VIRGINIA TECH

ECE Department

ME5524 Bayesian Robotics

Recursive Bayesian Filtering Practice for Robot Visual Search

Student:
Xiaolong Li

Supervisor:
Prof. Tomonari Furukawa

April 28, 2017

1 Background

Visual servoing has become a critical technique in today's robotic world, since vision could help automatic robots sense abundant information accomplish tasks in real environments. However, the real environments could be dynamic and unpredictable, existing methods of target tracking can easily lose visibility of targets, which will lead to servo failure[1], like mutual information [7] or histogram distances [6]. Although these methods added improvements to the robustness of the control schemes with respect to global changes such as illumination changes, these methods remained restricted in terms of convergence area due to the narrowness of the convex areaof the associated cost functions (which is highly nonlinear) around the desired positions[5].
When the lost target's location is not predictable or narrowly localized the sensor needs to acquire new data to locate it. This new data can be obtained by executing a time-optimized search, based on past data that the sensor has collected. Recursive Bayesian Filtering (RBF) algorithms have been used to generate the optimal search paths for various applications such as unmanned aerial vehicles searching for a lost target at sea and camera mounted mobile robots searching for a target [3, 4]. In this course project, I apply RBF to the task of automatic visual search in man-made environment. A gaussian motion model is used here, while also sensor-based observation model is adopted. I use a novel particle filter with a priori target tracking information considered. The simulation and experiment have shown this method has good robustness when the target is lost in FOV. Further improvement and application are also discussed.

2 Objective

The system's overall goal is to keep the target at the center of the image as the camera is tracking it, and search for the target when it is outside the camera field of view. The experiment set-up is as follows:

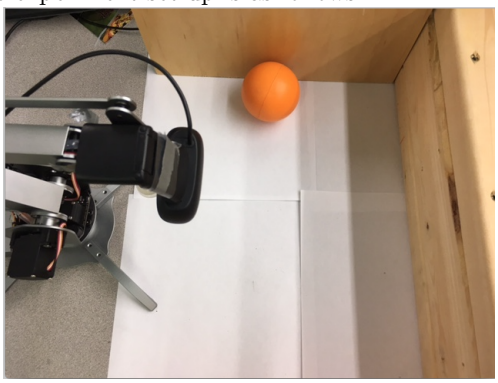

Figure 1: Experiment Setup

In our experimental system, the robot arm from uArm company is 3-DOF, with moving range 50mm - 320mm in the coordinates centered in its base. A wired webcam is mounted at right the end of this robot arm, it is connected to the computer to transfer 2-D 720p video flow. The target is a rubber ball, with pure color yellow, while the environment is set up with wood, its background color is mainly white and brown to help detect the object.

3 Autonomous perception

Our system is a simple visual-servo system, the vision is used for detection and navigation. When the ball is in our FOV, we will detect the ball based on both its color and shape. A simple optical flow is also used to better follow the ball. There are actually different cases of detection, so many environmental factors will actually influence our detection, like environment light, the fences, and so on. After adjusting code according to the environment, the whole stuff actually shows a very robust performance towards different situation:

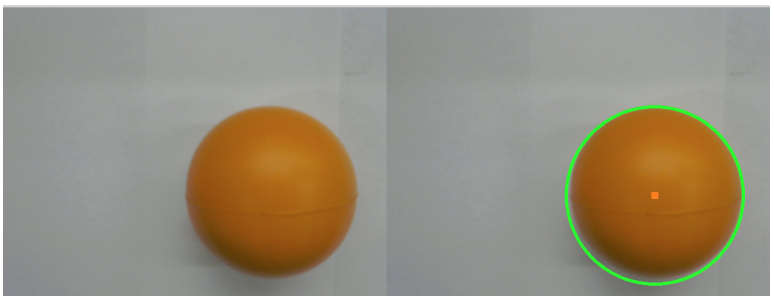

Figure 2: The FOV is clean

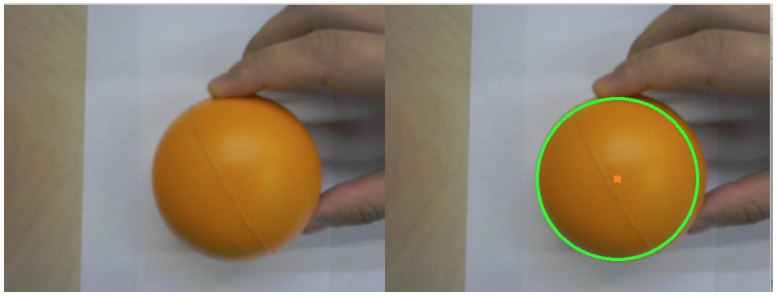

Figure 3: The FOV is not clean

4 Autonomous control

4.1 Motion and sensor models

The dynamic motion model of the target irrespective of its situation wit respect to the camera view is:

$$X_k^t = f(X_{k-1}^t, w_k^t) \tag{1}$$

since when the ball is moving inside the wood fence, its motion is determined by two factors: the frictional resistance from the floor; the block from fence when getting into collision. To make the model simpler, a constant acceleration motion model is used to estimate the target motion[1] in 2D space.

$$X_k^t = HX_{k-1}^t + Gw_{k-1}^t, H = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} \frac{1}{2}a_x^t dt^2 & 0 \\ 0 & \frac{1}{2}a_x^t dt^2 \\ a_x^t dt & 0 \\ 0 & a_x^t dt \end{bmatrix}$$

Since the camera sensor is fixed on the end of robot arm, so its motion should taken into consideration, and we know this by doing optical flow. For sensor model, since the whole process of searching-and-tracking has both the case when the target is inside the camera and the case when it is lost, so by referring to [1,8], I adopt a Bernouli model here.

$$P(Z_k^t|X_k^t) = [p(D_k^t|X_k^t)]^{I_1(X_k^t)}.p(\bar{D}_k^t|X_k^t)]^{I_2(X_k^t)}, I_i = 0 or 1. \tag{3}$$

For the sensor model in my case, since the key factor to help detector object when the target lost is detect the target when it is near or right at the edges of the image plane. So instead of using a simple gaussian distribution, a sigmoid function is applied to strengthen the detection on edges. So

$$P(D_k^t|X_k^t) = \frac{1}{N}\prod_{i=1}^{2}\frac{1}{1+e^{-11}\alpha(x-x_k^i)}.\frac{1}{1+e^{-11}\alpha(y-y_k^i)} \tag{4}$$

α denotes the slope of the sigmoid function, in our case $\alpha = 10 Below the picture shows a 2-D searching work space.$

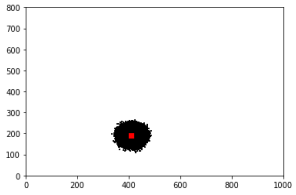

Figure 4: 2D searching space

4.2 Recursive Bayesian Filtering

According to the course materials, either Kalman filter or Extended Kalman filter is not appropriate here, due to the complex situation of non-gaussian distributions. So instead, I adopt a particle filter to handle this circumstance.
The particle filters are Monte Carlo-based method, they are actually a family of algorithms. Particles are sample points with positions and other necessary features. The general procedures can be described as : Randomly generate a bunch of particles, Predict next state of the particles and Mmve the particles based on how you predict the real system is behaving, then update the particles weighting based on new measurement. The last two steps are resample(to discard highly improbable particle) and compute estimate. In our case, we use the SIR filter[10], which belongs to the family of particle filters.

4.3 PI Controller implemented in ROS

A PI control algorithm is used to achieve the stead control. In each step, the controller is trying to move the end of robot arm to move to the position with highest probability that the target will exist.
The control part in ROS is shown below:

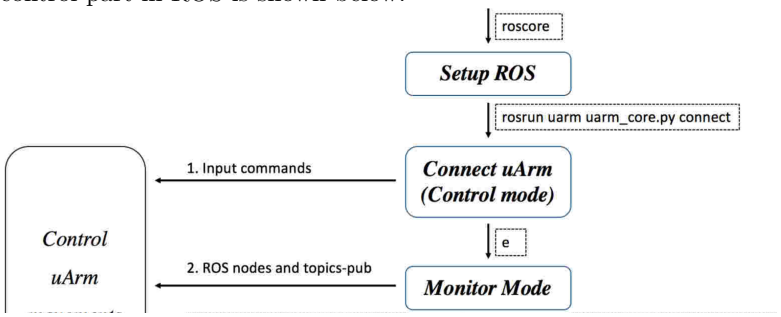

Figure 5: Control Arm in ROS

5 Experimental results

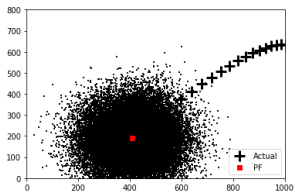

Figure 6: Failure Case

6 Experimental results

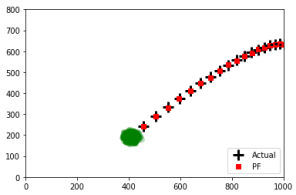

Figure 7: The best case with deterministic model

7 Experimental results

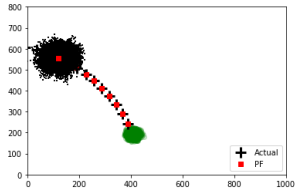

Figure 8: The non-deterministic case

8 Conclusions and future work

As for conclusion for our experiment, the whole framework based on visual detection, recursive Bayesian filtering, and PI control in ROS environment has been implemented with python 2.7 in ubuntu14.04, the basic idea of Bayesian estimation is practiced along this process. Current visual servo system could work well with the case when the ball is inside the field of view, when the target is run out of the view field, it could also try to find the most possible position based on the previous detection information. There are some certain failure cases, and the reason still needs to be carefully checked.
The work has been guided under the course Bayesian Robotics, I learned some of the most the important ideas like motion model, sensor model, and when bayes rule is combined into this framework, the whole view becomes different. More work needs to be done to make this visual-servo system work better to handle those complex situation.

References