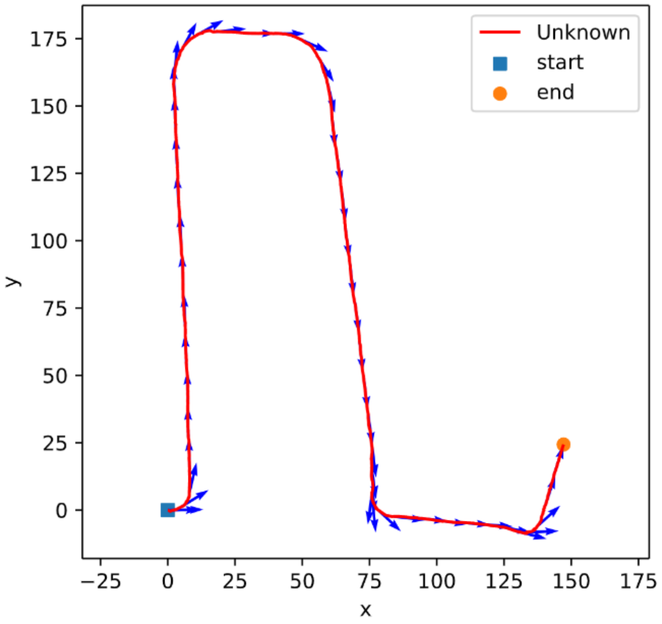

Fig. 5. Trajectory estimate by taking 644 (out of 3220) evenly sampled features per timestamp (Dataset 0022 V=15,W=0.001)

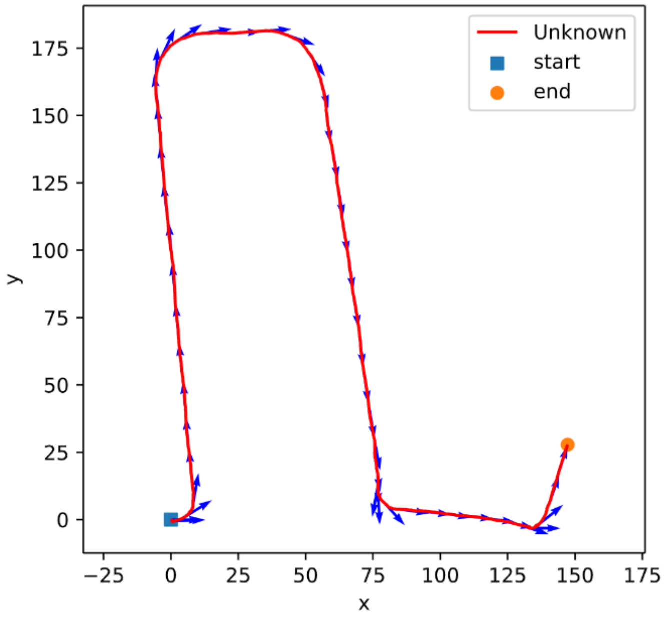

Fig. 6. Trajectory estimate by taking 129 (out of 3220) evenly sampled features per timestamp (Dataset 0022 V=15,W=0.001)

is very robust to both motion and process noise, this is a direct result of the assumptions we made in section II.