Bayesian Filter based Object Traction in Computer Vision

Liang Xu
Electrical and Computer Enineering
virginia commonwealth university

***Abstract*—As most widely used in cell phone and other areas, camera be come the most widely used sensor. Also, there are lots of information contains in one or a serial of picture. This project will focus on Bayesian based tracking in computer vision. For tracking, there are two areas, one is tracking the object in a serial image, another areas is tracking the camera's location. The first case most called object tracking, and the second is called Simultaneous Localization and Mapping(SLAM).**

Particle filter based tracking is widely used in computer vision field. Tracking is the problem of generating an inference about the motion of an object given a sequence of images. Generally, we will have some measurements that appear at each tick of clock. These measurements could be the position of some image points, the position and moments of some image regions, or pretty much anything else. These are not guaranteed to be relevant, in the sense that some could come from other objects or from noise. We will have an encoding of the object's state and some model of how this state change from thick to thick. We would like to infer the state of the world from the measurement and the model of dynamics.

The past decades have been significant progress in robot navigation and Simultaneous Localization and Mapping(SLAM). SLAM is the computational problem of constructing a map of an unknown environment on the fly while simultaneously keeping track of a moving object with a sensor on it. It has been widely used in the robotic community. Different sensors can contribute their own knowledge with estimating the environment.

Accelerator, IMU, GPS and many another sensors could be used for SLAM. However, many surprising reasons vision is an attractive choice for SLAM sensor: cameras are compact, accurate, informative, cheap and ubiquitous. Vision also is the primary navigation tool for human and animals. So this project will use the Extended Kalmen Filter(EKF) knowledge learned from the class and Computer Vision to build a real-time visual SLAM system.

I. Introduction

Tracking problems are of grate practical importance. There are very good reasons to want to, track aircraft using radar returns. Motions capture, if we can track the 3D configuration of a moving person accurately, the we can make an accurate record of their motions. Once we have this record, then we can use it to drive a rendering process. Recognition from motion: the motion of objects is quite characteristic. We might be able to determine the identity of the object from its motion. Tracking is not only for track the object in the image or video, it also include to track the camera's location. The Visual-SLAM is the area for this topic.

The real-time V-SLAM have two parts, one is the computer vision and another is the parameter estimation. The computer vision can give a better knowledge of the image get by the camera, and parameter estimation can help us to estimate the current position.

So, how to estimate the current point by just using the image? First, we are assuming the camera is moving, and it is a dynamic movement. This means the movement is a continuous function, and follow the Newton's Law. Second, there are Natural Visual Landmarks on each image. This marks could be some special points or corners in the image. There is the certain case, we can not get any feature points from the image, but we are not going to discuss that case. According to the first assumption, the object movement is a continuous function, so there are strong relation between the images. So we can use the position different between to image to estimate the position of the camera. This the whole idea of this project.

II. Particle Filter based Object Tracking

In this section will focus on how to track the object in the image or video. In this section, I am going to implement tracking method for image sequence and videos. The main algorithm used in this section is particle filter.

A. Particle Filter Algorithm

Below is the particle filter procedure.

- Object is the thing that is actually being tracked.
- x(t) represent the state of the model at time t.
- A dynamic model p(x(t)|x(t-1)) distribution of the state at time t given the state at t - 1.
- A measurement z(t) that somehow captures the data of the current image.
- A sensor model p(z(t)|x(t)) that gives the likelihood of a measurement given the state. For Bayesian-based tracking, we assume that tome t -1 we have a believe about the state. For represented as a density p(x(t-1)), and that given some measurements z(t) at time t we update our Belief by computing the following equations Figure 1.

$$Bel(x_t) \propto p(z_t|x_t)p(x_t|u_t, x_{t-1})Bel(x_{t-1})$$

Fig. 1: Belief Updating

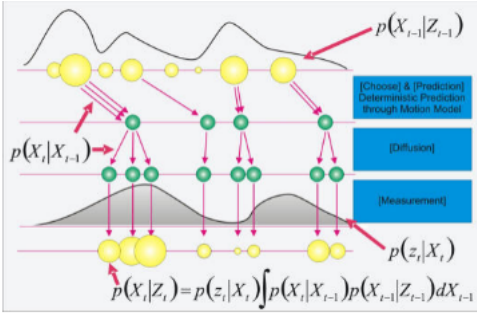

Fig. 2: Particle Filter Updating

In this project, the problem will track an image patch template taken from the first frame of the video. For this project, the object is simply going to be the image patch, and the state will be only the 2D center location of the patch. Thus each particle will be a (u,v) pixel location(where u and v are the row and column number respective) representing a proposed location for the center of the template window equal size in the current image, the Mean Squared Error as Figure 3. The funny indexing is just because (u, v) are indexed from 1 to M (or N) but the state <u,v> is the location of the center. MSE, of course, only indicates how dissimilar the image patch is, whereas we need a similarity measure so that more similar patches are more likely. Thus, we will use a squared exponential equation (Gaussian) to convert this into a usable measurement function Figure 4:

$$MSE(u_p, v_p) = \frac{1}{mn}\sum_{u=1}^{m}\sum_{v=1}^{n}(Template(u,v) - Image^t(u+u_p-m/2, v+v_p-n/2))^2$$

Fig. 3: Mean Square Error for Template

$$p(z_t|x_t) \propto \exp(-\frac{MSE}{2\sigma^2_{MSE}})$$

Fig. 4: Measurement Function

B. Basic Particle Filter

In this problem, we implement a basic particle filter to a simple object Figure 5. Figure 6 is the first picture. The object is doing a nonlinear motion. When the object touch the boundary, it turns the direction. So the particle's variance increased after touch the right boundary, like Figure 8

Fig. 5: Tracking Object

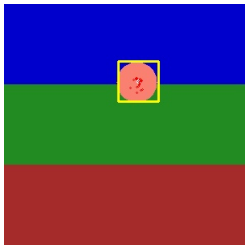

Fig. 6: First Image 000

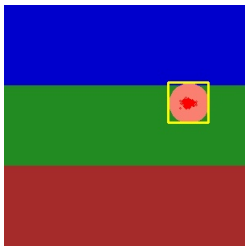

Fig. 7: First Image 015

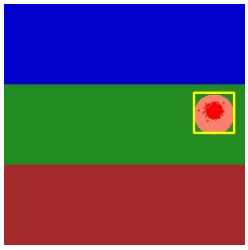

Fig. 8: First Image 026

C. Changes in Appearance

in the last section we were working with a static template, assuming that the target will not change in shape. This can be addressed by updating the model's appearance over time. In this case, we need include a step to use the history to update the tracking window model. We can accomplish this using an Infinite Impulse Response(IIR) filter. The concept is simple: we first find the best tracking window for the current particle distribution as displayed in the visualizations. Then we just update the current window model to be a weighted sum of the last model and the current best estimate.

$$Template = \alpha Best(t) + (1-\alpha)Template(t-1)$$

where Best(t) is the patch of the best estimate or mean estimate. It's easy to see that by recursively updating this sum, the window implements an exponentially decaying weighted sum of (all) the past windows.
For this part, we are going to track governor Romney's hand. Because his hand is moving, so each particle will 4 states which are u, v, u speed, and v speed. Here are the result Figure 9,10,11,12. Figure 9 is the initial update, we can easily check all the particle are every where across the whole image. And we can check the hand is change the shape as well. As the

hand change too quickly, the particle have a hard time to catch the movement, as Figure 12, the particle's variance increased.

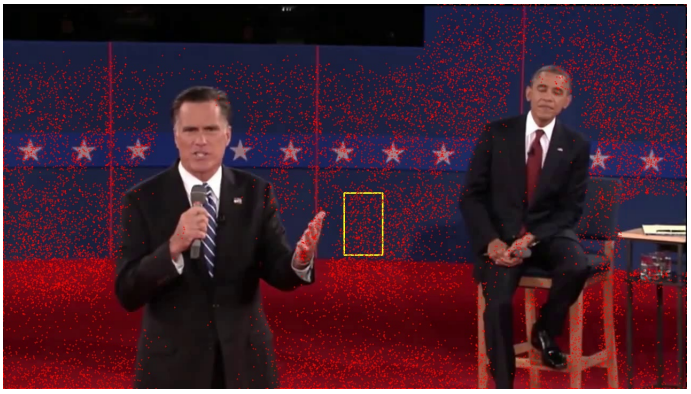

Fig. 9: P2_000

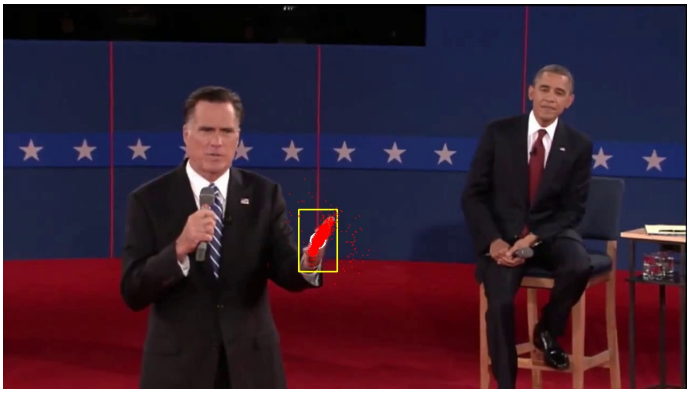

Fig. 10: P2_006

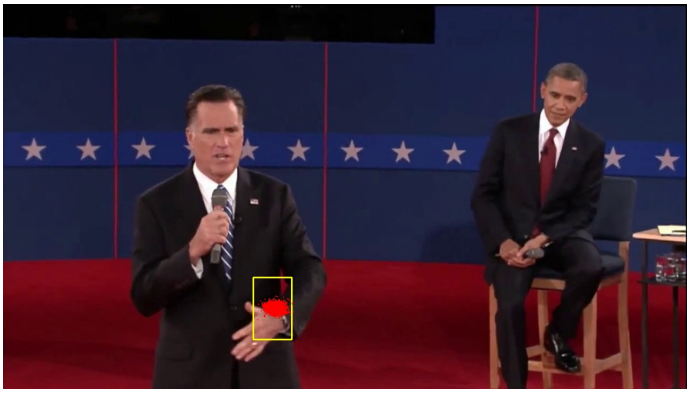

Fig. 11: P2_022

Fig. 12: P2_164

D. Occlusions

For this part we will work with a much more difficult video to perform tracking with, we would like to be able to track the blond-haired woman (the one with the white jacket) as she crosses the road. If you try applying your adaptive tracker to this video, you will probably find that you will have difficulty dealing simultaneously with occlusion and the perspective shift as the woman walks away from the camera. Thus, we need some way of relying less on updating our appearance model from previous frames and more on a more sophisticated model of the dynamics of the figure we want to track.

For this problem, I modify my state to [x, y, Velocity_x, Velocity_Y, Frame_Scale]. So, this problem have 5 state variables. X and Y are position for center of the tracked object. Velocity_x and Velocity_Y can help me to build the dynamic motion model for the tracked object. One important change is the Frame_scale, this variable will track size of the object. The tracking template will change according to the particle variance and the error between the current object and saved template.

The template will stop updating if the current object have a large error, which means the occlusion happened. During the occlusion, the particle will move according to its dynamic motion, because the occlusion happened at middle of the tracking, so the particleâĂŹs dynamic model has been already build by filter out the other unfitted model. Therefor during the occlusion, the template will stop updating, and the particles are moving according to its own dynamic motion model.

Figure 13 14 15 16 17 are the results pictures. Figure 13 is the initial image from this video, compare with the Figure 17, the size of the figure is changing. And in the middle of the tracking, there are tow times for occlusion happened. From Figure 14 we can easily find out the variance of the particle get larger. Figure 15 is the image after the first occlusion, we can see the variance become smaller. Also the size of the object changed during the tracking.

Fig. 13: Occlusions_000

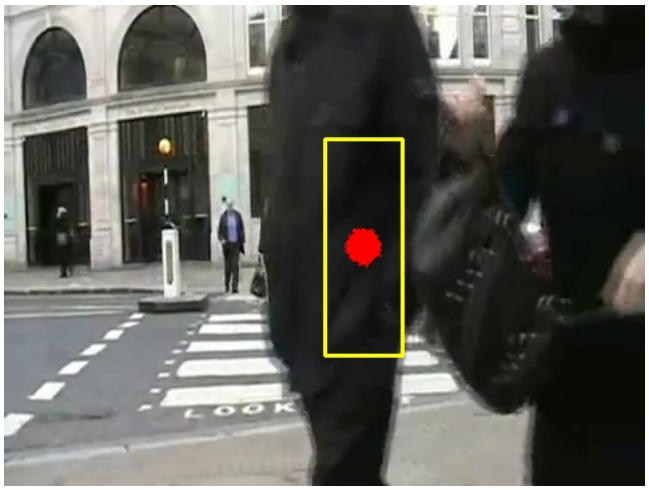

Fig. 14: Occlusions_135

Fig. 15: Occlusions_158

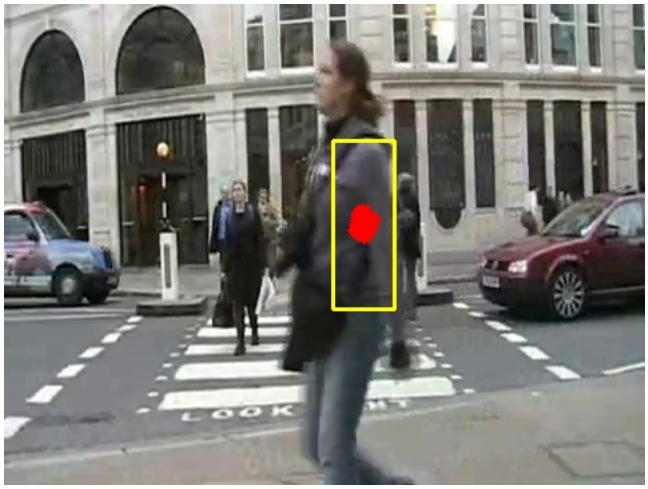

Fig. 16: Occlusions_198

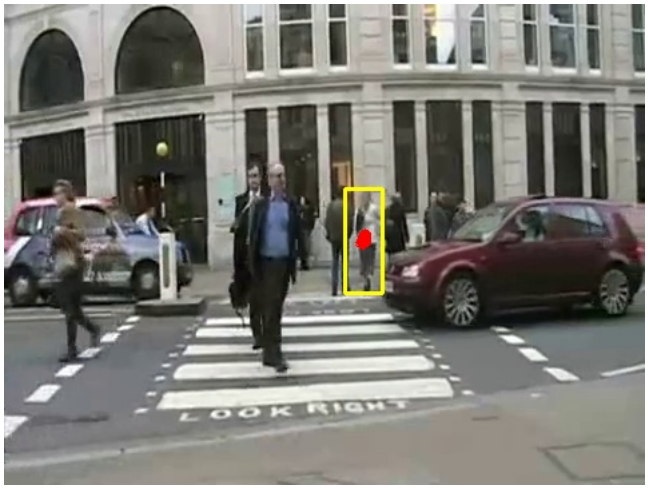

Fig. 17: Occlusions_318

E. Tracking Multiple Targets

In this section, I will investigate more on multiple object tracking. Figure 18 are the three objects, in this problem, the object appear at different time and vanish at different time also. Figure 19, 20, 21, 22 are the results. Multi-object tracking is pretty easy, just have multiple particle filter for each object.

Fig. 18: Multi Objects

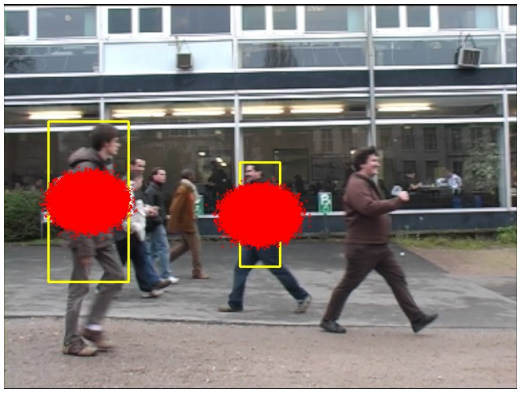

Fig. 19: Multi Objects 001

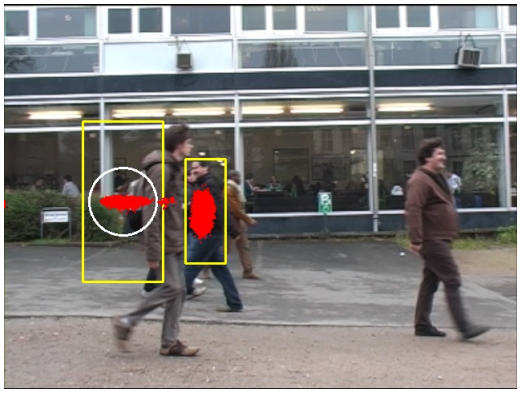

Fig. 20: Multi Objects 012

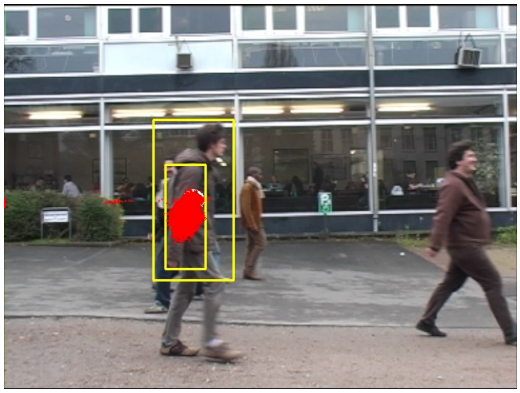

Fig. 21: Multi Objects 017

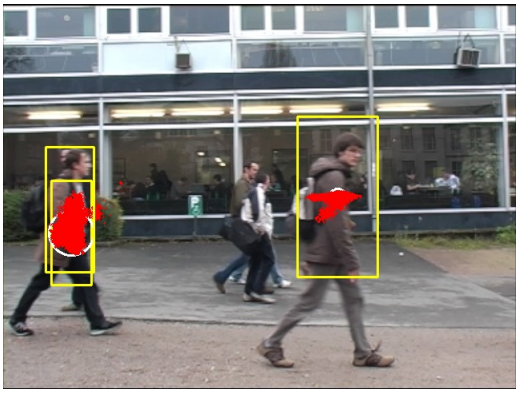

Fig. 22: Multi Objects 036

III. Simultaneous Localization and Mapping(SLAM)

A. Feature Point Extraction

There are whole chapter and papers on the feature point extraction. So I am not going to discuss too much on this topic. Figure 23 is one of the examples of feature points, the red circle in the image are the feature points. Scale-invariant feature transform(SIFT) is the most common used feature points extraction technique, and the feature points are also called key points. SIFT is a robust algorithm to scale, transform and rotation. So the key-point find in the current frame has a high probability in the next frame also.
Feature matching also another topic. However, in this report, we only use this technique. Each of the key points will have its own descriptor, and the matting algorithm will search the descriptor for the target image. The expected movement of those key point is not too far from each other. So according to the sampling time, we can estimate the movement of the camera.

Fig. 23: Feature Point of an Image

B. Camera Position Transformation

Before estimate the camera position, we need to know how the 3D sense project to 2D image. The work flow is this: 3D sense project to 2D image using camera, estimate the position and movement using the key-point in 2D image, estimate the 3D position of the camera in the world coordinate, and then

build a 3D map.
For the camera, there are intrinsic and extrinsic parameters Those parameters are fixed by the manufacturer. Figure 24 is the example of transform the world coordinate to the camera coordinate and also the image coordinate.

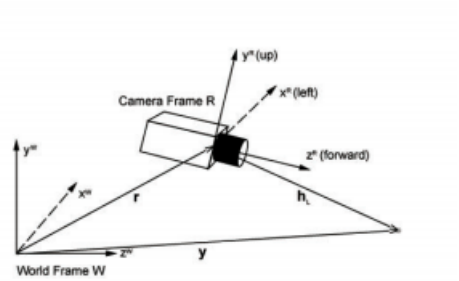

Fig. 24: Camera Parameter

C. Camera Position Estimation

With the knowledge of the coordinate transform and feature points information, we can plug in the Extented Kalman Filter to estimate the position of camera in world coordinate. Figure 25 is a simulated image for camera position estimation. Figure 26 and 27 explain the states of the camera. Figure 28 are the equation to update the camera states.
The map is initialized at system start-up and persists until operation ends, but evolves continuously and dynamically as it is updated by the Extended Kalman Filter. The probabilistic state estimates of the camera and features are updated during camera motion and feature observation. When new features are observed the map is enlarged with new states and, if necessary, features can also be deleted.
The probabilistic character of the map lies in the propagation over time not only of the mean best estimates of the states of the camera and features but a first order uncertainty distribution describing the size of possible deviations from these values. Mathematically, the map is represented by a state vector as figure 26. State vector x is composed of the stacked state estimates of the camera and features and P is a square matrix of equal dimension which can be partitioned into submatrix elements.
In doing this, the probability distribution over all map parameters is approximated as a single multivariate Gaussian distribution in a space of dimension equal to the total state vector size.

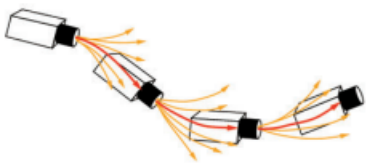

Fig. 25: Camera Parameter

$$\mathbf{x}_v = \begin{pmatrix} \mathbf{r}^W \\ \mathbf{q}^{WR} \\ \mathbf{v}^W \\ \omega^R \end{pmatrix}.$$

Fig. 26: Camera state

$\mathbf{r}^W$ 3D position vector
$\mathbf{q}^{WR}$ orientation quaternion
$\mathbf{v}^W$ velocity vector
ω^R angular velocity vector

Fig. 27: Camera state represent

D. Motion Modeling and Prediction

After start-up, the state vector is updated in two alternating ways: 1) the prediction step, when the camera moves in the blind interval between image capture and 2) the update step, after measurements have been achieved of features.
We assume that, in each time step, unknown acceleration aW and angular acceleration W processes of zero mean and Gaussian distribution cause an impulse of velocity and angular velocity.

$$\mathbf{f}_v = \begin{pmatrix} \mathbf{r}^W_{new} \\ \mathbf{q}^{WR}_{new} \\ \mathbf{v}^W_{new} \\ \omega^R_{new} \end{pmatrix} = \begin{pmatrix} \mathbf{r}^W + (\mathbf{v}^W + \mathbf{V}^W)\Delta t \\ \mathbf{q}^{WR} \times \mathbf{q}((\omega^R + \Omega^R)\Delta t) \\ \mathbf{v}^W + \mathbf{V}^W \\ \omega^R + \Omega^R \end{pmatrix}.$$

Fig. 28: Camera state update

E. Real Time Tracking

Figure 29 is the result using my desktop and web camera.

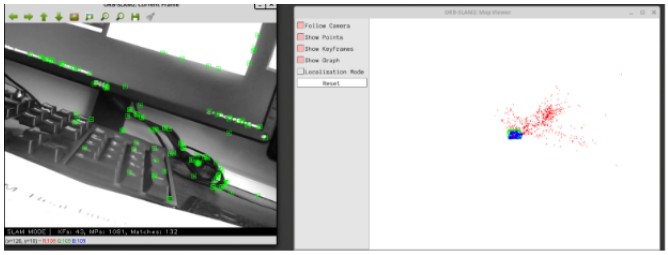

Fig. 29: Real time v-slam