Practical – Intensity-based Visual Servoing

This practical aims to track an object, in images acquired by a camera, and to simultaneously estimate its pose, knowing a *3D model* of it. The tools that are going to be used are a digital camera (Kinect) for the hardware, and the C++ language, CMake and ViSP library for the software side. ViSP stands for Visual Servoing Plaform and comes with its documentation. This is a framework for vision-based control of a robot and visual tracking using several features. CMake is a C++ project configuration tool to be used with many IDEs (e.g. qtCreator).

To configure the practical programming project:
Run CMake, select the practical source and build directories and then configure and generate.

ViSP proposes to track a template using image registration algorithms directly based on the intensities. Contrary to the common approaches based on visual features, this method allows to be much more robust to scene variations.

Part 1: Image acquisition, pattern selection and tracking

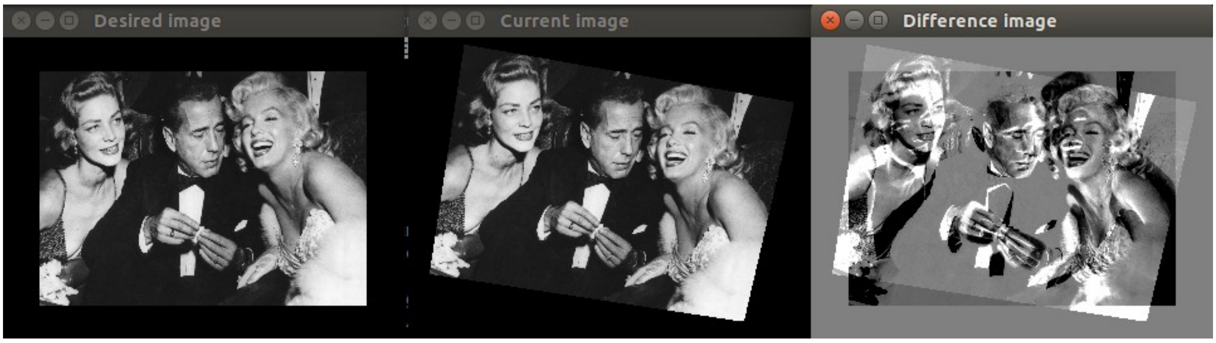
1.
   

   a. Image acquisition is done using a *MyFreenectDevice* object. After declaring it, open the device using the *open* method. Start to acquire an image using the *getVideo* method, taking a *vpImage* object as a parameter, still on the same object.
   b. Image display is done using a *vpDisplayX* object. Declare and instantiate it with four parameters: the image to display, the horizontal and vertical positions of the window and the name of the window. Then, use the *vpDisplay::display* static method to prepare the image display and the *vpDisplay::flush* method to flush the display and effectively display the image.
   c. Loop on these acquisition and display stages to display the camera video flow.
2. Two different similarity functions have been implemented in ViSP to compare two patterns: the "Sum of Square Differences" and the "Zero-mean Normalized Cross Correlation".
   a. Declare and initialize a *vpTemplateTrackerSSDInverseCompositional* tracker object.
   b. Initialize the pattern pose by using the initClick method of the tracker.
   c. Use the pattern tracking in the acquisition and display loop, calling the *track* method, to test its robustness with respect to motion and various orientations.

Part 2: Intensity-based Visual Servoing

1. Initialize the robot pose (cMo).
2. Set the camera at the desired pose (cdMo), acquire the desired image and display it.
3. Set the camera at the initial pose (cMo), acquire the initial image and display it.
4. Compute the initial difference image using the static method *imageDifference(…)* of the object *vpImageTools*. Display it.

5. Create and initialize two visual features based on the intensities of the desired and the current images (*vpFeatureLuminance*).



6. Program the main loop of the Intensity-based Visual Servoing:
    a. Acquire and display the current image using the *getImage(…)* function of the *vpImageSimulator* object.
    b. Update and display the current difference image.
    c. Build the current visual feature (see *vpFeatureLuminance* methods).
    d. Compute the current error (see *vpFeatureLuminance* methods).
    e. Compute the robot velocities following a Gauss-Newton control law.
    f. Send the resulting velocities to the robot.
    g. Get the new pose of the robot end effector and use this pose to set the new camera pose.
7. Loop on the steps described in question (6) until the visual servoing reach a end criteria of your choice.