Progress Report

Bardia Mojra

March 22, 2022

Robotic Vision Lab

The University of Texas at Arlington

1 Specific Research Goals

- VPQEKF (April 1st): Work on the paper.
- DLO Manipulation Dataset (September - ICRA or IROS?)

2 To Do

- QEKF Paper - 30% extension (April 1st):
  - Edit VEst section and add updates.
- QEKF/QuEst+VEst Implementation (Feb. 28th):
  - Implement QuEst 5-point - On-going.
  - Implement VEst
  - Address scale factor (depth-scale) issues
  - Address "hand off" issue when objects enter or leave field of view
  - Real-time streaming images for real-time operation (optional)
  - Experiments
  - Feature point extraction - On-going.
  - Noise issue: noise cannot be modeled
- DLO Manipulation:
  - Related work literature review
  - Real dataset + paper (September 2022 - ICRA):
    - Watch IROS manipulation workshop videos. - Done.
    - Design, discuss and build a data collection and test rig.
  - Unity dataset
    - Recreate virtual duplicates of physical test material
    - Model dynamics and deformity

3 Progress

The following items are listed in the order of priority:

- VPQEKF (April 1st, 2022): The rewrite and debugging of the QuEst the algorithm in Python is finished but a few utility functions such as evaluation, data logging, and visualization remain unfinished. The new code has been compared against the original implementation line by line and so far everything has matched and with few improvements. There are three notable improvements thus far, 1) better feature matching and ranking of best-matched features between frames, 2) and more fine-tuned control over of variable data type in Python with Numpy Quaternion module. 3) Also, the Python implementation has resulted in fewer nonzero Quaternion solutions for equation 20 in [1].

  In the new Python implementation, I took advantage of highly the developed OpenCV modules and used ORB feature detector instead of SIFT. The initial results have been very promising and ORB is supposed to have a shorter detection time than SIFT. I still need to time my implementation and compare it against what is presented in [1]. Moreover, for this implementation, I used Numpy Quaternion module that provides a fast and reliable Quaternion algebra library. By default, it declares all floating-point variables as *float64* and it has a dedicated Quaternion data type, i.e. *Quaternion*. In Numpy, a Quaternion is declared as a set of four *float128* variables. In order to preserve full numerical accuracy, all data handlers are declared *float128* at initialization instead of default data type, *float64*. Additionally, the implementation repeatedly produces a smaller set of solutions compared to it Matlab implementation, 24 instead 32. This is beneficial since it shrinks the pool of possible solutions. The root cause of this is under investigation but it might be related to numerical accuracy and the computational rounding effect. For the evaluation part, we followed [2] where the author presented six functions for measuring the distance between 3D rotations. This is particularly trickly for Quaternions as they are normalized when representing rotation from the previous frame or relative orientation. The paper presents 5 metrics that are suitable for operation on *SO(3)*, three of which I am currently implementing.

  Next, I will finish the implementation and begin writing on the paper this week. I know enough about the paper that I can write it from scratch with you and Dr. Gans as co-authors rather than doing an extension. I would appreciate your guidance on this.

- DLO Manipulation: I read the paper you assigned to me and thank you

for doing that. At this point, I have had a similar picture in my mind regarding how the grasping task could be achieved. As described in the paper, the authors first detect the objects and localize them within the image frame. Then, they proceed to estimate the grasping pose for each object. They use a mask to isolate each object and they provide RGB and depth as input for their deep learning model. They trained their model to estimate midpoint, angle, and 'probability of grasping point' which I assume refers to the inferred probability of ideal grasping pose. They follow up with a alignment step that technically should decrease the rate of falling items. It is a clever and simple solution to a relatively complicated problem and it could be useful in real-world situations, e.g. pick and place on a production line.
But the authors do not provide much on the current literature and related works. Moreover, they do not compare their grasping performance against any other existing methods, nor do they mention whether such methods exist or not. On the merit of the submission, there is no scientific or academic contribution or achievement, nor do the authors make such claims. What is described is an engineered solution to the grasping problem which in all fairness is difficult to implement. This does not amount to a scientific contribution. Moreover, the authors claim their proposed method grasps unknown or never-seen-before objects, and in my opinion, that is stretching it too far; it is referring to cylindrical objects versus cylindrical batteries. In my opinion, that should be considered known objects as to the system they are simply, both cylindrical objects.

- Pose Estimation: I will need it for DLO segment localization.
- NBV-Grasping Project: I talked to Chris about his grasping project and he is not pursuing that project anymore. I will work on grasping after the QEKF project.
- PyTorch Tutorials: Transfer learning.
- SBD Funding: I was thinking about writing a short report for them to attach or to provide with our thank you message. In there, we could explain multiple applications where machine learning could be used to solve their problems. I know where they need robots, it would be in final-stage manufacturing automation and tool qualification test procedures and both are considered pick-and-place applications. Final-stage manufacturing automation would be a UR5 robotic arm moving

parts from a tray to a conveyer belt or placing raw material in a CNC machine and removing the finished part. The tool qualification test procedures are highly dependent on the tool and mostly involve simulating physical use of the tool as a user would. Technicians test a large number of tools on various predetermined tests every time there are ANY changes made to the design of a tool. These cost a lot of money and the jobs are tedious as they have very low employee retention rates. The testing is a real bottleneck and we got yelled at if our code ever failed the test because it has to be fixed and tested from the beginning with new tools. They often test hundreds of the same tools at a time if big enough changes were made to its design. This might not seem much, but there are about four or five brands with 250 tools under each and the entire engineering operation is handled by 3000 employees where only about 15 are embedded software engineers. They produce and sell millions of tools every month, quality is extremely important and every penny matters to them. If we can provide some demos and perhaps set up a test rig with two UR5s in Towson, I am willing to bet could ask for half to a million-dollar from them. I have seen them pay two million dollars for a useless BLE source code that my colleague ended up rewriting anyway. It is a wild world.

4 Intermediate Goals - Fall 2021:

- QEKF: Finish paper.
- UR5e: Do the tutorials.

References

[1] K. Fathian, J. P. Ramirez-Paredes, E. A. Doucette, J. W. Curtis, and N. R. Gans, "Quest: A quaternion-based approach for camera motion estimation from minimal feature points," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 857–864, 2018.

[2] D. Q. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.