


Idea

Pick and place setups are broadly used in many industrial applications which provides high accuracy, repeatability, and precision. In this project, we are implementing this type of setup to a specific application in printing sites at UB (University at Buffalo). Conventionally, in these sites, the printed papers that come out of the printer are taken by a student worker and placed at the corresponding slots in the racks, which are classified by the starting letter of the UBIT name**. For example, if the UBIT name of the person who ordered the print is “kkdas”, the printed papers are taken by the worker and placed at the slot, K. These racks are positioned in such a way that the printer and rack have some distance between them and one of the rack’s sides is used as the loading side and the opposite is used as pickup side for the student customer. The ultimate objective of this project is to automate the task of the worker, which is described above. Although this task for a worker would be easy, it would be challenging for a robot to do the same because of the complexities involved in motion planning and placing the paper in the exact right spot by overcoming the noise that is present in the sensor readings. Hence, we plan to implement a simpler version of the problem initially and improvise depending on the results and plan our future work. Further, this type of setup can be widely used in many other applications like mail classification, inventory management, etc.

***every student is provided with their unique university username called UBIT name at University at Buffalo*

Problem Statement

We have a pile of papers printed by the printers at the UB printing sites placed in a bin (let's call it the bin 0). Each front paper is printed exactly identical to the one shown in the image below excepting the UBIT name which changes based on who ordered the print job.



UB Information Technology

Date: 10/16/2019

Job: 151

Time: 3:14:08 AM

kkdas

Figure 1

The number of bins that need to be there will be equal to the number of classes which in turn is equal to the possible starting letters of UBIT name. For the actual problem, this number will be equal to 26. In our simpler problem, we limit the number of classes to 4, typically bin A, bin B, bin C, bin D and instead of the bunch of papers we're classifying only the front page.

The task of the setup is to:

- Take an Image from the camera.
- Analyze the starting letter of the UBIT name by using OCR algorithm. This tells the robot which is the destination bin for the particular sheet.
- Pick up the sheet from bin 0 which is already analyzed by the camera.
- Move close to the destination bin.
- Align the arm according to the orientation of the paper and place it in the destination bin.
- Move back to the bin 0
- Repeat the previous steps until there are no sheets in bin 0.

In this project, we are using uArm swift robotic arm with four degrees of freedom to pick and place the paper. uArm has two

rotational joints , one rotational and one suction gripper at the end-effector of the robot. This device is also capable of working wirelessly with Bluetooth or a micro USB connector. The maximum payload of the arm is 500g. To move the paper from one point to the other, combination of arm joint movements and turtle bot movements are used. Turtle bot is attached to the base of the uArm for its mobility as shown in Figure 2. To analyze the username character on front page an 800*600 resolution camera is used to take 2D images.

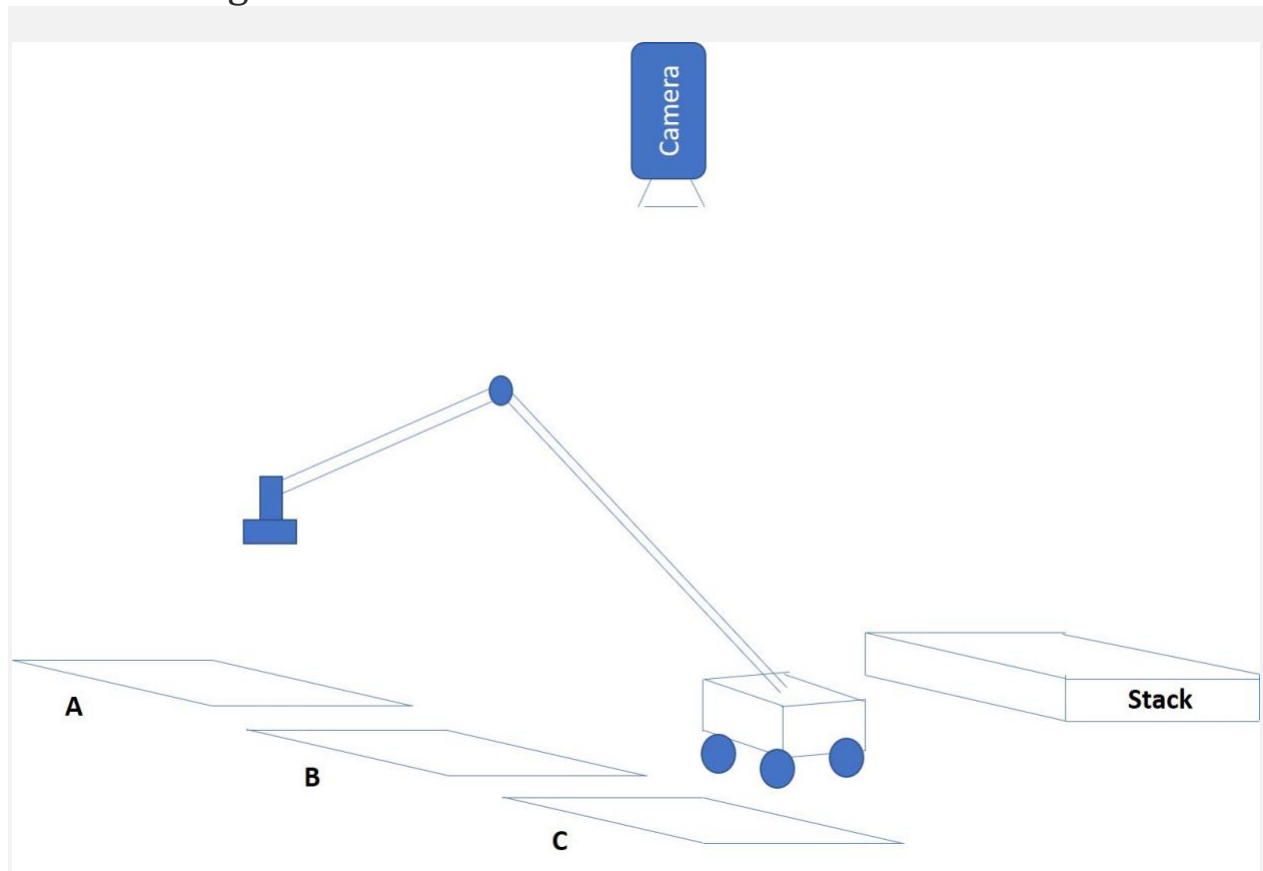


Figure 2. A, B and C are the classified bins (racks) whereas the stack is the printed papers which has to be classified.

Robot working procedure:

1. Stack overhead camera will take the picture and pass it to the tesseract OCR algorithm to identify the characters written. This algorithm will provide the user name in which we'll only extract the first alphabet and decide the destination bin.
2. To pick up the paper, robot have to be localized beside the paper stack (normal to the paper stack). To find this position turtle bot will keep rotating 360 degrees until it'll detect the april tag of the stack.
3. When robot detects the destination position, it'll start approaching towards it by just publishing the linear velocity on /cmd_vel topic.
4. After reaching the desired position, it'll correct it's orientation according to the stack.
5. Then uArm metal will pick up the sheet of paper by just publishing the x, y, z coordinates of the uArm robot by using `uarm.set_position()` and switching the pump by using `uarm.set_pump()`.
6. It'll then localize the destination bin same as it localized the stack.

7. After reaching the destination bin drop the arm by publishing the x, y, z coordinates of the aArm robot by using `uarm.set_position()` and switching the pump by using `uarm.set_pump()`.

Results:

In OCR, camera will only take the picture (Picture shown in figure 1) of the lower portion of page where username of the student is printed. Then this image is taken to the OCR algorithm which will firstly select the portion (as shown in Figure 2) of the image where text is written and then apply the character recognition as shown in Figure 3.

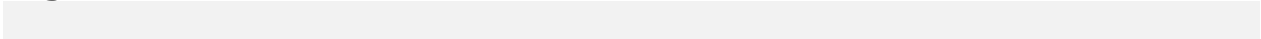




Figure 1. Image acquired by overhead camera.



Figure 2. Processed Image to detect username.

uArm metal robot is operated in three configurations i.e. home position, pick up mode and drop mode. Home position is shown in figure 5 (0,100,150 for x,y and z respectively). whereas, pick up position was dependent on the height of the stack. In our condition the pick up position was 0,180,14 for x,y,z respectively. In real application height of the stack will vary, to overcome this, a depth



Figure 4. uArm with home position on top of turtlebot 3 burger.

We were also able to identify the april tag configuration by accruing the images. We were interested in the tag ID, z direction and orientation of the april tag. This will help robot to localize itself in the workspace.

Challenges

- The part of the project that was most challenging was finding ROS open source packages for the Arpril tags and the UArm that worked on our Versions of operating system. Most versions would give errors when we tried to compile. Probably the versions that we looked for in python were

originally written in other languages and were unofficially ported as a wrapper into our desired language ,which is, python.

- One other thing that was equally challenging was the communication between the different modules like turtleBot, UArm and the laptop. We connected each of the modules you to a single router. The hardest part was to synchronize time in different modules, which we couldn't in the given time frame. So we chose to make the computation on the relatively slower raspberry pi and then work around the problem that may arise by trying to code to accommodate the slowness.
- In the bin detection part, we faced the problem of getting different results for similar images taken at different times of the day from our text recognition code. This was because of the difference in amount of lighting during difficult times of the day. We got our desired result only when we change the threshold value. We solved this problem by using selectROI() function and setting a region of interest which is just the UBID and this gave us better results.