CMPUT 615 Project Report
A robotic photocopy machine (RPM)
http://blog.jinchris.com/index.php/RobotDrawing.html

Jun Jin [1]

[1]University of Alberta

1. PROJECT SPOTLIGHT

Here bellow are some artworks produced by RPM.

- Drawing a rectangle (fig. 1):
- Robot self-portrait (fig. 2):
- Robotic caligraphy (fig. 3):
- Ink refill automatically (fig. 4):

2. SYSTEM OVERVIEW AND ASSUMPTIONS

ROS is used to develop the whole system. An overview of ros nodes are as following:

- Tracker Node (ar_track_alvar): We use AR_Tracker ROS package for the tracking task. AR_Tracker is a robust AR code tracker. Prior to use, we need to set up the configuration by editting the launch file as shown in fig. 5.
- Drawing task publishing node (chris_detection): This node includes the function of Geometry shape detector and the homography transformation.
- IBVS node (ibvstest2): The IBVS node subscribes to several topics published by (chris_detection), and takes into the current position, as well as the desired position, then perform the IBVS task.

rqt_graph of the ROS topics is shown in fig. 9.

2.1. Assumptions

- The resilience of a pen is adaptive enough, thus, force control will not be considered in the project. A Chinese brush is used.

3. METHODOLOGY

3.1. Camera robot configuration

In this project, we use the eye-to-hand camera robot configuration. Fig. 10 shows the setup of the project. The fixed kinect camera is used and we only obtain the RGB color image, depth data is not used in the project (fig. 10).

3.2. Camera-Robot configuration

The camera-robot configuration in the project is the eye-to-hand configuration as defined in [Chaumette and Hutchinson 2007]. One standby camera will be used in the system.

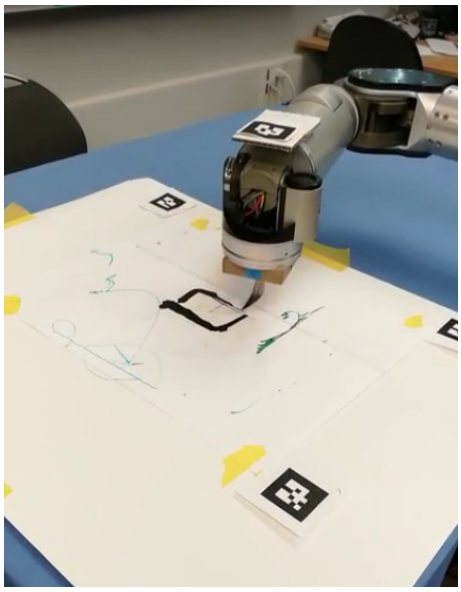

Fig. 1. Example of drawing a rectangle

Fig. 2. Example of Robot self-portrait

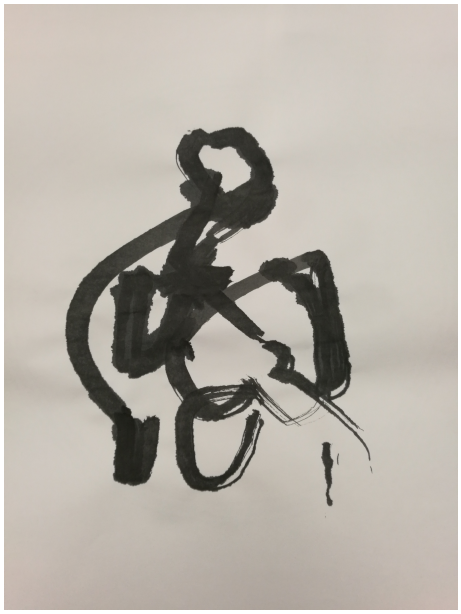

Fig. 3. Robot caligraphy

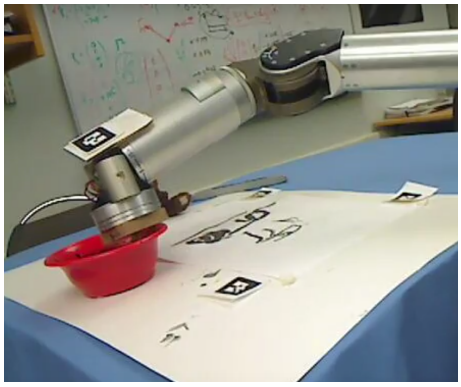

Fig. 4. Ink refill

```
<launch>
        <arg name="marker_size" default="4.4" />
        <arg name="max_new_marker_error" default="0.08" />
        <arg name="max_track_error" default="0.2" />

        <arg name="cam_image_topic" default="/kinect_head/depth_registered/points" />
        <arg name="cam_info_topic" default="/kinect_head/rgb/camera_info" />
        <arg name="output_frame" default="/torso_lift_link" />

        <node name="ar_track_alvar" pkg="ar_track_alvar" type="individualMarkers" respawn="false" output="screen">
                <param name="marker_size"           type="double" value="$(arg marker_size)" />
                <param name="max_new_marker_error"  type="double" value="$(arg max_new_marker_error)" />
                <param name="max_track_error"       type="double" value="$(arg max_track_error)" />
                <param name="output_frame"          type="string" value="$(arg output_frame)" />

                <remap from="camera_image"  to="$(arg cam_image_topic)" />
                <remap from="camera_info"   to="$(arg cam_info_topic)" />
        </node>

</launch>
```

Fig. 5. Setting the launch file for AR_Tracker

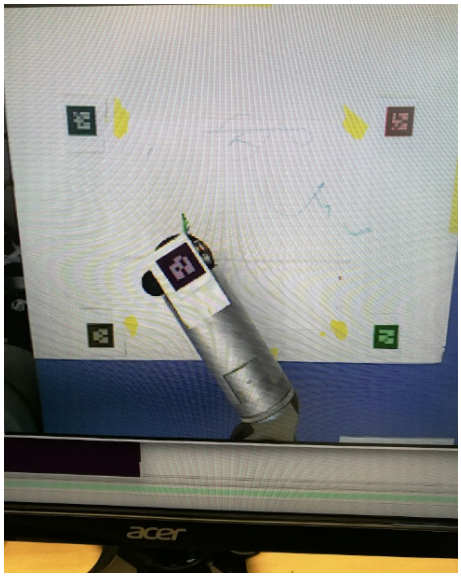

Fig. 6. Tracking result of the AR_Tracker

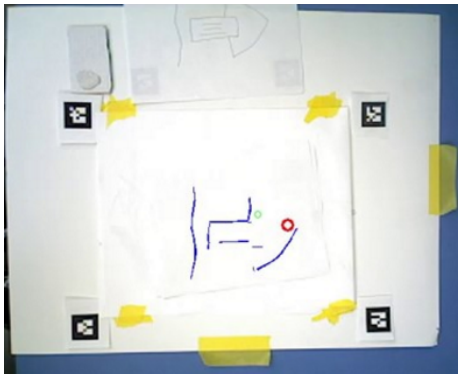

Fig. 7. Drawing task publishing node

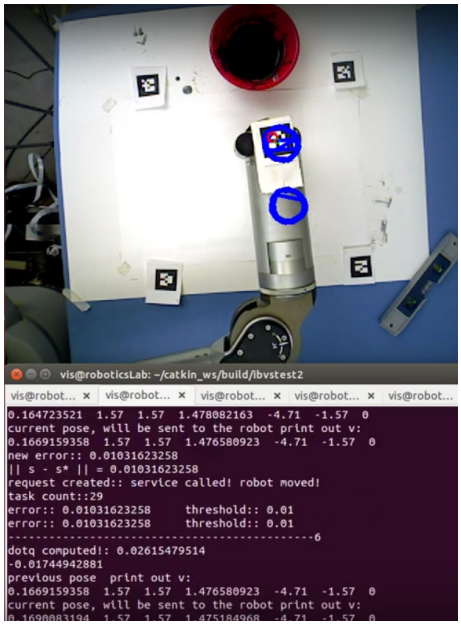

Fig. 8. Running result of ibvstest2 node.

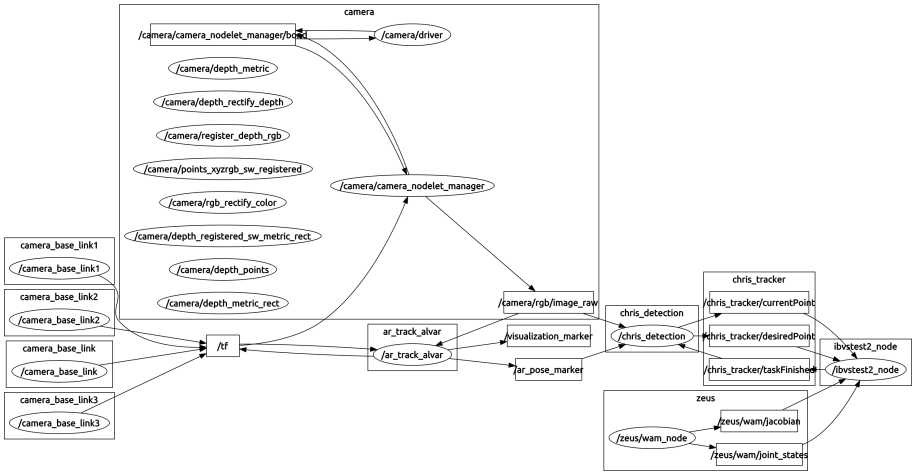

Fig. 9. ROS topics in the project.

Fig. 10. Camera robot configuration.

Algorithm 1: ELSD

Input: Gray-level image x, **parameters**: none
Output: $\mathcal{L}_f$ – list of valid features (line segments, circular arcs, elliptical arcs).

$grad \leftarrow$ compute_gradient(x);
foreach *pixel* p_i *in* x **do**
  $R \leftarrow$ region_grow($p_i, grad$);
  $C \leftarrow$ curve_grow($R, grad$);
  $line \leftarrow$ fit_rectangle(R);
  $circle \leftarrow$ fit_circular_ring(C);
  $ellipse \leftarrow$ fit_elliptical_ring(C);
  $(\mathrm{NFA}_{line}, \mathrm{NFA}_{circle}, \mathrm{NFA}_{ellipse}) = \mathrm{NFA}(line, circle, ellipse)$;
  $\mathrm{NFA}_{min} \leftarrow \min(\mathrm{NFA}_{line}, \mathrm{NFA}_{circle}, \mathrm{NFA}_{ellipse})$;
  if $\mathrm{NFA}_{min} \leq 1$ **then**
    add feature corresponding to NFA_{min} in $\mathcal{L}_f$;
  end
end

Fig. 11. ELSD algorithm

3.3. Geometric shape detector

The automatic detection of elementary geometric features (line segments, elliptical arcs) in images, is as old as computer vision itself [Pătrăucean et al. 2012]. The most annoying thing in such shape detectors is the need of fine tune parameters. In the project, "a parameterless line segment and elliptical arc detector with enhanced ellipse fitting" (ELSD) will be used.
The main idea in ELSD is [Pătrăucean et al. 2012]:

- First, feature candidates are identified using a heuristic.
- Then, each candidate has to pass a validation phase. Owing to the multiple families of features addressed.
- A model selection step is required to choose the best geometric interpretation.

An example of applying ELSD is shown in fig 11.

3.4. IBVS

In image-based visual servoing, the error signal and control command are calculated in the image space. The task of the control is to minimize the error of the feature-parameter vector. The key process is to calculate the image Jacobian, which is able to map the image space velocities to the robot's task space velocities.[Corke and Hutchinson 2001]

3.4.1. Approximating trajectory using discretized points

In IBVS, there in uncertainty in the trajectory of the end-effector. It'll be difficult for the manipulator following a desired trajectory when drawing. However, we can discretize line segments and elliptical arcs into a set of desired points, by approaching every discretized point, the end-effector can form an approximated trajectory.

3.5. Control Law

The control law is determined by our camera robot configuration. The following control law is used in our project:

$$\dot{s} = -L_S {}^C\mathrm{V}_F {}^F\mathrm{J}_e \dot{q} \tag{1}$$

- In this control law, the joint velocity $\dot{q}$ is used as an input to robot control. In order to get a more precise control, we apply an constant interval δt to the velocity, so that we can directly control the robot arm using joint angles.
- ${}^{C}\mathrm{V}_{F}$ is the SE(3) transformation from the Robot base frame to the camera's frame. ${}^{C}\mathrm{V}_{F}$ can be determined by estimating the camera's extrinsic calibration parameters. It turns out that, the estimation of ${}^{C}\mathrm{V}_{F}$ can be very rough, and IBVS is still able to converge in a good result. So, we simply measure the translation of the camera wrt. the robot base frame, and apply a rotation matrix so that, $X_{robot} = X_{cam}$, $Y_{robot} = -Y_{cam}$ and $Z_{robot} = -Z_{cam}$.
- ${}^{F}\mathrm{J}_{e}$ is the robot jacobian, which describes the robot's inverse kinematics. ${}^{F}\mathrm{J}_{e}$ can be directly obtained from the standard rostopic published by WAM_Node.
- It should be noted that, in order to estimate $\dot{s}$ and L_S, we need to update feature point positions in the image space, i.e., update the current point P and the desired point P_d. All the coordinates of P and P_d should be provided after camera's intrinsic parameter calibration.
- Commonly, if we want to control a N Dof robot, then we need N/2 feature points. In our task, we only have one feature point, which can contribute two rows in the interaction matrix. So, it's reasonable for us to use only 2Dof of the WAM arm robot.

4. CONCLUSIONS AND DISCUSSIONS.

4.1. Conclusions

As a conclusion, the aforementioned eye-to-hand robot drawing system needs the following knowns:

- Robot's inverse kinematics ${}^{F}\mathrm{J}_{e}$. ${}^{F}\mathrm{J}_{e}$ needs to be updated in every iteration.
- Camera's intrinsic and extrinsic calibration parameters. They are required in the initialization. However, those parameters can be roughly estimated.
- The current position of the tracker on the end-effector and the desired position. Both coordinates are defined in image space and need to be updated in every iteration.

The overall project goal is reached within the deadline, however, there are still some problems with the system:

- The controlling loop is very slow. For the safety issue, we manually set the exponential convergence factor λ be very small. Besides, as we use ROS service to control the robot, it's been tested that the maximum command frequency capability for an effective WAM robot response is 4 to 5Hz, which means that in IBVS, the iteration frequency is restricted to 4 to 5Hz. This problem can be solved if we control using ros topics or rewrite the WAM ROS package using actions.
- Jumping errors. The robot is required to jump from line to line, however, in practice, errors may happen randomly, due to the feedback latency of the node chris_detection. This problem can also be solved if we can use ROS action instead of subscribing a ros topic to update current point and desired point locations.

4.2. Discussions

It's very natural to ask the question, if no camera external calibration data and robot's inverse kinematic information are provided, can we still perform the IBVS? It's observed from the experiment that, if the Jacobian which can map the image plane and the robot's joint space be estimated and updated, the answer is yes. Later, I want to explore the uncalibrated visual servoing tasks.

We also tried to think about applications with the robot drawing problem. And there are actually so many applications with visual servoing.

- 1. The first one is laser cutting, which is discussed in class presentation.

- 2. Make the UAV keep following a target.
- 3. Tasks require very precise robot manipulation , for example, a surgery robot manipulator.
- 4. Reconfigurable industry robots working in unstructured environments.

After this course project, my understanding for visual servoing goes beyond the equations. I can understand the intuitive way of people inventing the IBVS technology. I will also contribute my code, which provides an example of using calibrated IBVS, working with the WAM arm robot.

At last, I would like to give my thanks to Dr. Perez and Dr. Dehghan. Thanks for their patience in the discussions, so taht I can have a better understanding of how IBVS works and how to use a simple estimation to do the camera extrinsic calibration. I am so lucky to have them by my side.

5. PERSONAL CONTRIBUTIONS.

Peter, Rong and I collaborated in this project. However, I am the only one registered on the course. I started the project, wrote the project proposal, designed the system setup configurations, wrote chris_detection node, the ibvstest2 node and finished this project report. Considering the special situation of this project, I am wondering can I get extra bonus?

Peter and Rong are not registered on the course, but they are enrolled in the whole project duration. They provided help to setup the system, providing ideas, working on presentable materials. We worked together on weekends and over nights. We discussed a lot and learned from each other. Thanks for their help!
PS: This course project will have a web page for demonstration. Please visit http://jinchris.com/index.php/RobotDrawing.html for experiment videos.

References

Chaumette, F. and Hutchinson, S. (2007). Visual servo control. ii. advanced approaches [tutorial]. *IEEE Robotics & Automation Magazine*, 14(1):109–118.

Corke, P. I. and Hutchinson, S. A. (2001). A new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation*, 17(4):507–515.

Pătrăucean, V., Gurdjos, P., and Von Gioi, R. G. (2012). A parameterless line segment and elliptical arc detector with enhanced ellipse fitting. In *Computer Vision–ECCV 2012*, pages 572–585. Springer.