Panoramic Images for Robot Vison

Overview:

Panoramic images have more information than normal images. This feature of panoramic images makes them very special for robot vision. Robots with multiple cameras or a rotating camera can be used to create panoramic images. These panoramic images can be used to perform many different important tasks by robots such as object detection and recognition, motion detection, localization, and obstacle avoidance etc.

Motivation:

The robot that we use in the lab fails to recognize object if it not directly in front of the camera. It needs a better vision system.

Goals:

The main goal of this project is creating panoramic images using the two cameras located in the head of the humanoid robot. The paper Recognizing Panoramas by M. Brown and D. G. Lowe is used as an example to build panoramic images.

- Use a reliable and fast panoramic image stitching and panorama algorithm with OpenCV to create panoramic images.
- Python is used as a programming language
- Test the system
- Compare the results and performance of panoramic images with normal images

Tools:

OpenCV: I have used OpenCV many times in different types of projects before.

Python: I am an experienced Python programmer.

Camera: Two 3 MP USB cameras.

HROS5: I have so much experience with these humanoid robots. We have three of them in the Intelligent Robotics Lab at PSU.

Top-Level Project Design:

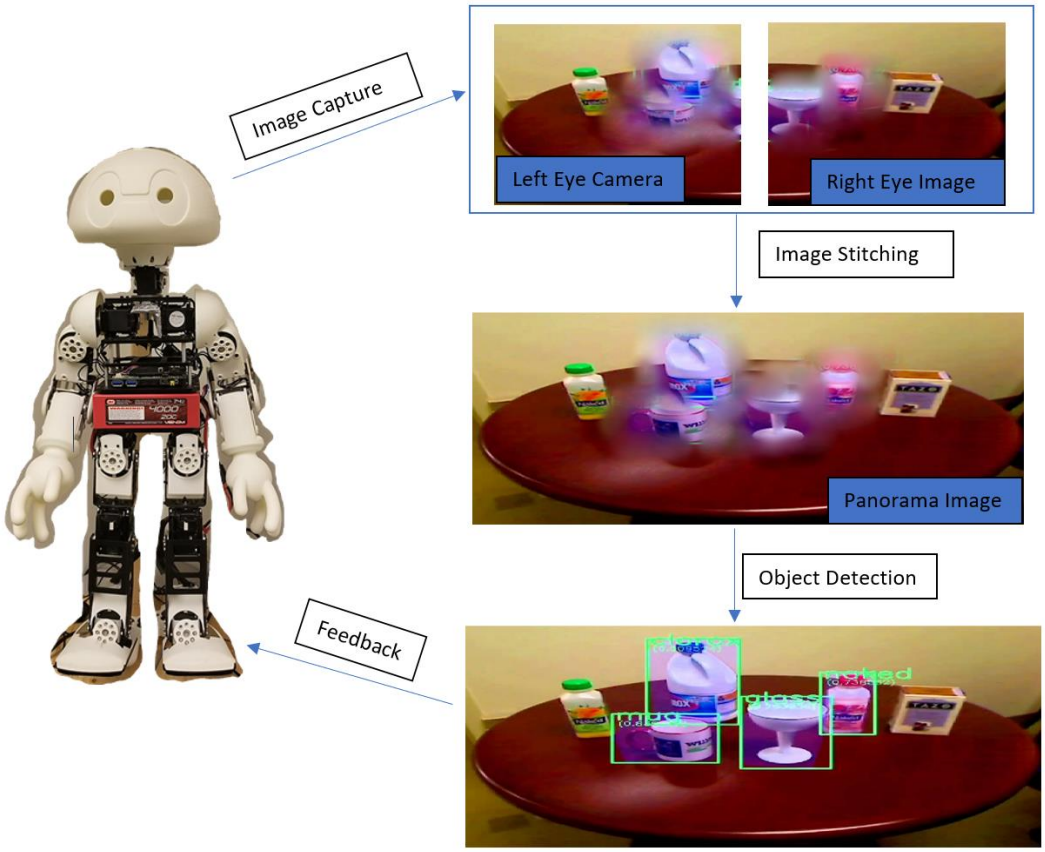
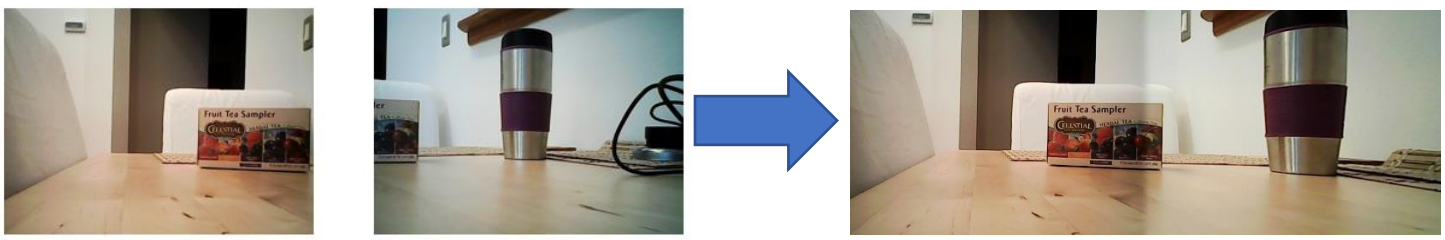


Figure 1 – Project Design Proposal

(*images are picked from the internet)



Left **Right** **Panoramic Image**

Five Steps of Panorama

There are five main steps to follow to create a panoramic image.

1- **Capture Images**

The USB eye cameras are used to take images.

2- **Feature Detection and Matching**

Feature detection is done by using SIFT algorithm and RANSAC algorithm to eliminate the outliers in all matches.

3- **Align Images**

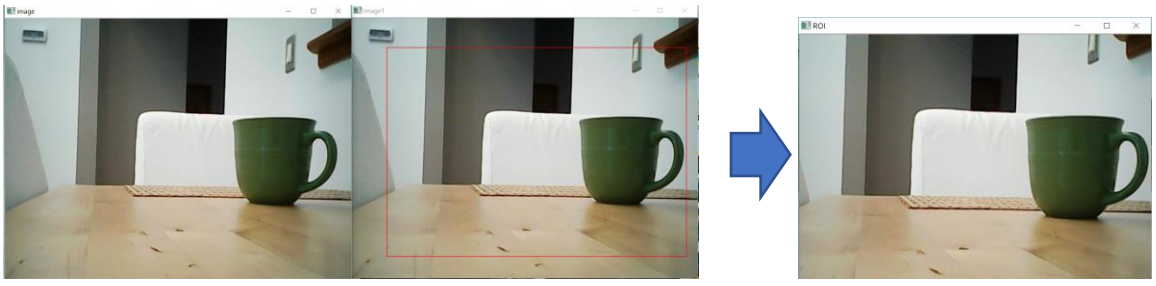
After detecting the key features in both images homography matrix is used to warp the right eye view. Homography matrix tells us how much the right eye camera is tilted, rotated and translated from the left eye camera, so we can align two images. Resulting image is places on a black mask.

4- **Blending**

Stitching two images together without any blending makes panoramic image unrealistic and creates artifacts especially in where two images are stitched. Pyramid blending is used as an image blending method. In this method overlapped region is chosen manually. Number of pixels such 150px. For blending I used two scripts that I found on the Internet. I used the code (Belding 1) as example for pyramid blending and used the same idea to modify the code (Blending 2) for my needs in this project.

5- **Cropping**

Cropping is an optional step to make a panoramic image look nicer. Although it causes some information loss especially at the edges and corners of the panoramic image, it makes the result look one image taken by a single camera and much nicer. Cropping is done manually. I used a python scripty for cropping (Cropping 1) and it modified it to create an image cropping program. Left click defines the left upper point of the ROI rectangle (Region of Interest) and right click defines the right lower point of the ROI rectangle. When cropping is done, button C can be pressed to save the image file. Button R can be pressed the reset the cropping process.



You can find the detailed visual explanation of what is happening in each step below.

Step 1 – Capture Images

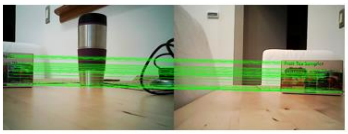
Capture Images


Create a mask size of 2 images

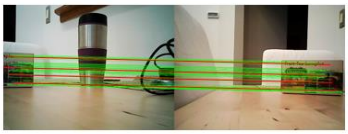
- Create a mask for the warped image

Step 2 - Feature Detection and Matching

Find Matches - SIFT

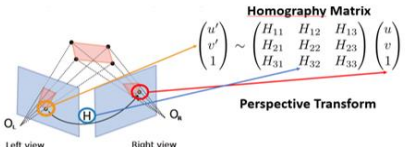


RANSAC



Step 3 – Aligning Images

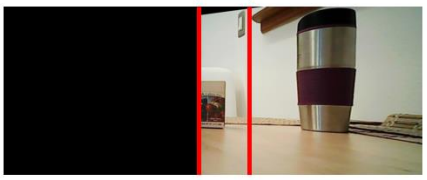
Calculate Homography and warp the image



Step 4 Blending


Overlap 20%

Overlap 20%

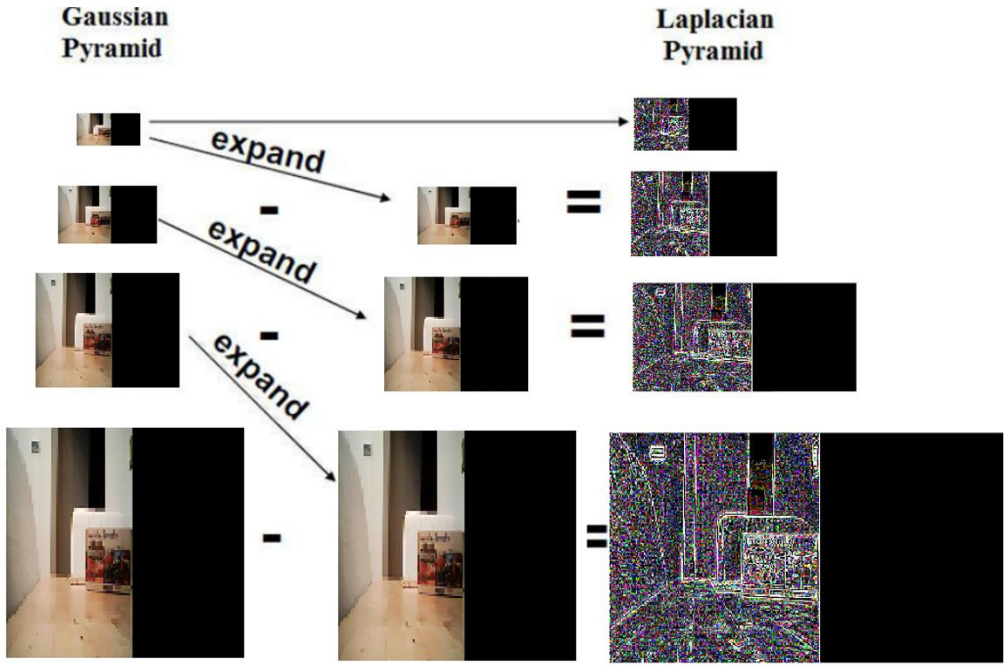



Preprocessing images

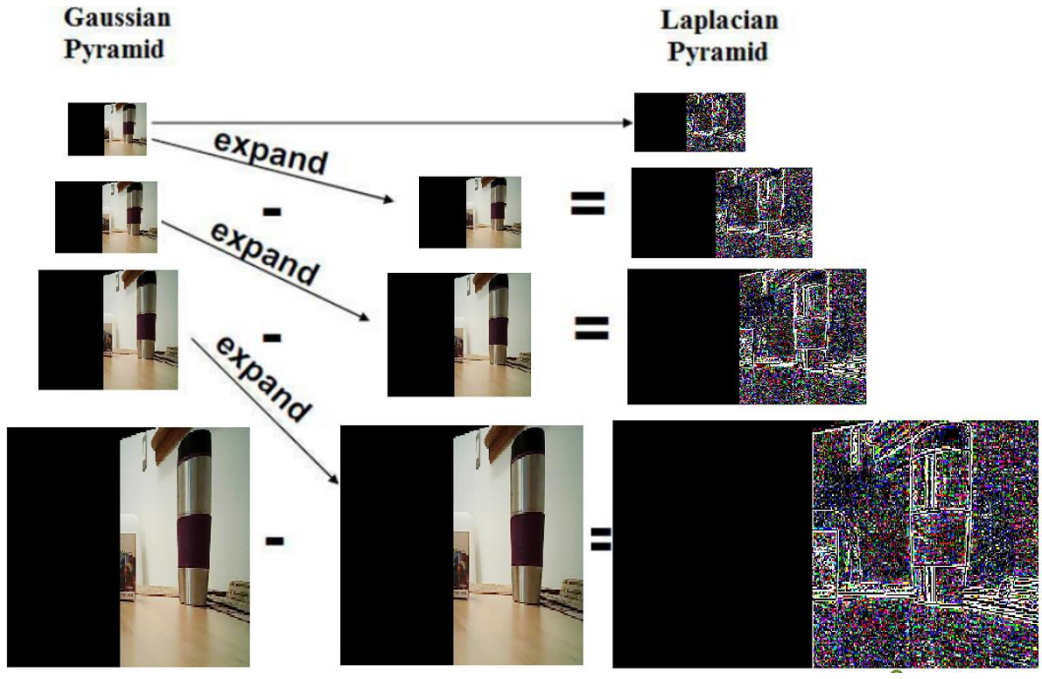
- Create a mask for input images

- Create a mask for the final image

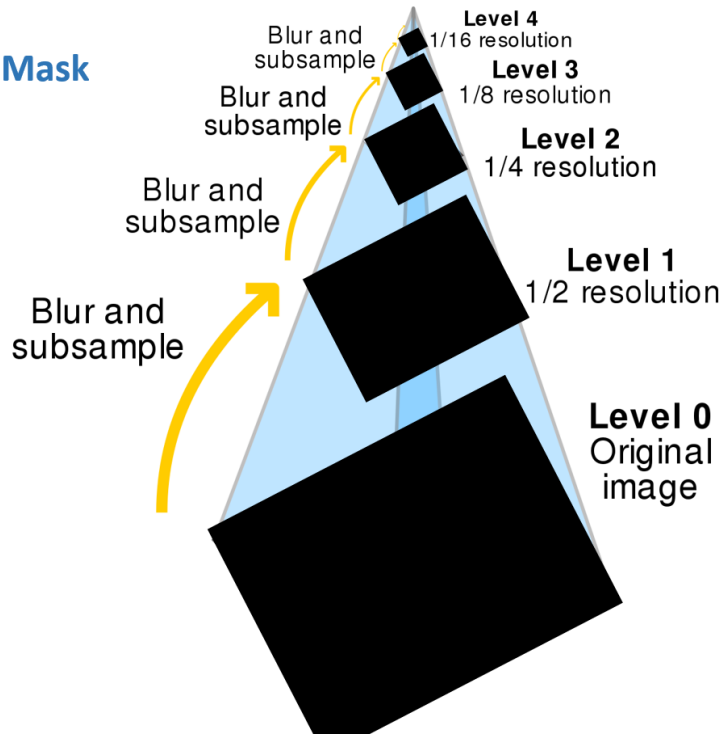
- **Build Laplacian pyramids LA**



- **Build Laplacian pyramids LB**

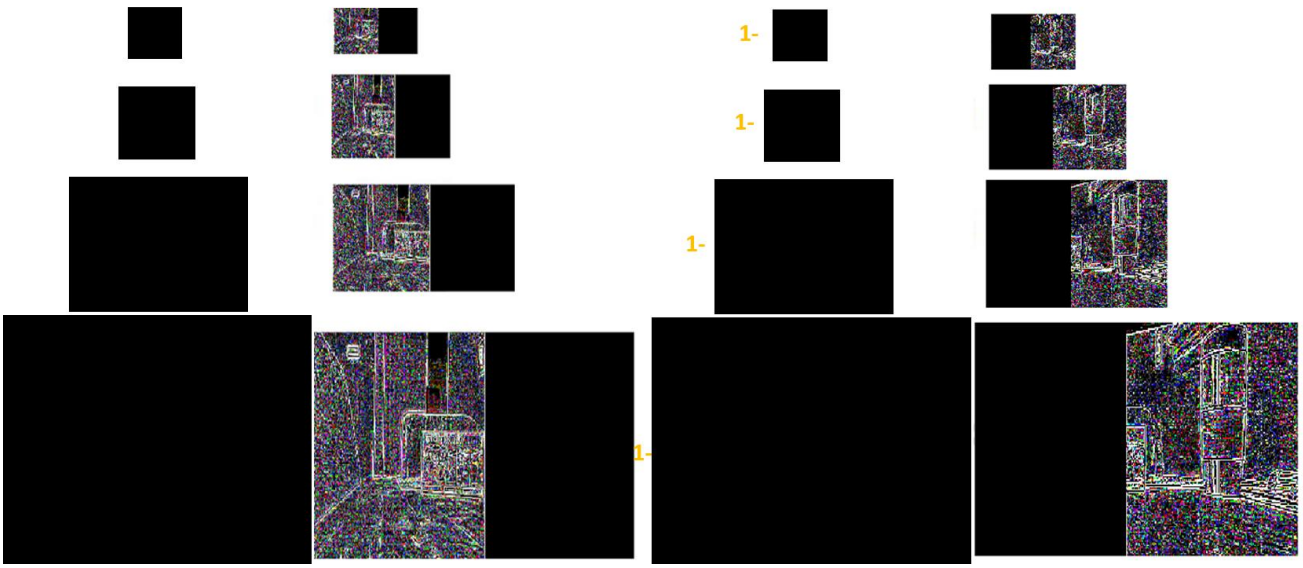


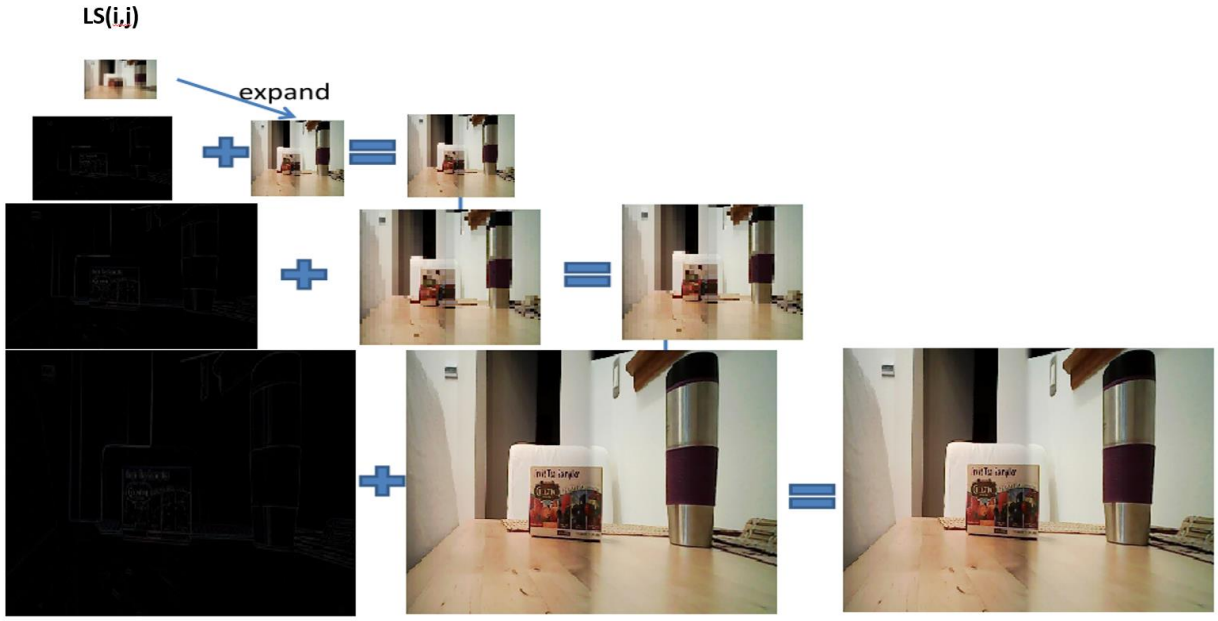
Gaussian Pyramid of the Mask



LS(i,j) = GR(I,j,) * LA(I,j) + (1-GR(I,j)) * LB(I,j)

GR(I,j,) * LA(I,j) + (1-GR(I,j)) * LB(I,j)




LS(i,j)
expand

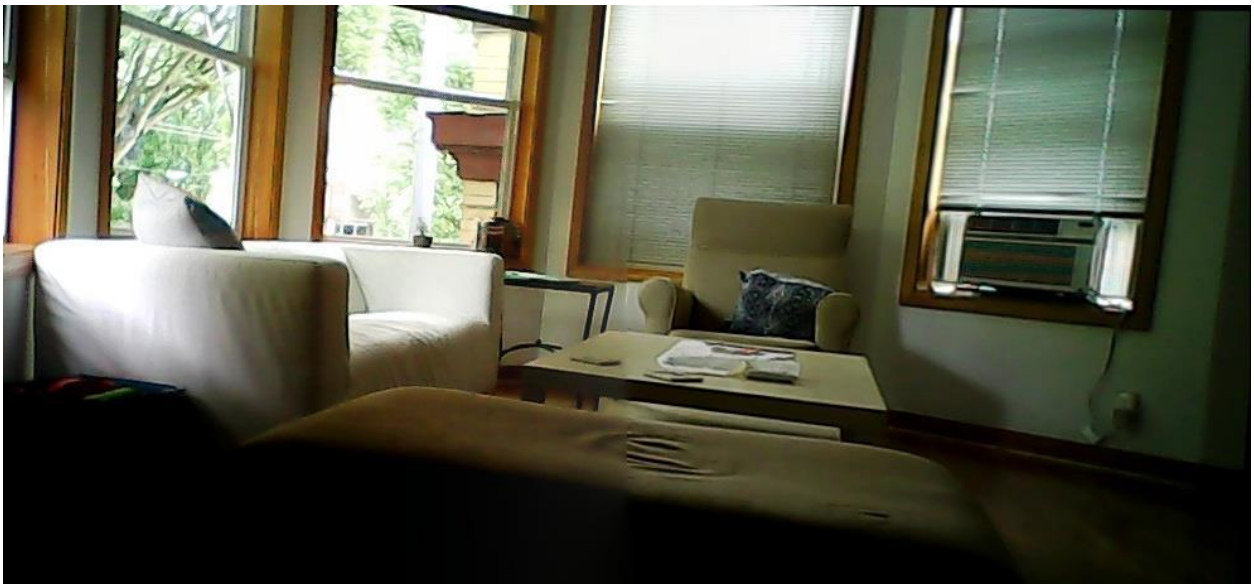
Results:

I have tested my project with 10 different images that I found on the Internet and also captured with two eye cameras. You can find the panoramic image result below.

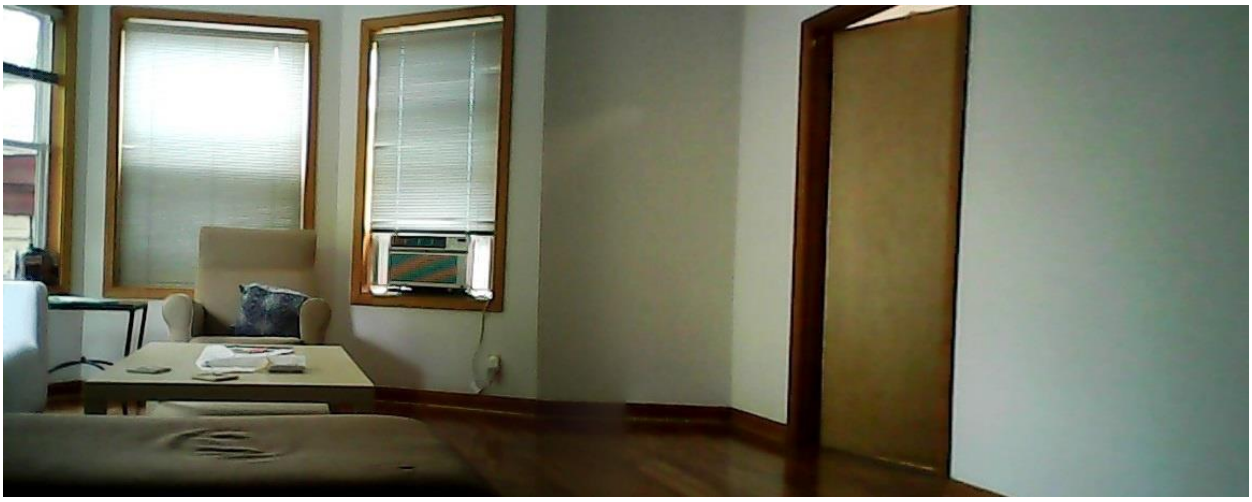
Husnu Melih Erdogan PhD Student - CS 510 - Computational Photography Final Project Report

Husnu Melih Erdogan PhD Student - CS 510 - Computational Photography Final Project Report

Husnu Melih Erdogan PhD Student - CS 510 - Computational Photography Final Project Report



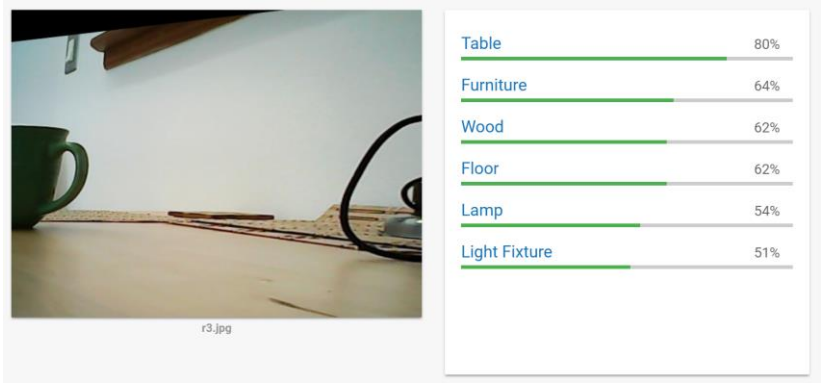
Husnu Melih Erdogan PhD Student - CS 510 - Computational Photography Final Project Report



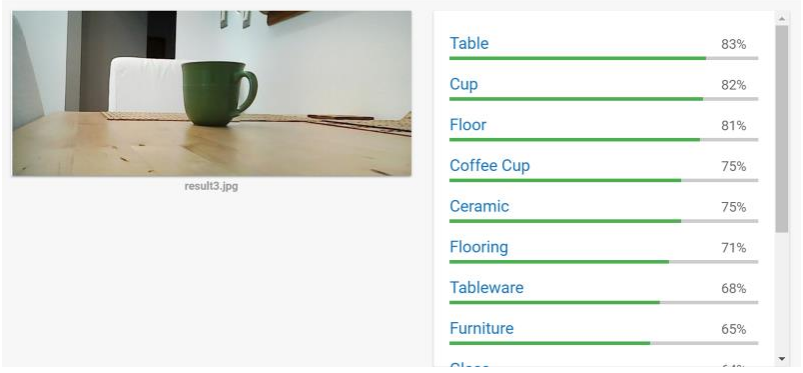
Evaluation:

In order to evaluate the system, I took a picture of a coffee cup with the eye cameras.

First, I uploaded the right eye image to Google Vision API to get results. It was able find the coffee cup.



Second, I created a panoramic image and uploaded to Google Vision API and get results. It was able to find the coffee mug and much more about the image.



Recourses:

Panorama and image stitching:

1. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7377344
2. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5586723
3. http://web.cecs.pdx.edu/~fliu/courses/cs510/project.html (Panorama)
4. https://courses.engr.illinois.edu/cs498dwh/fa2010/lectures/Lecture%2017%20-%20Photo%20Stitching.pdf
5. https://docs.opencv.org/3.4/d7/dff/tutorial_feature_homography.html
6. http://matthewalunbrown.com/papers/iccv2003.pdf

Blending:

1. https://github.com/cynricfu/multi-band-blending/blob/master/multi_band_blending.py
2. https://gist.github.com/royshil/0b20a7dd08961c83e210024a4a7d841a
3. http://embeddedfunda.com/blending-example-opencv-multiband-blending/
4. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_pyramids/py_pyramids.html

Cropping:

1. https://www.pyimagesearch.com/2015/03/09/capturing-mouse-click-events-with-python-and-opencv/

OpenCV:

1. https://www.pyimagesearch.com/2016/01/25/real-time-panorama-and-image-stitching-with-opencv/
2. https://docs.opencv.org/trunk/d8/d19/tutorial_stitcher.html
3. https://www.geeksforgeeks.org/stitching-input-images-panorama-using-opencv-c/
4. https://kushalvyas.github.io/stitching.html
5. https://www.pyimagesearch.com/2016/01/11/opencv-panorama-stitching/

Google Cloud Vison API:

1. https://cloud.google.com/vision/

Robot:

1. http://www.ros.org/
2. https://github.com/Interbotix/HROS5-Framework
3. https://github.com/mlherd/jimmy
4. http://www.trossenrobotics.com/HR-OS5