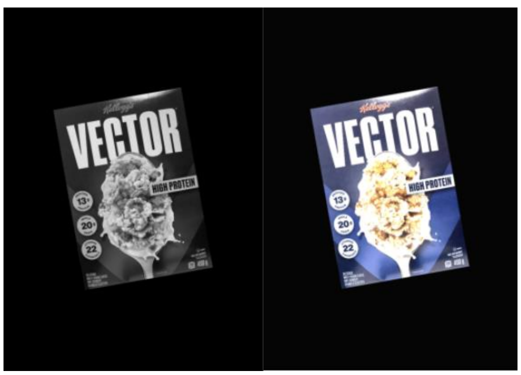


ELEC 474

Lab 4 – Object Detection

Lab 4 – Object Detection

Kevin Huang and Michael Greenspan

Contents

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 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 **rotated and scaled version** of your **reference image (two different images)**. Once you find the Affine Transform you will transform your **reference image** and overlay it onto your **rotated and scaled** version. Follow these steps:

1) Load in your **reference image** as **grayscale**.
2) Rotate your 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 **rotated and scaled** version of your **reference image**

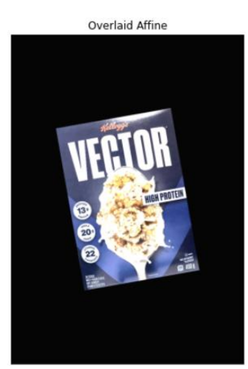
Reference image ("cereal.jpg") and output of rotated and scaled "cereal.jpg":

3) Use your **feature detection function** to retrieve your **key points**, **descriptors**, and **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 your Affine Transformation matrix.
6) Modify a **copy** of your **reference image** (e.g. change colour)
7) With your modified image, use **cv2.warpAffine()** to apply your Affine Transformation Matrix onto your modified image to retrieve your **transformed image**.
8) Overlay the **transformed image** onto your **test image**.

RGB version of **input image** ("cereal.jpg") onto rotated and scaled version of "cereal.jpg":



1.2 Perspective Transform

In Computer Vision, any two images of the same planar surface in space are related by a Homography. For this lab you will need to calculate the Homography from your reference image (“cereal.jpg”) to your test image (e.g. “cereal_l.jpg”) and apply a Perspective Transform onto a modified version of your **reference image** to overlay it onto your **test image**. This can be done from the following steps:

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

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

9) Use **cv2.findHomography()** to retrieve your Homography matrix.
10) Modify a **copy** of your **reference image** (e.g. change colour)
11) With your modified image, use **cv2.warpPerspective()** to apply your Homography Matrix onto your modified image retrieve your **transformed image**.
12) Overlay the **transformed image** onto your **test image**.

RGB version of **input image** (“cereal.jpg”) onto **test image** (“cereal_r.jpg”):

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**