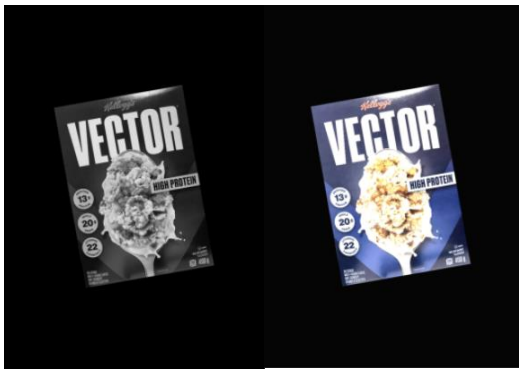
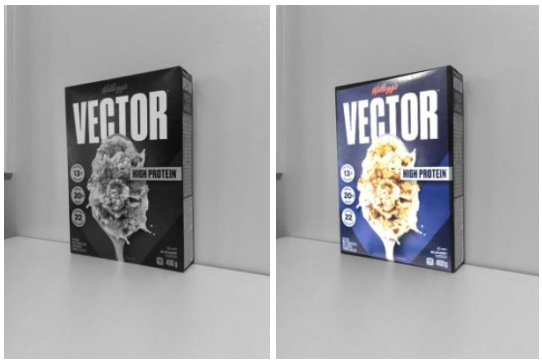




ELEC 474

Lab 4 – Object Detection

Lab 4 – Transformation Estimation

Kevin Huang and Michael Greenspan

Contents

1. Image Tranformations 1
1.1 Matching 1
1.2 Affine Transform 2
1.2 Perspective Transform 3
2. Submission 4

1. Image Tranformations

For this lab you will make use of a feature detection algorithm to detect objects in an **input image** based on a **reference image** of the object. Depending on which feature detection you use (e.g. SIFT or SURF) you might have to import the ***opencv_contrib*** library.

You can install the ***opencv_contrib*** library by typing into the Anaconda Prompt:

```
pip install opencv-contrib-python
```

Once your feature detection is installed perform your matching.

1.1 Matching

you will need to create a **Python function** which will return **key points**, **descriptors** as well as **Lowe's Ratio matches**. You can use **any** feature detection algorithm and you should follow this procedure for object detection:

1) Initialize two image inputs for your function.
2) Initialize your feature detection algorithm of choice.
3) Depending on your algorithm, detect and compute your **key points** and **descriptors**.
4) Using any **cv2** matcher (**cv2.FlannBasedMatcher()** is used here), find matches with your **descriptors**. When computing, set your matcher output **(k=2)** to output **two** possible matches for Lowe's Ratio Test.
5) Apply **Lowe's Ratio's** to filter you matches.
6) **return** your **key points**, **descriptors**, and **Lowe's Ratio matches**.

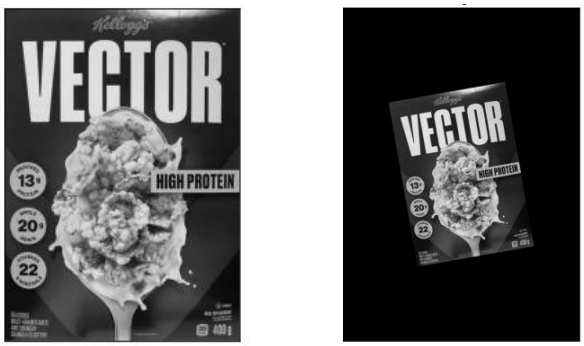
1.2 Affine Transform

For this portion of the lab you will find the *affine transform* from your **reference image** to a **target image,** which is a rotated and scaled version of the reference image. Once you find the affine transform, you will apply this transform your reference image and overlay it onto the target image. The basic recipe is as follows:

1) Load in your **reference image** as **grayscale**.

2) Rotate and scale your reference image to create the **target image**:
   - Find your **reference image's center** coordinates;
   - Define arbitrary **rotation and scaling** values;
   - Retrieve your **rotation matrix** for your image using **cv2.getRotationMatrix2D();**
   - Use **cv2. warpAffine()** to retrieve your **target image.**

As an example, below is a **reference image** ("cereal.jpg") and rotated and scaled **target image** :



a) Reference image b) Target image

Figure 1: Reference and target images

3) Use your **feature detection function** to retrieve your **key points**, **descriptors**, and apply **Lowe's Ratio matches**.

4) You need to format your points for use in OpenCV's transformation functions, this can done using this generic code snippet:

```
ref_pts = np.float32([kp1[m.queryIdx].pt for m in lowe_matches]).reshape(-1,1,2)
img_pts = np.float32([kp2[m.trainIdx].pt for m in lowe_matches]).reshape(-1,1,2)
```

5) Use **cv2.estimateAffinePartial2D()** to retrieve the affine transformation matrix.

6) Load in a colour version copy of the **reference image**.

7) Using the color reference image, use the **cv2.warpAffine()** method to apply the calculated affine transformation matrix onto the modified image to generate a new **target image**. The new **target image** will have a colour overlay of the cereal box, positioned at the recovered transformation.

Figure 2: Colour reference image, transformed with recovered affine transform

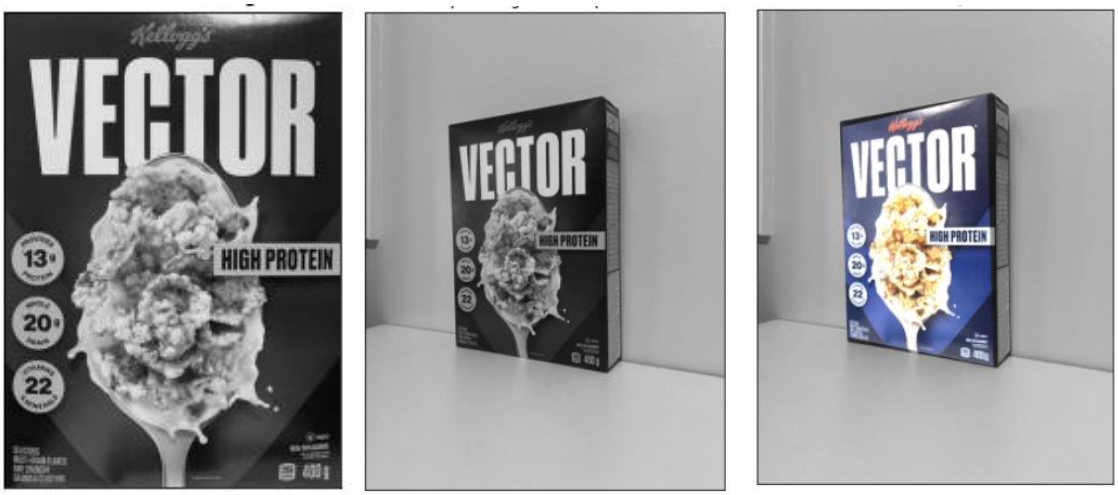
1.3 Perspective Transform

In Computer Vision, any two images of the same planar surface in space are related by a *homography*, which encodes not only translations, rotations and scaling, but also perspective transformations. For this lab you will need to calculate the homography from your reference image ("cereal.jpg") to your target image (e.g. "cereal_l.jpg") and apply the recovered perspective transform.

This can be done using the same basic recipe as described in Section 1.2, with the following differences:

- use **cv2.findHomography()** in step 5 (instead of **cv2.estimateAffinePartial2D()**) ;
- use **cv2.warpPerspective()** in step 6 (instead of **cv2.warpAffine()**).

Apply this procedure for the 4 images provided (i.e. "cereal_l.jpg", "cereal_r.jpg", "cereal_tl.jpg", "cereal_tr.jpg"). An example of the expected output for "cereal_r.jpg" is shown in Figure 3 below.



a) Reference image b) Target image c) Overlay from recovered homography

Figure 3: Reference image warped to target image from recovered homography

2. Submission

The submission for this prelab should include a .zip of:

- .ipynb file that includes:
  - Your code for **Affine** and **Perspective Tranformations.**
  - Tested your code with "cereal.jpg" **reference image** on four cereal **test images** ("cereal_l.jpg, cereal_r.jpg, cereal_tl.jpg, cereal_tr.jpg").
  - Your code should display images of your **reference image, test images,** and both overlaid **Affine** and **Perspective transforms**

Your code will be run in Jupyter Lab to test for functionality. The marking rubric is as follows:

Section	mark
1.1 Matching	0.5
1.2 Affine Transformation	1.5
1.3 Perspective Transformation	1.5
Correct submission format	0.5
Total:	**4**