Investigation of Different Filters Based on Gesture Recognition System

Chen Du
PID: A53223649
c9du@eng.ucsd.edu

Wenjun Zhang
PID: A53218995
wez177@eng.ucsd.edu

Jingyi Yang
PID: A53216941
jiy349@eng.ucsd.edu

Jinzhao Feng
PID:
@eng.ucsd.edu

Abstract

Gesture recognition is an important task of modern Computer Vision, which has been widely used in human-computer interaction, game industry and internet of things. A big challenge in gesture recognition is the high precision tracking of fingers and hand. Many filter algorithms has been proposed for object tracking with different characterizations. Given correct motion model, Unscented Kalman Filter is one of the best choice. In this project, we build a gesture recognition system which could identify simple number gestures, and implement four kinds of filter algorithms to investigate their performances in different scenarios. Results show that Unscented Kalman Filter has overall best performance and robustness, and particle filter with enough particles also has great tracking precision without interference.

1. Introduction

Gestures are naturally performed by human-being. Produced as a part of deliberate actions, signs or subconscious intentions and attitude, hand gestures are essential in communication, which has been the focus of studies. Gestures are present in most daily human actions and participate in human communication by either complementing speech or substituting themselves to spoken language in environments requiring silent communication (under water, noisy environments, secret communication, etc.) or for people with hearing disabilities. [1]

Given the indubitable importance of gestures in human activities, there has been huge interest from the Computer Vision and Machine Learning communities to analyze human gestures from visual data in order to offer new non-intrusive technological solutions. In particular, interfaces incorporating hand gestures have gained popularity [2] in automobile industry. For example, BMW has applied gesture recognition in their newest iDrive system so as to control the vehicle's navigation, buttons, knobs, windows and voice commands. Moreover, depth cameras have been widely utilized in camera tracking for augmented and mixed reality [3], as well as popular games such as Just Dance from Ubisoft.

To capture the meaning of the gesture, finger tracking is the fundamental part. Many tracking techniques are available in object tracking. For example, the Kalman Filter, which has been introduced decades ago, and its corresponding improvements, the Extended Kalman Filter and Unscented Kalman Filter, which are used to deal with nonlinear problems. Also, the Particle Filter, which is usually applied with color detection in object tracking. However, each technique is of their own strengths and weakness, under different scenarios, their performance may differ. An obvious way to determine the performance is by visually comparing the object and trajectory. But a more convincing way is to set the ground truth of the object and compute the precision statistically.

In this article, we introduce a self-built gesture recognition system with MATLAB programming, and implement Kalman, Extended Kalman, Unscented Kalman and Particles filters with different parameters for object tracking and gesture recognition. We investigate their performance statistically under different scenarios, including linear motion, nonlinear motion, motion with object being occluded and circumstance in low light.

Identifying an object in a real-world environment is of great difference from tracking the detection of the object. In this article we mostly focus on the tracking of already identified objects rather than the method of how to detect the object. The structure of the paper consists of 4 parts: first we provide a brief description of the gesture recognition system we build, followed by the theory introduction of the filters we applied in the system. Next, we provide the experiments that we implemented for the comparison of the filters' performance. Finally, we draw conclusion based on the results of the experiments.

2. Gesture Recognition System

We built our gesture recognition system which could recognize number gesture from one to five and show the corresponding numbers. This system is based on Matlab, which is one of the most popular computer vision platforms. The block diagram of the system is shown in Figure 1. This system could be separate into 3 parts: detection, tracking and recognition.

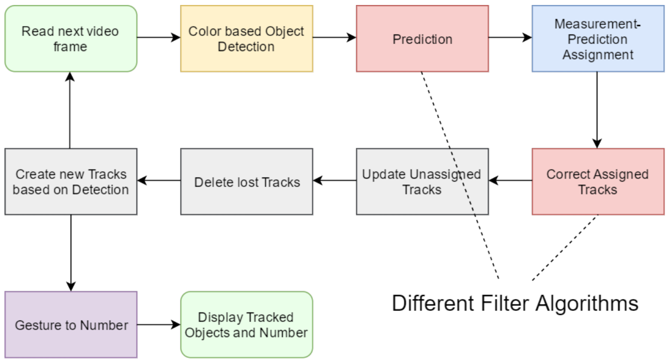
Read next video frame
Color based Object Detection
Prediction
Measurement-Prediction Assignment
Create new Tracks based on Detection
Delete lost Tracks
Update Unassigned Tracks
Correct Assigned Tracks
Gesture to Number
Display Tracked Objects and Number
Different Filter Algorithms

Figure 1. Diagram of Gesture Recognition System

2.1. Object Detection

We have tested 3 detection methods including correlation bases detection [4], foreground detection [5] and color-based detection [6]. Since the correlation-based object detection needs a specific template for each object and is unstable when object or lens rotates, it appeared low precision for finger detection where lots of rotations and random motions happen. Similarly, the foreground detection has a demand for fixed camera, and is hard to tell the fingers from non-target moving objects interference. Color-based detection is widely used in scenes where the objects have specific color, and has the advantage of robustness for moving object detection compared with other two methods. Using a glove with specific colored fingers, the color-based detection showed high precision and robustness. So we choose the color-based object detection to capture the fingers and hand locations. In detail, in each frame, the RGB value of each pixel $< r_x, g_x, b_x >$ is compared with the target RGB value $< r_t, g_t, b_t >$, and the Euclidean norm of the difference is calculated as the distance:

$$dist_x = ||r_x - r_t|| + ||g_x - g_t|| + ||b_x - b_t|| \qquad (1)$$

If the distance is within a given threshold $dist_{th}$, this pixel is judged as target (foreground).

$$target_x = \begin{cases} 1, & dist_x < dist_{th} \\ 0, & dist_x > dist_{th} \end{cases} \qquad (2)$$

A mask of target is formed after all pixels have been judged. A blob analysis is performed on the mask and the finger-like connected components within given size will be output as the fingers and the centers are calculated as locations. Before blob analysis, we remove small connected components within a given area threshold (e.g. 100 pixels) to prevent false positive, then a morphologic closing is applied to remove the false negative holes caused by noise.

2.2. Tracking

We applied 4 kinds of filters which will be discussed in section 3. For multiple object tracking, one challenge is to correctly assign the predicted locations with the detected ones. Here we use the state-space distance [7] to compute the confidence.

$$d(z) = (z - Hx)^T \sum^{-1} (z - Hx) + ln|\Sigma| \qquad (3)$$

$$\Sigma = HPH^T + R \qquad (4)$$

In the above equations, z is the measurement and $d(z)$ is the corresponding state-space distance, x is the predicted state, P is the state estimation error covariance, H is the measurement model, where all these metrics will be discussed in section 3. For each predicted state, we assign it to the measurement with lowest distance. For Kalman filters the assigned trackers will be corrected to get the tracked locations, while the unassigned trackers will just use predicted states as the tracked locations. For trackers which are unassigned for a certain number of continuous frames, we regard them as lost objects and delete them. For unassigned measurements new trackers will be created once they show up longer than a certain number of continuous frames.

2.3. Convert Gesture to Number

Once we have the tracked locations of the fingers and hand, we establish a 3x3 grid block based on hand location and count the finger numbers in each block, then output the corresponding meaning of the gesture, here the output contains numbers from 1-5. The connection between number count and the output is set manually, and is adjustable to add other meanings of gestures.

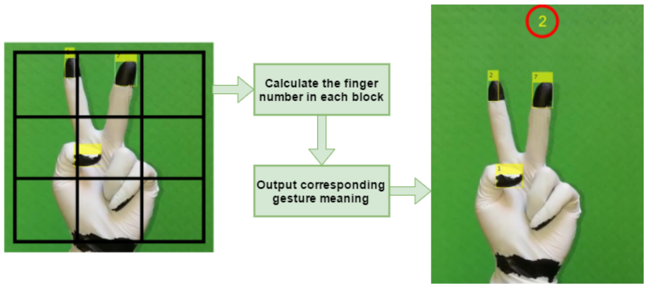
Calculate the finger number in each block
Output corresponding gesture meaning
2

Figure 2. Gesture-to-number Conversion

3. Filter Theory

3.1. Kalman Filter

Kalman filter has been widely applied in object tracking. The Kalman filter is the optimal filter to estimate the state of a linear dynamic system when only noisy measurements are available [8]. Due to analytic tractability both operation and measurement noises are required to be Gaussian. In 1960, Ruodolf Kalman first introduced the system [9], and it was first applied in estimating the trajectory for the NASA Apollo program in 1969 [10].

The operation of the filter basically consists of 2 steps:

1. Prediction, based on the last estimated state $\hat{x}_{k-1}$ and covariance matrix P_{k-1}:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{5}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{6}$$

where A is the time-transition matrix, u_k is the control or external input, B is a linear matrix which characterize the effects of the external input, and Q is the operation noise covariance matrix.

Equation (5) predicts the most probable state $\hat{x}_k^-$ based on the last estimation and the external input. Equation (6) predicts the new covariance matrix P_k^- of the predicted state, given the previous estimated covariance matrix and operation noise.

2. Correction, based on the measurements to correct the prediction:

$$K_k = P_k^- H^T(HP_k^- H^T + R)^{-1} \tag{7}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{8}$$

$$P_k = (I - K_k H)P_k^- \tag{9}$$

where K_k is the Kalman gain, the term $z_k - H\hat{x}_k^-$ is called the innovation, H is the measurement matrix, and R is the measurement noise covariance matrix.

The Kalman gain K_k controls the effect that the current measurement has over the new state. Equation (8) and (9) correct the prediction based on the Kalman gain to estimate the state $\hat{x}_k$ and covariance matrix P_k.

It should be noted that the transition matrix A and measurement matrix H in Kalman filter are certain, so when the model is non-linear, no such certain matrices A or H exists, and Kalman filter no longer works.

3.2. Extended Kalman Filter

The Extended Kalman filter (EKF) [11] [12] has been applied to alleviate the constraint of linearity. The basic framework of Extended Kalman filter is similar to that of Kalman filter, which is shown in Figure 3.

For convenience, the control input u_k has been set to zero. The difference of the Extended Kalman filter are to use the nonlinear functions f and h for state and measurement prediction, and to substitute the transition matrix A

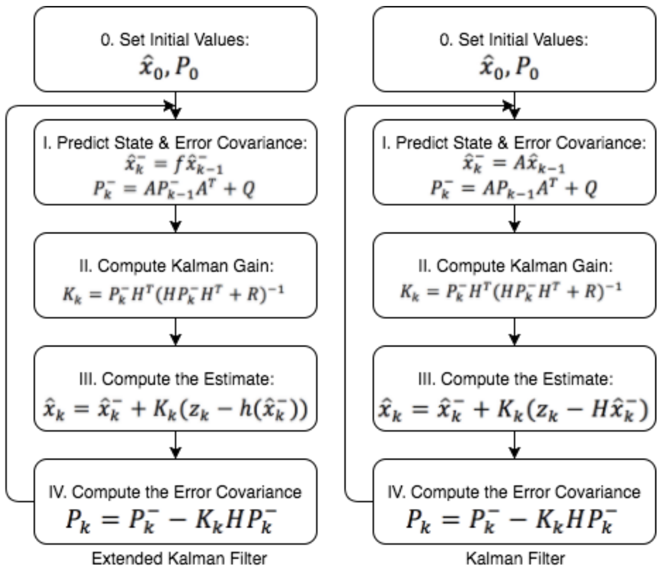
0. Set Initial Values:
$\hat{x}_0, P_0$
I. Predict State & Error Covariance:
$\hat{x}_k^- = f\hat{x}_{k-1}^-$
$P_k^- = AP_{k-1}^- A^T + Q$
II. Compute Kalman Gain:
$K_k = P_k^- H^T(HP_k^- H^T + R)^{-1}$
III. Compute the Estimate:
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-))$
IV. Compute the Error Covariance
$P_k = P_k^- - K_k HP_k^-$
Extended Kalman Filter
0. Set Initial Values:
$\hat{x}_0, P_0$
I. Predict State & Error Covariance:
$\hat{x}_k^- = A\hat{x}_{k-1}$
$P_k^- = AP_{k-1}A^T + Q$
II. Compute Kalman Gain:
$K_k = P_k^- H^T(HP_k^- H^T + R)^{-1}$
III. Compute the Estimate:
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
IV. Compute the Error Covariance
$P_k = P_k^- - K_k HP_k^-$
Kalman Filter

Figure 3. Diagram Comparison between EKF and KF

and measurement matrix H with Jacobian matrices derived from the non-linear functions:

$$A = \frac{\partial f}{\partial x}|_{\hat{x}_{k-1}} \tag{10}$$

$$H = \frac{\partial h}{\partial x}|_{\hat{x}_k^-} \tag{11}$$

The Jacobian matrix is obtained through the first order partial derivative, so the Extended Kalman Filter can be viewed as "first order" approximation to the optimal terms[6]. However, the Extended Kalman filter can introduce large errors when the high order terms cannot be overlooked. As shown in Figure 4 [13], linearized transformation is not reliable if propagation cannot be well approximated by a linear function [14]. In addition, linearization can be applied only if the Jacobian matrices exists. Furthermore, the heavy computation of Jacobian matrices makes EKF impractical in many situations.

3.3. Unscented Kalman Filter

The Unscented Kalman Filter [15] addresses the approximation issues of the Extended Kalman Filter. The intuition is to approximate a probability distribution rather than a non-linear function. To achieve this, a set of points are carefully chosen to described the mean and covariance, and the points are called the sigma points [16].

The Unscented Kalman Filter can be seen as a combination of Unscented Transformation and Kalman Filter. We first start by explaining the unscented transformation [17].

The unscented transformation(UT) is a method to compute the statistics of a random variable undergoing nonlinear transformation by transforming each sigma point

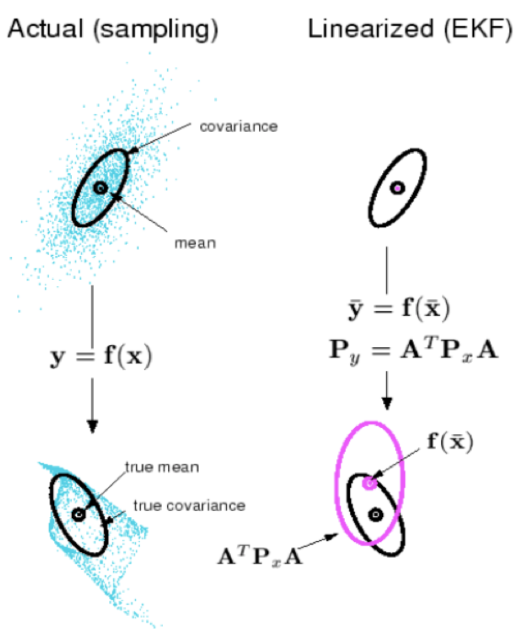
Actual (sampling)
Linearized (EKF)
covariance
mean
$\mathbf{y} = \mathbf{f}(\mathbf{x})$
$\bar{\mathbf{y}} = \mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{P}_y = \mathbf{A}^T\mathbf{P}_x\mathbf{A}$
true mean
true covariance
$\mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{A}^T\mathbf{P}_x\mathbf{A}$

Figure 4. Mean and Covariance Propagation

through the nonlinear function. Suppose a random variable x (dimension n) with mean μ and covariance matrix $\sum$ is propagated through a nonlinear model with function $y = g(x)$. A matrix X of $2n+1$ vectors X_i is formed as the set of sigma points, and each vector is corresponding with weight W_i. The procedures are according to the following:

Choosing the sigma points

$$X^{[0]} = \mu \tag{12}$$

$$X^{[i]} = \mu + (\sqrt{(n+\lambda)})_i \quad for\ i = 1, ..., n \tag{13}$$

$$X^{[i]} = \mu + (\sqrt{(n+\lambda)})_{i-n} \quad for\ i = n+1, ..., 2n \tag{14}$$

Weights sigma points

$$w_m^{[0]} = \frac{\lambda}{n+\lambda} \tag{15}$$

$$w_c^{[0]} = w_m^{[0]} + (1 - \alpha^2 + \beta) \tag{16}$$

$$w_m^{[i]} = w_c^{[i]} = \frac{1}{2(n+\lambda)} \quad for\ i = 1, ..., 2n \tag{17}$$

where $\lambda = \alpha^2(n+k) - n$ is a scaling parameter. α is usually set to small value and k is usually set to zero, and both of them determine how far the sigma points are away

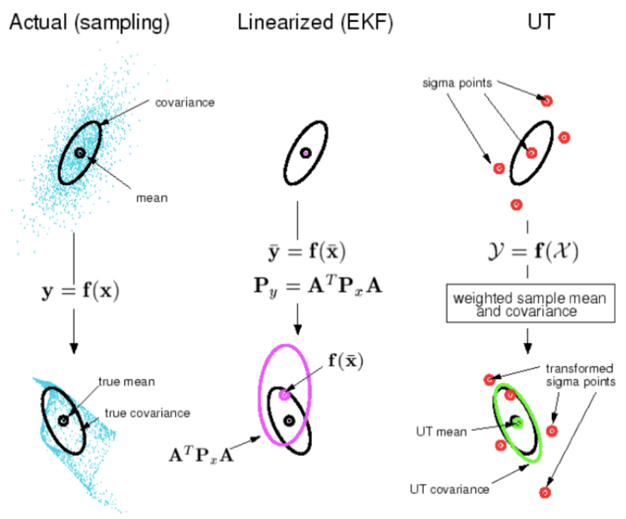
Actual (sampling)
Linearized (EKF)
UT
covariance
mean
sigma points
$\mathbf{y} = \mathbf{f}(\mathbf{x})$
$\bar{\mathbf{y}} = \mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{P}_y = \mathbf{A}^T\mathbf{P}_x\mathbf{A}$
$\mathcal{Y} = \mathbf{f}(\mathcal{X})$
weighted sample mean and covariance
true mean
true covariance
$\mathbf{f}(\bar{\mathbf{x}})$
$\mathbf{A}^T\mathbf{P}_x\mathbf{A}$
transformed sigma points
UT mean
UT covariance

Figure 5. Mean and Covariance Propagation

from the mean. β is used to incorporate prior knowledge of the distribution of x, where $\beta = 2$ is the optimal choice for Gaussians. $(\sqrt{(n+\lambda)})_i$ is the i th row of the matrix square root. These sigma vectors are propagated through the nonlinear model,

$$Y_i = g(X_i) \quad for\ i = 0, ..., 2n \tag{18}$$

The mean and covariance of y can be approximated by the summation of the weighted sample mean and covariance of the transformed sigma points,

$$\mu' = \sum_{i=0}^{2n} w_m^{[i]} g(X^{[i]}) \tag{19}$$

$$\sum{}' = \sum_{i=0}^{2n} w_c^{[i]} (g(X^{[i]}) - \mu')(g(X^{[i]}) - \mu')^T \tag{20}$$

These sigma points completely capture the true mean and covariance accurately to the 3rd order for any nonlinearity. As shown in Figure 5 [13], the right plots show the performance of the UT when using 5 sigma points, which outperforms the other two methods obviously.

Therefore, when the mean and covariance of the input are certain, the mean and covariance of the non-linear transformation can be computed using UT. The algorithm of Unscented Kalman Filter is shown in Algorithm 1: Unscented Kalman Filter Algorithm. The basic framework is the same as the Extended Kalman Filter. The difference is that, instead of using the nonlinear functions directly for prediction and the Jacobian matrices to linearize the model, the Unscented Kalman Filter applies the UT to obtain the mean and covariance of the nonlinear propagation [18], so it does not require the computation of the Jacobian matrices.

Algorithm 1 Unscented Kalman Filter Algorithm

```
1: Unscented Kalman Filter($\mu_{t-1}, \quad \Sigma_{t-1}, \quad u_t, \quad z_t$)
2: $\chi_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \sqrt{(n+\lambda)\Sigma_{t-1}} \quad \mu_{t-1} - \sqrt{(n+\lambda)\Sigma_{t-1}})$
3: $\bar{\chi}_t^* = g(u_t, \chi_{t-1})$
4: $\bar{\mu}_t = \sum_{i=0}^{2n} \omega_m^{[i]} \bar{\chi}_t^{*[i]}$
5: $\bar{\Sigma}_t = \sum_{i=0}^{2n} \omega_c^{[i]} (\bar{\chi}_t^{*[i]} - \bar{\mu}_t)(\bar{\chi}_t^{*[i]} - \bar{\mu}_t)^T + R_t$
6: $\bar{\chi}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{(n+\lambda)\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{(n+\lambda)\bar{\Sigma}_t})$
7: $\bar{Z}_t = h(\bar{\chi}_t)$
8: $\hat{z}_t = \sum_{i=0}^{2n} \omega_m^{[i]} \bar{Z}_t^{[i]}$
9: $S_t = \sum_{i=0}^{2n} \omega_c^{[i]} (\bar{Z}_t^{[i]} - \hat{z}_t)(\bar{Z}_t^{[i]} - \hat{z}_t)^T + Q_t$
10: $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} \omega_c^{[i]} (\bar{\chi}_t^{[i]} - \bar{\mu}_t)(\bar{Z}_t^{[i]} - \hat{z}_t)^T$
11: $K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$
12: $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$
13: $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$
14: $return \quad \mu_t, \Sigma_t$
```

Though no Jacobian matrices are needed, the Unscented Kalman Filter belongs to the same complexity class with the Extended Kalman Filter. Besides, the Unscented Kalman Filter is still restricted to Gaussian distributions.

3.4. Particle Filter

Another popular tool in object tracking is the Particle Filter [19], which is also known as sequential Monte Carlo method [20]. The Particle Filter method is an efficient method to apply simulation for estimating unknown probability distributions, and it is suitable in all kinds of frameworks, regardless of linearity and Gaussian.

The basic idea of the algorithm is that any distribution can be represented as a set of weighted particle pairs $< x_k^{(i)}, w_k^{(i)} >$, where $x_k^{(i)}$ is a possible value of the system state and $w_k^{(i)}$ is its plausibility. Figure 6 indicates the diagram of particle filter. In particle filtering a set of particle is first generated randomly to represent the initial distribution. The posterior distribution is defined by a new particle set and it's determined by a function f, which describes the dynamics of the system. The measurement update step is achieved by setting the weights according to the measurement data. The relation between the measurement data and the latent states is given by the conditional h. The most probably estimated state will be:

$$x_k = \sum_{i=1} x_k^{-(i)} w_k^{(i)} \tag{21}$$

It frequently occurs that almost all weights are zero except for one which tends to be one when implementing the Particle Filter in its basic form. To eliminate the effect, the resampling [21] is very important. The distribution can be updated as the particles be resampled with the weights

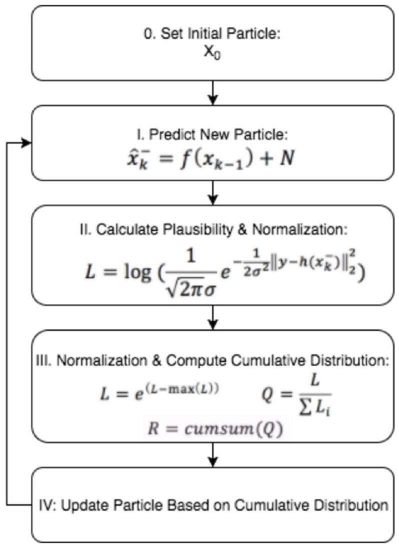
0. Set Initial Particle:
X_0
I. Predict New Particle:
$\hat{x}_k^- = f(x_{k-1}) + N$
II. Calculate Plausibility & Normalization:
$L = \log(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}\|y-h(x_k^-)\|_2^2})$
III. Normalization & Compute Cumulative Distribution:
$L = e^{(L-\max(L))}$ $Q = \frac{L}{\sum L_i}$
$R = cumsum(Q)$
IV: Update Particle Based on Cumulative Distribution

Figure 6. Diagram of Particle Filter

which we obtain from the measurement data, as shown in Figure 7.

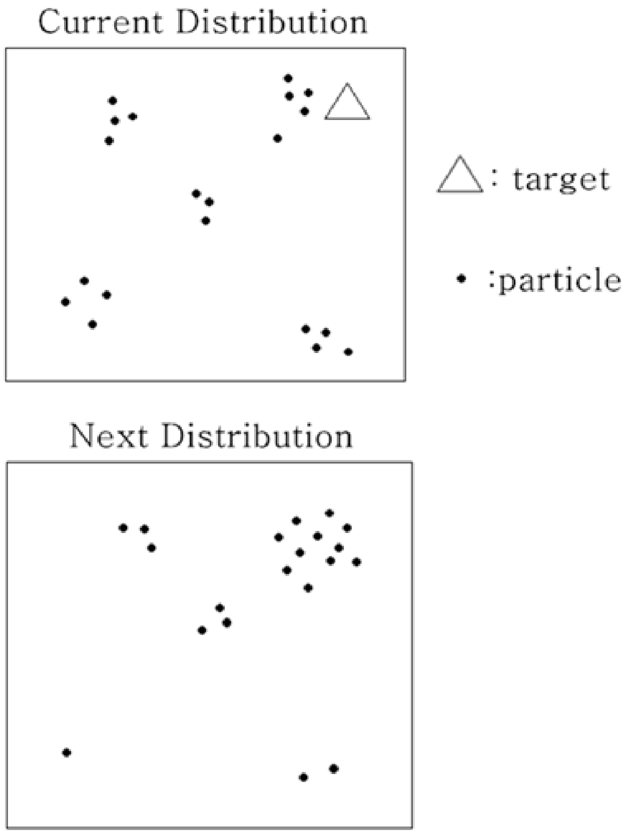

Figure 7. Particles Resampling

When used in object tracking, the Particle Filter is usually applied with color detection and it is relatively easy to implement. However, since the information of a pixel is just RBG information, it cannot work for gray-scale video, and its precision decreases sharply for low-light and occluded objects. Another issue is the decision of the sample size for adequate performance. An efficient set size has been introduced by Kong and Liu [21], which can be obtained from the weights of the particles following the equation below:

$$N_{eff} = \frac{N_s}{1 + var(w_k)} \tag{22}$$

where N_s is the original set size. Notice that $N_{eff} \leq N_s$, and small N_{eff} indicates severe degeneracy, which is an undesirable effect in particle filters. A direct way to reduce this effect is to use very large N_s. So sufficient amounts of particles are prerequisite to ensure accurate tracking, and it can require high computation load for large-size video.

4. Experiment Design and Results

To investigate the performances of different filters and confirm the hypothesis, we use the detection and tracking parts of our system to do a quantitative evaluation experiment, which aims to compare the tracking using different filters in various scenarios.

4.1. Scenario Selection

Considering factors which could affect the tracking performances, we design and record videos in scenarios shown below: 1) Linear motion model: the objects perform simple linear motion, e.g. straight line motion. Note that there is no perfect linear motion in real life, so the linear motion is actually approximately linear motion. Despite small vibration or disturbance, the movement in a short period of continuous frame could be regarded as linear. 2) Nonlinear motion model: the objects perform mostly random motion, e.g. hand doing random gestures. It?s hard or impossible to manually calculate the state transition model in this case. 3) Occluded objects: objects are usually occluded in real life videos. A robust tracker should be able to handle the occlusion. In this kind of videos, objects are occluded partially or entirely in some frames, the occluded frame ratio varies from 10% to 50%. 4) Low-light condition: light is a very important factor that could influence the detection results. In this condition, objects moves in low-light environment which makes it hard to tell the target color for each pixel. By choosing one of 2 motion models and one of 3 interferences (no interference is one choice), we have 6 combinations of scenarios.

4.2. Evaluation Metric

Many metrics for quantitative evaluation of target tracking have been proposed by researchers [22] [23] [24]. Since we focus on the tracking performance comparison, the precision plot which is used in recent year tracking benchmarks [25] [26] is a suitable choice. The precision of a tracker means the percentage of frames whose estimated location is within the given threshold distance of the ground truth [8]. We are able to see the tracking performances with different error tolerances from precision plot directly.

4.3. Parameters

The system parameters are shown in Table 1, for different trackers and videos, the noise value is selected based on the scenario. When the detector works well, i.e. there is enough light and no occlusion, the measurement noise variance is set to be low, while in low-light or occluded conditions, the measurement noise variance is set to be high. The motion noise is based on the goodness-of-fit of transition matrix/function to the motion model, for Kalman filter and Particle filter, motion noise variance is set to be high for complex motion, and low for simple motion.

Table 1. System Parameters

System Parameter	**Value**
Motion Noise	10 to 1000
Measurement Noise	10 to 10,000
Video(Frame) Size	486x864
Frame Per Second	12

The filter parameters are shown in Table 2. As what has been discussed, the performance of three kinds of Kalman filters depends on the correct transition matrix/function, but for most non-linear motion, there is no general model to calculate transition function for EKF and UKF. To estimate the transition function of the hidden motion model, we assume the state of the target is approximately an autoregressive process:

$$x_k \approx f(x_{k-1}, w) + v_k \tag{23}$$

Using labelled ground truth, a feedforward neural network is applied to get the estimation of the transition function f(parameterized by w). The residual error after convergence was taken to be the motion noise. We apply particle filters with 3 amounts of particles with linear transition function.

4.4. Result Analysis

Figure 8 shows the precision of trackers in linear motion scenario. All three kinds of Kalman filters have good performance, where the trival difference among them is because the motion model is not perfectly linear, and EKF and UKF could better capture the little vibration. When the threshold becomes larger, there is no obvious difference between KF, EKF and UKF. Particle filters with 5000 and

Table 2. Filter Parameters

Filter	Parameter
Kalman Filter	Constant acceleration model
Extened Kalman Filter	f approximated by single-layer FNN
Unscented Kalman Filter	f approximated by single-layer FNN, $\alpha = 0.001, \beta = 2, k = 0$
Particle Filter-50	Particle Number = 50
Particle Filter-500	Particle Number = 500
Particle Filter-5000	Particle Number = 5000

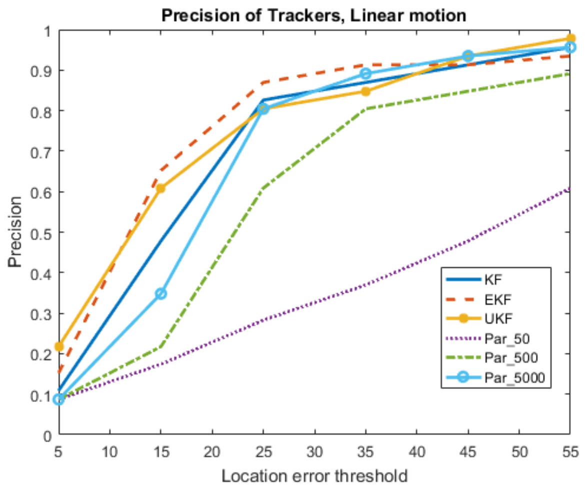

Figure 8. Precision of Trackers in Linear Motion Scenario

500 particles has also good and acceptable performances. But particle filters with 50 particles has relatively poor precision, because it's hard to compute the accurate location using such few number of particles, where even little noise would sufficiently influence the result.

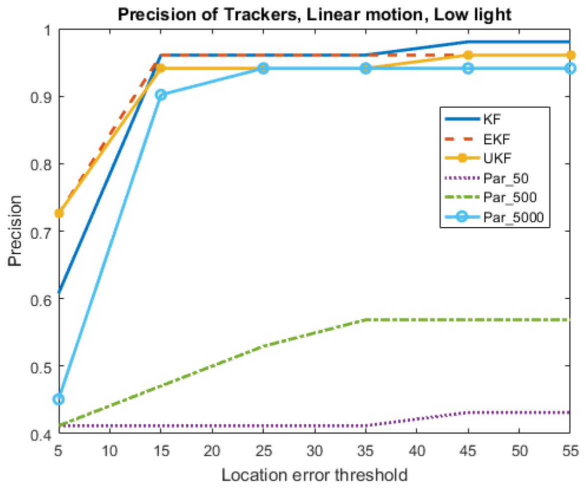

Figure 9. Precision of Trackers in Linear Motion and Low Light Scenario

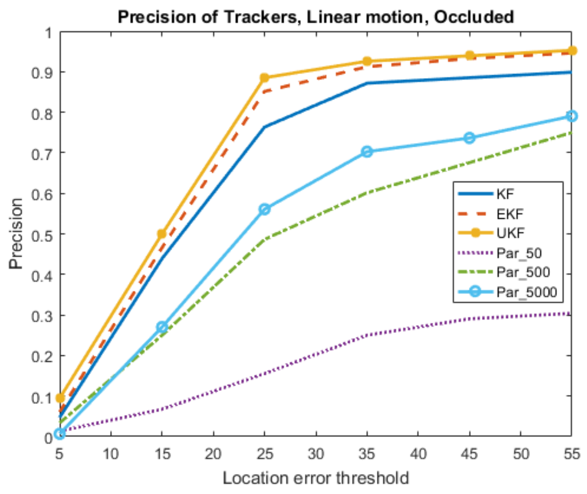

Figure 10. Precision of Trackers in Linear Motion and Occluded Scenario

Figure 9 shows the results in low light scenario. We can see that the 3 Kalman filters and particle filter with 5000 particles are still robust with good precision. But the particle filters with 500 particles and 50 particles have poor performance. This is because in low light condition, it's more difficult to tell the accurate difference of color, i.e. the particle filter without enough particles have low accuracy when calculating the location.

From Figure 10 we can see that UKF and EKF outperform KF as a result of more accurate prediction in occluded frames. While the 3 particle filters have relatively low precision. This is because when object is occluded particle filter will try to find the most similar region to target in background, i.e. the result is dominated by noise.

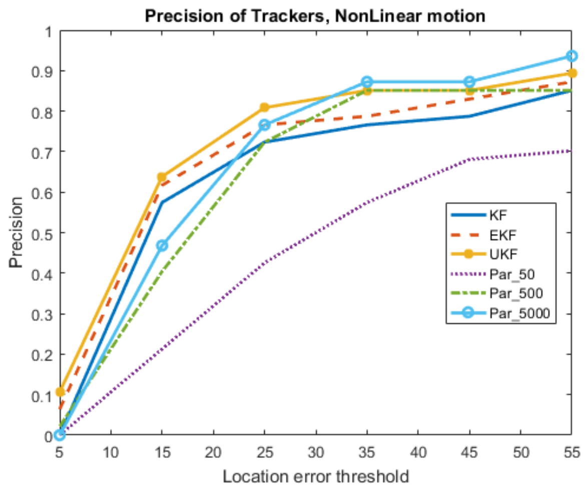

Figure 11. Precision of Trackers in Nonlinear Motion Scenario

Figure 11 shows the results in nonlinear motion scenario, we can see that clearly UKF>EKF>KF, because UKF has the most accurate approximation to the nonlinear model,

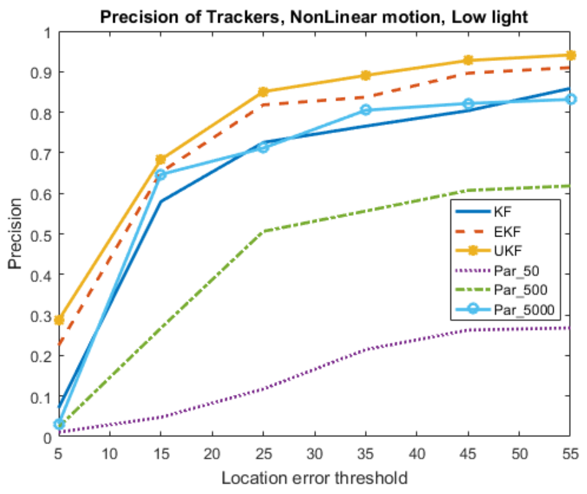

Figure 12. Precision of Trackers in Nonlinear Motion and Low Light Scenario

which confirms the hypothesis. On the other hand, particle filters with 5000 and 500 particles both have good performance.

Figure 12 shows the results in nonlinear motion and low light scenario. Compared with Fig. 6, we can see that three Kalman filters are still robust while UKF>EKF>KF. Particle filter with 5000 particles is also of good precision, even though influenced by low light inference. However, the particle filters with 500 and 50 particles have low precision because of the low light interference.

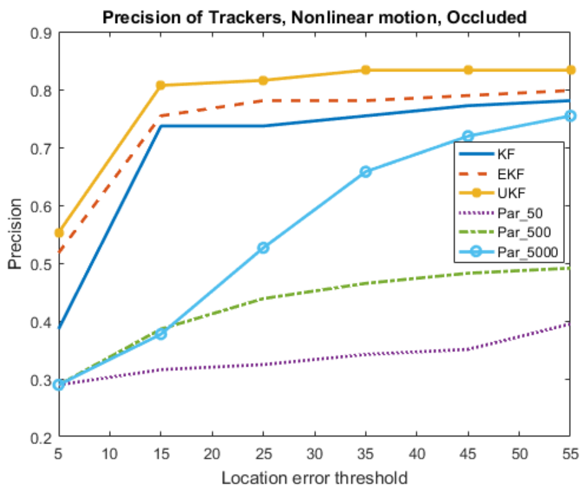

Figure 13. Precision of Trackers in Nonlinear Motion and Occluded Scenario

From Figure 13, we can see that in occluded nonlinear motion scenario, it is obvious that Kalman filters are more robust than particle filters, while UKF>EKF>KF. Particle filters again have low precision because of occlusion.

5. Conclusion

For linear motion scenario, all three Kalman filters perform well; for nonlinear motion, UKF>EFK>KF corresponding to the different precisions of approximation to the model. And all three kinds of Kalman filters have good stability in low light and occluded conditions. Particle filter needs sufficient amount of particles to ensure robustness, and it has good performance for both linear and nonlinear motion model. But once in low light or occluded conditions, the performance of particle filter declines sharply. For future work, there are many ways to improve the track precision. For example, using RGB-D camera, it is possible to reconstruct the 3-D model of target and get better tracking results [27].

References

[1] S. Escalera, V. Athitsos, and I. Guyon, "Challenges in multimodal gesture recognition," *Journal of Machine Learning Research 17*, 2016.

[2] E. Ohn-Bars and M. M. Trivedi, "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations," *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, vol. 15, no. 6, 2014.

[3] O. Korkalo and S. Kahn, "Real-time depth camera tracking with cad models and icp," *Journal of Virtual Reality and Broadcasting*, vol. 13, no. 1, 2016.

[4] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[5] J. Gallego Vila, "Foreground segmentation and tracking based on foreground and background modeling techniques," Master's thesis, Universitat Politècnica de Catalunya, 2009.

[6] S. Khan and M. Shah, "Object based segmentation of video using color, motion and spatial information," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 2. IEEE, 2001, pp. II–II.

[7] G. Welch and G. Bishop, "An introduction to the kalman filter," 1995.

[8] L. C. Botond A. Bocsi, "Visual tracking with filtering algorithms," in *2008 4th International Conference on Intelligent Computer Communication and Processing*, 2008, pp. 269–274.

[9] R. E. Kalman, *A new approach to linear filtering and prediction problems:Transactions of the ASME–Journal of Basic Engineering*, ser. 82(Series D):35–45, 1960.

[10] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB*, S. edition, Ed. John Wilney and Sons, Inc., 2001.

[11] P. S. Maybeck, *Stochastic Models, Estimation, and Control*. Academic Press, Inc., 1979, vol. vol. 141 of Mathematics in Science and Engineering.

[12] E. . W. Sorenson, "Kalman filtering: Theory and application," *Piscataway, NJ: IEEE*, 1985.

[13] C. Stachniss, "Unscented kalman filter."

[14] G. F. Ye Yuan and W. Zhang, "Extended and unscented kalman filters for parameter estimation of a hydrodynamic model of vessel," in *2016 35th Chine Control Conference(CCC)*, 2016, pp. 2051–2056.

[15] S. . J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *In Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls.*, 1997.

[16] R. Turner and C. E. Rasmussen, "Model based learning of sigma points in unscented kalman filtering," in *Machine Learning for Signal Processing(MLSP)*. IEEE International Workshop, 29 Aug – Sept 2010.

[17] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," in *Preceedings of the IEEE*, vol. 92, no. 3, Mar 2004, pp. 401–422.

[18] F. Gustafsson and G. Hendeby, "Some relations between extended and unscented kalman filters," in *IEEE Transactions on Signal Processing*, vol. 60, Feb 2012, pp. 545–555.

[19] N. Gordon, J. Salmond, and A. Smith, "A novel approach to non-linear/non-gaussian bayesian state estimation," *IEEE Preceedings on Radar and Signap Processing*, pp. 107–113, 1993.

[20] C. P. Robert and G. Casella, *Monte Carlo Methods*, S. edition, Ed. Springer, 2004.

[21] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2005.

[22] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 1, pp. 1–10, 2008.

[23] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybridboosted multi-target tracker for crowded scene," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2953–2960.

[24] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1619–1632, 2011.

[25] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[26] ——, "Object tracking benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1834–1848, 2015.

[27] R. Y. Takimoto, M. d. S. G. Tsuzuki, R. Vogelaar, T. de Castro Martins, A. K. Sato, Y. Iwao, T. Gotoh, and S. Kagei, "3d reconstruction and multiple point cloud registration using a low precision rgb-d sensor," *Mechatronics*, vol. 35, pp. 11–22, 2016.