Multi-Sensors Fusion and Tracking

Practical - Point-based virtual visual servoing

MENG DI NOUR ISLAM SEPIDEH
Universite de Bourgogne
December 14, 2017

1 Introduction

The objective of this practical is to track an object based on the feature points in the video, which are the image sequences acquired by a camera. Meanwhile, the pose of the object is estimated. We are supposed to grasp the knowledge of using open source library - Visp(Visual Servoing Platform) in the practical. Besides, the Kinect sensor is used as hardware and qtCreator is used as software in this program. Coding language is C++.

2 Part 1: Image acquisition, dot detection and tracking

To acquire images, *MyFreenectDevice* object is used here. The member function *getVideo* is used to get the RGB color images from the device. For displaying the image, we use *vpDisplay* object. Images are displayed and flushed by member functions *Display* and *Flush* respectively. We acquire images from Kinect and display them in a loop so that we can see a video flow from the display window.

Here, a pattern with five dots is represented as our object. We aim to detect the dot in the image and track it during the motion. In order to do so, the object *vpDot2* from Visp library is used. The initial position of the dot is manually set by function *initTracking*. It takes the location where we click and does the detection of a dot(circle) based on this position. Then the detected dot can be tracked along with the video flow by function *track*. From the display window, we can see that the edge and center of the tracking dot are marked as color red. The dot is tracked properly with various speed of motion and various orientations.

3 Part 2: Initial pose computation

In this part, we want to estimate the pose of the object in the camera frame. So that the intrinsic parameters of the camera should be known. We use *vpCameraParameters* object to introduce the intrinsic parameters. $\alpha_u, \alpha_v, u_0, v_0$ are assigned with constant values.

Instead of tracking only one dot in the part 1, we select four dots representing the object. *vpPoint* is used to create an array for the four points. We set their world coordinates manually with specific values. The coordinates of the four points in the image plane can be get from *getCog* function, which is to obtain the dots center of gravity. We then convert the four points from image plane to normalized plane by using method *vpPixelMeterConversion::convertPoint*. Moreover, the four points can be added to *vpPose* object by function *addPoint* effectively. Therefore, the initial frame of the object can be shown by function *display*. We can see from the image that the coordinate system is present in the center of the pattern in red.

4 Part 3: Point_based virtual visual

We aim to do visual servoing based on four dots on the object in this part. Firstly we define an object *vpRobotCamera* which is working like a virtual robot. Its initial pose is set with the last pose we obtained from part 2. For initialization, the servoing frame is set to camera frame itself by function *vpServo::ETEINHAND_CAMERA* and the type and computation method of interaction matrix are initialized by using methods from object *vpServo*. Then we use the four points from part 2 as feature points and define four desired feature points. By changing the frames, we update the feature points. Each iteration, we compute the control law which is the velocity to update the current pose. At the end, each frame can be displayed with new object pose and the four feature points match the desired ones in the last frame.

A simple demo is provided:

`https://www.youtube.com/watch?v=paiVo2uTgcE`