

Approach to Flood Depth Estimation

An intuitive way to estimate flood depth is by taking advantage of objects with known dimensions that are partially submerged. When you see a familiar object like a car in floodwaters, you can infer the water level by recalling the object's height, and comparing it with the actual height you're seeing in the water.

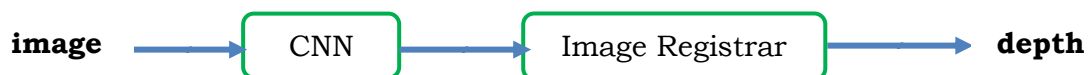
A person can instinctively follow a process similar to the following:

1. Recognize the portion of the scene covered by floodwaters.
2. Locate an object in the flooded area whose dimensions are known. This serves as the reference object.
3. Compare the visible portion of the object with its typical height to determine how much is submerged. During this process, the brain subconsciously accounts for differences in perceived size due to distance and viewing angle (adjusting for perspective).
4. The submerged portion represents the flood depth relative to that reference object.

The computational framework is thus designed to mimic this cognitive process using computer vision techniques.

The Framework

It consists of two major components: A convolutional neural network to perform step 1 and 2 described above, that is flood area and reference object detection; and an Image Registrar, to perform steps 3 and 4, that is adjusting for perspective distortions and convert submerged portion to a depth in real-world units.



The CNN will be implemented using the Mask R-CNN architecture (because its good for instance segmentation). This model will be trained to identify our reference object and the flooded area. Only reference objects in direct contact with the floodwaters will be considered for depth estimation.

Once the reference object is identified in the input image, it is matched with its corresponding representation in a reference image. This reference image contains the same object, but with an important addition: the real-world dimensions represented by each pixel (both width and height) are known. For example, if the width of a single pixel corresponds to 100 mm in real-world units, then moving horizontally by two pixels in the reference image would be equivalent to moving 200 mm in the real world. In simple terms,

the reference image is a calibrated reference space where real-world measurements can be performed.

Matching the detected reference object with its corresponding representation in the reference image will be done using the **Scale-Invariant Feature Transform** (SIFT). SIFT is used since the input image and reference image may have differences in perspective, lighting, and scale.

The correspondences established by SIFT will be used in computation of a **homography matrix**. The homography matrix is simply a 3×3 matrix that maps points from the input image to their corresponding locations in the reference image via a perspective transformation. This transformation accounts for differences in viewpoint, scale, and orientation between the two images. It will allow us to align the input image with the reference image. Since the reference image has a known real-world scale (i.e., each pixel corresponds to a specific measurement in millimeters), this alignment ensures that any flood depth measurements taken in the transformed input image are valid in real-world units.

With the homography established, the flooded region from the input image will be warped onto the reference image. The flood depth will then be estimated by counting the vertical number of pixels that obscure the reference object in the reference image.

Specifically, the top boundary of the flood will be identified in the reference space. The depth will then be computed as the vertical number of pixels from the top of the flood to the baseline of the reference object. Given that the reference image has a known scale (mm per pixel), the flood depth can be converted into real-world units (multiplying the number of pixels by the vertical unit of a pixel in mm).

Idealization

I have idealized your problem as simply finding how much of the reference object is obscured by a flood. In the images below, the text book represents our reference object, while the yellow box represents the flood. I have implemented the framework as describe above.

This is our reference image, the text book in this image is the reference object. Because I know the actual dimensions of this textbook, I calibrated the pixel dimensions in this reference image.



Now here is an image taken containing the reference object. The differences in scale, lighting and perspective are visible. The yellow box represents the flood.

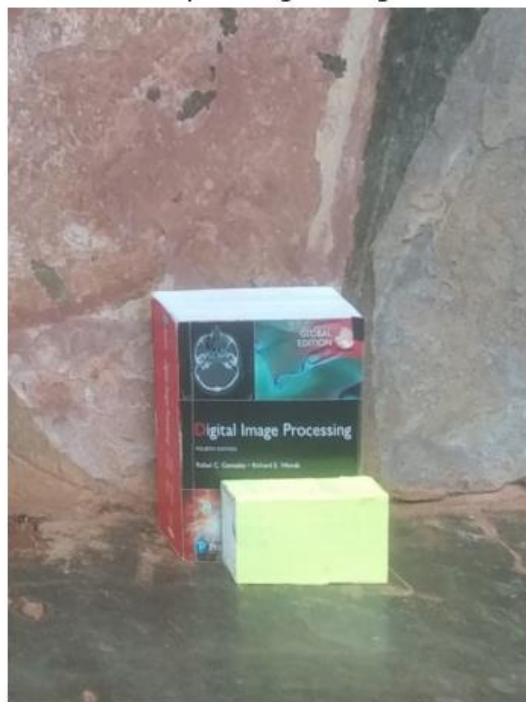


This image demonstrates the above process, identification of the reference object in the input image and matching it against the same object in the reference image (the lines in left image indicate correspondences). The middle image shows the registered input image. The last image demonstrates the flood (yellow box) obscuring the reference object. The red arrows on the last image indicate the flood depth, while the green arrow indicates the baseline from which depth is measured.

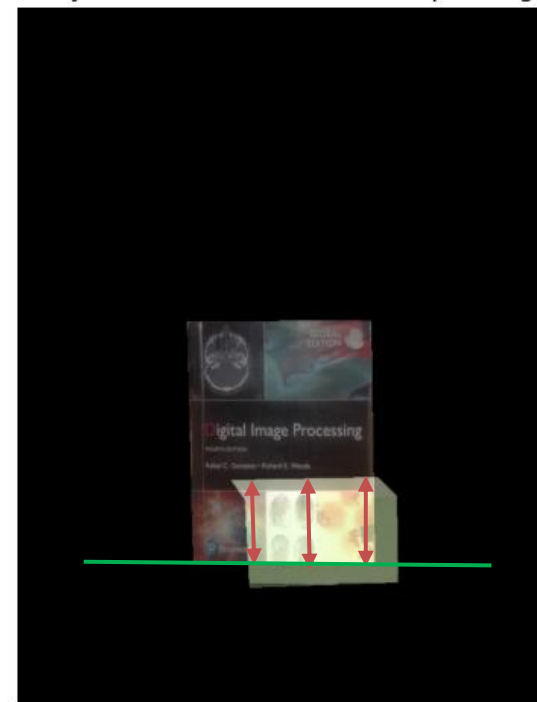
Feature Matches (SIFT + BFMatcher)



Warped Target Image

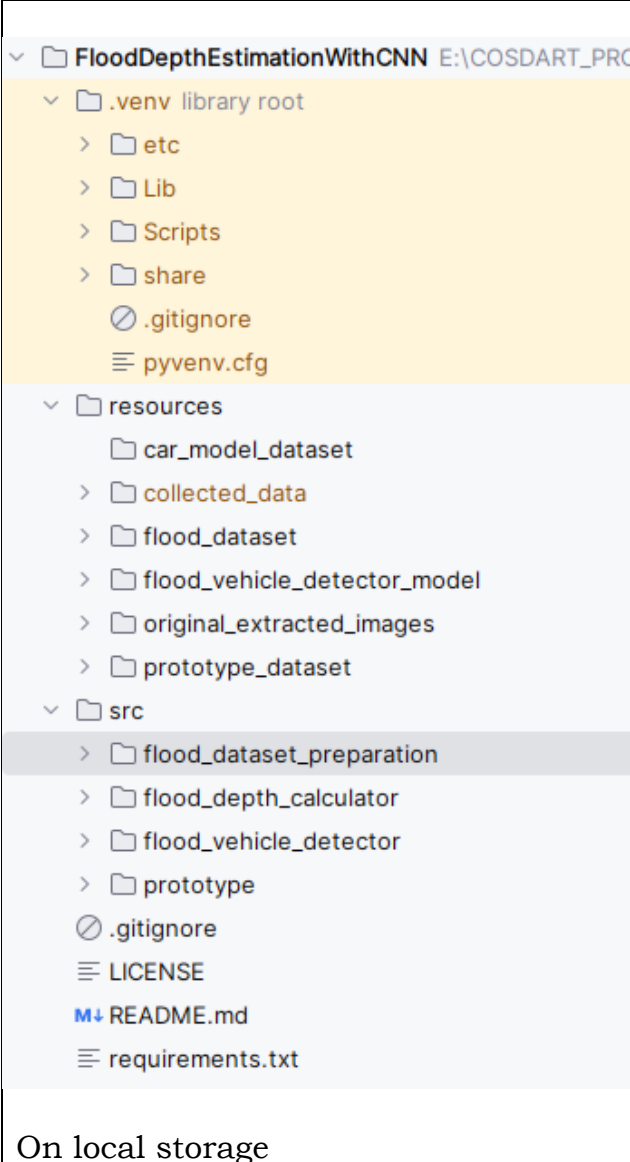
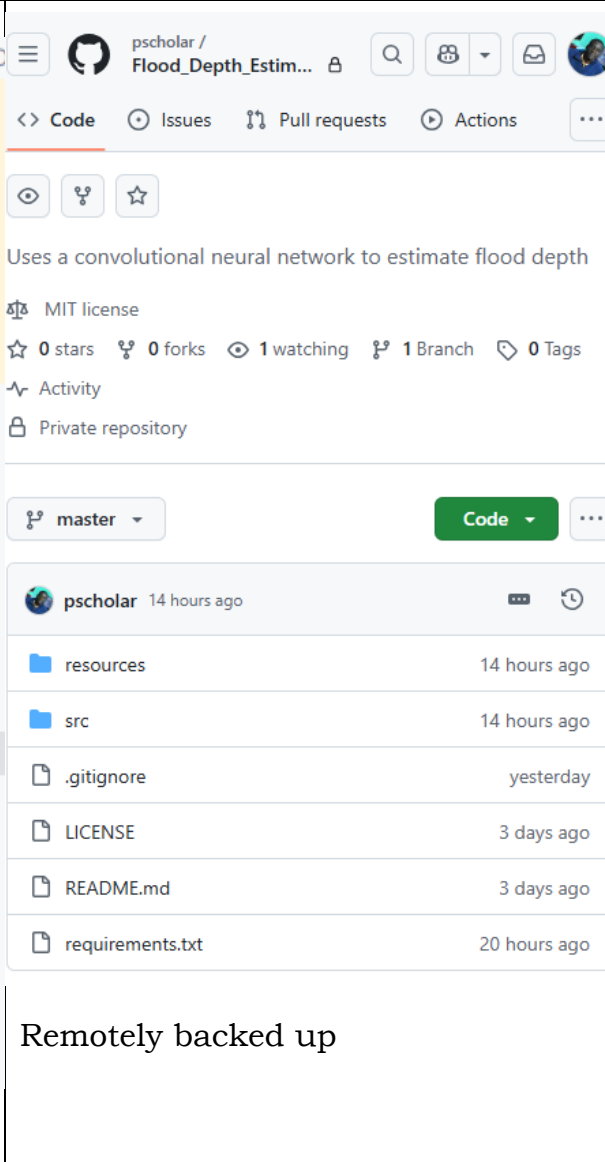


Overlay: Reference + Selected Warped Region



Where are We So Far?

Your project has already been setup, both locally and remotely (on my GitHub) so you guys don't have to worry about my setup getting faulty and losing your work. Its automatically back-up to the GitHub repository whenever I make changes to the project.

 <p>On local storage</p>	 <p>Remotely backed up</p>
--	---

The framework as described above has already been implemented in python and training of the CNN to identify the flooded region begun already, using the data you guys provided.

What is missing now are the reference images. We had earlier talked about using cars as reference objects, but even for cars there are very many models. Your research is both time and budget constrained. So, we shall do what all researchers do, "limit the scope for the project to be feasible". From cars, we shall select a single car model. From the videos you provided, the taxi is one of the most predominant cars in a flood. So, we shall use the taxi

as our reference object. But even for the taxi, there are very many models, there are these new ones called drones, they are less common, so I suggest we use the most common taxi shown in figure below.



This taxi will serve as our reference object, so we need to train the CNN to reliably identify it. Fortunately, there are many images of this taxi available online. You will need to search the internet and collect pictures of this taxi. Additionally, you can take your own photos, as this taxi is very common. Given that these taxis are frequently seen on the road and in taxi parks, obtaining at least 1000 images should not be difficult. Since the taxi is a 3D object, ensure that you capture images from many different angles to provide a comprehensive dataset. Note that these images are to be used for training the CNN only, we need to intentionally create the reference images.

Creating The Reference Images of the Taxi

You will need to take at least 36 views of the taxi. These have to be of high quality, so as we can accurately calibrate the reference images. Follow the criteria described below.

1. When capturing a side view, ensure that if a baseline were drawn at the bottom of the tires, it would appear as a perfectly horizontal line in the image. The same applies to the front and rear views—the bottom of the tires should align horizontally in the image.
2. The taxi should be in the foreground of the image, meaning you should not be too far away. The taxi should be clearly visible and well-defined.

3. Capture images where both the front and side views of the taxi are visible. Similarly, take images where both the rear and side views are visible.
4. Include images where only the side view is visible, as well as separate images where only the front or rear views are visible.
5. In images containing only the rear, side, or front views, ensure that horizontal edges on the taxi appear perfectly horizontal in the image. Vertical edges should be as close to perfectly vertical as possible.
- 6. Avoid taking plan views, we can't infer depth from images that are closely plan views of the reference object!!!!**

The above criteria means that for perfect calibration of the reference images, you guys won't take images of a random taxi on the street, you will have to get a taxi, place it on a level surface and take the pictures. You will have to also take actual dimensions of the taxi; the dimensions should be such that one can recreate the taxi in a drawing software.

Creating Ideal test data

The data you previously shared was taken randomly. However, you now need to intentionally capture images of a taxi in floodwaters, knowing that these images will serve as test cases for our framework. This means that as you take these photos, ensure that they closely follow the same criteria used for capturing the reference images. At least 50 carefully taken photos will be sufficient, but aim for more whenever possible. Take advantage of any opportunity to capture such images. Also try to take at least 50 images unintentionally. The good thing is that if I spot such a taxi in floodwaters, I can easily capture 10 ideal images just by shifting my position.

Also, enable your camera's location services while taking these images, and give permission for your camera to access your location. This information could be valuable in the future when scaling up the framework for real-world deployment. For example, in addition to estimating flood depth, geolocation data can be used to track where flooding occurs. This could help aggregate information to create a flood map for a given area e.g. Kampala, and keep records

How Do we Evaluate Our Framework?

We can easily evaluate the performance of the CNN component of the framework using mean Average Precision (mAP). But what of assessing the accuracy of the flood depth estimated by the image registration! Measuring flood depth along a moving taxi during heavy rain is both impractical and unsafe. This means that for each of the 50 images, you would need to

physically measure the corresponding flood depth along the length of the taxi in that view. This is an approach that is neither safe nor feasible.

Proposed Solution

As demonstrated in the images above and those to follow, the working principle of your problem is determining the portion of an object with known dimensions that is obstructed by another object. A practical way to validate the accuracy of our framework is to construct a scaled-down prototype of a vehicle moving through floodwaters in a controlled indoor environment. This would allow us to compare the estimated flood depths with ground-truth measurements in a safe and controlled setting.

Implementing this solution is no small task, it essentially requires replicating everything described above, just at a smaller scale. In other words, it's like executing your entire project a second time. **I can implement this solution for you if you're willing to facilitate it, as it is beyond the scope of your current budget.**

The accuracy measurements obtained from this scaled model will be representative of the performance at full scale. Let me know if you're open to this idea. Implementing this validation step would significantly enhance the completeness of your work.

What you need to read about

The following are from Digital Image Processing 4th Edition by Gonzalez and Woods.

1. Image Registration – Chapter 2, Section 2.6, page-number 103
2. Image Segmentation Fundamentals: Chapter 10, section 2.1, page-number 700.
3. Scale Invariant Feature Transform: Chapter 11, section 11.7, page-number 881.
4. Neural networks and deep learning: Chapter 12, from section 12.5-12.7, page number 931.

The Images that follow simply demonstrate how the framework works.

Reference Image with the reference object: (text book)

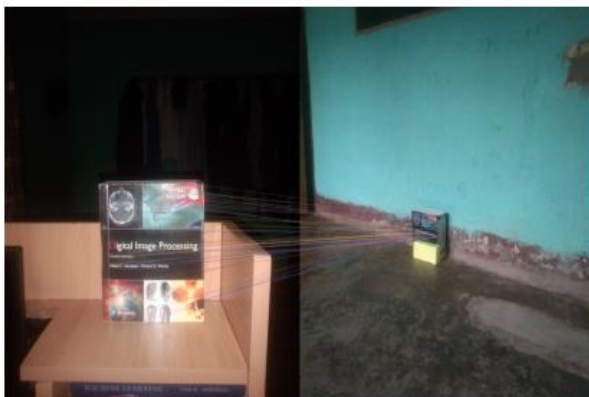


Image Taken from an arbitrary setting, but it contains the reference object.



After math of the process

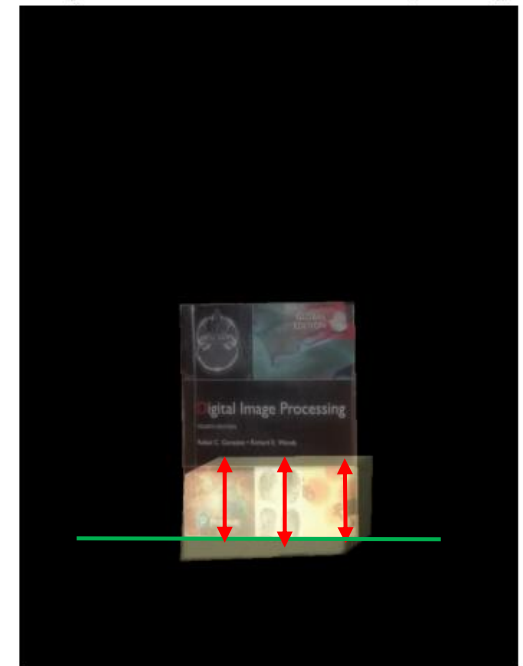
Feature Matches (SIFT + BFMatcher)



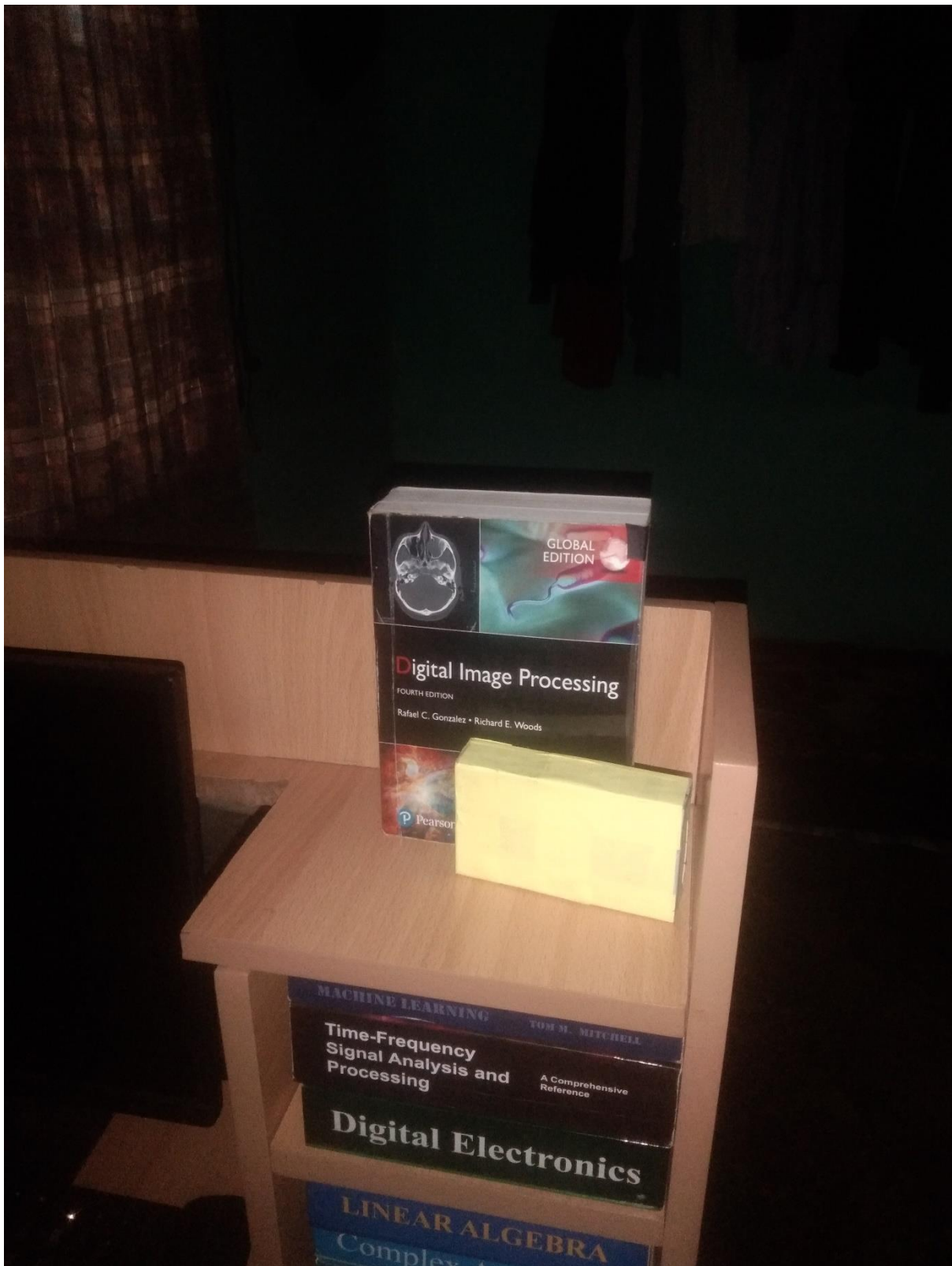
Warped Target Image



Overlay: Reference + Selected Warped Region

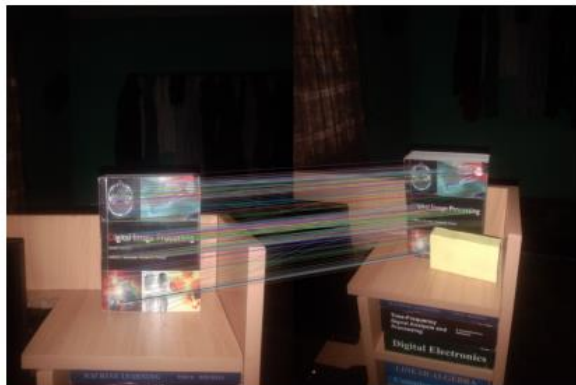


Another image containing the reference object in an arbitrary setting

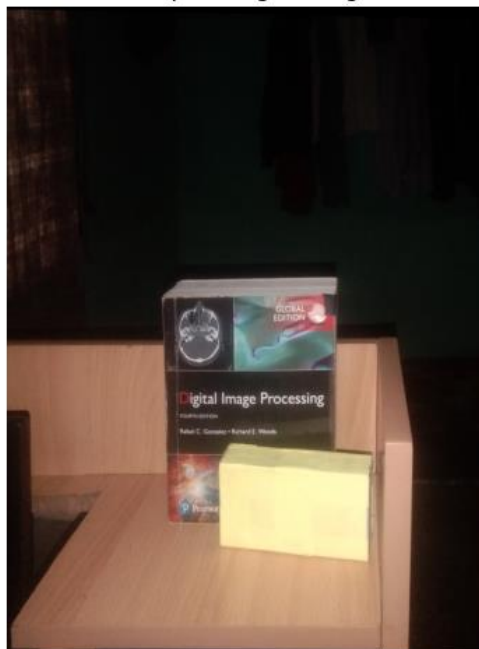


After math of the process

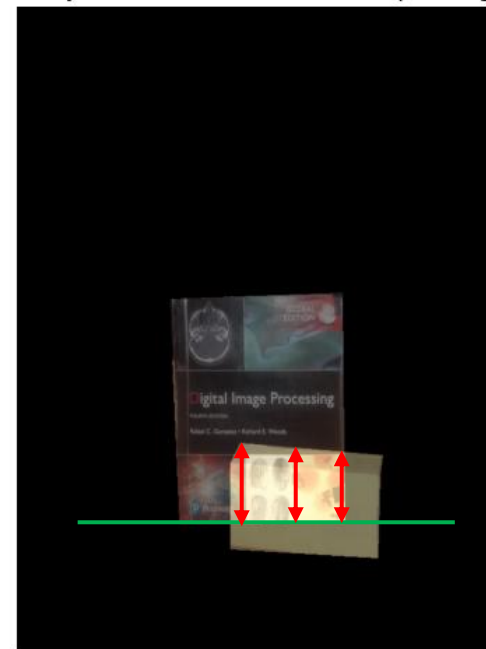
Feature Matches (SIFT + BFMatcher)



Warped Target Image



Overlay: Reference + Selected Warped Region



THE END