Match Features between two Images using Brute-Force Matching

Experiment 1.0

Group: **Group 1615:**

- *1651059 - Do Thai Bao*
- *1651048 - Nguyen Ba Nhat Khanh*
- *1651022 - Tran The Loc*
- *1651065 - Ha Nhat Minh*

Lecturer: **Assoc. Prof. TRAN Minh-Triet**

Faculty of Information Technology
University of Science - Vietnam National University HCM

ABSTRACTION

Customers are the most important part of brand development, brands need to know how customer react to their product. Therefore, they spend a lot of money to improve customer experience. Base on that problem, our group think about using logo detection to detect logo in customers' pictures posted on social media, in that way company can analyze picture to understand how people react to their products.

INTRODUCTION

The problem statement of our group is to focus on logo detection. So, in the first experiment, we will test the accuracy of the Brute-Force Matching algorithm. Then we will observe the result to examine our hypothesis.

In order to understand how a brand logo can be detected in a picture, we choose to use the Brute-Force Matching algorithm from OpenCV to perform the experiment.

***Below is the detail explaining how this method matches features between two images* https://docs.opencv.org/3.2.0/dc/dc3/tutorial_py_matcher.html

*****Brute-Force Matching with ORB Descriptors***

Here, we will see a simple example on how to match features between two images. In this case, I have a queryImage and a trainImage. We will try to find the queryImage in trainImage using feature matching. (The images are /samples/c/box.png and /samples/c/box_in_scene.png)

We are using ORB descriptors to match features. So let's start with loading images, finding descriptors etc.

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

img1 = cv2.imread('box.png',0)          # queryImage
img2 = cv2.imread('box_in_scene.png',0) # trainImage

# Initiate ORB detector
orb = cv2.ORB_create()

# find the keypoints and descriptors with ORB
kp1, des1 = orb.detectAndCompute(img1,None)
kp2, des2 = orb.detectAndCompute(img2,None)
```

Next we create a BFMatcher object with distance measurement cv2.NORM_HAMMING (since we are using ORB) and crossCheck is switched on for better results. Then we use Matcher.match() method to get the best matches in two images. We sort them in ascending order of their distances so that best matches (with low distance) come to front. Then we draw only first 10 matches (Just for sake of visibility. You can increase it as you like)

METHODS

Brands	Number of train Images
BMW	13
Citroen	14
DHL	22
HP	7
Heineken	10
Pepsi	18
Adidas	10
Total	**94**

- First we download **queryImage** of 7 different brands: *BMW, Citroen, DHL, HP, Heineken, Pepsi* and *Adidas* from Google.
- Then with each brand we collected **trainImage** from **flick_logos_27_dataset**. The total number of train images is 107, the detail of each brand train images are listed below
- Then we use Feature Matching algorithm to find key points between **queryImage** and **trainImage**.

RESULTS and DISCUSSION

Experiment results show that this algorithm has some disadvantages that need to be fixed to improve the accuracy. Specifically, the results are inaccurate in four main cases:

1. small logo
2. uncompleted logo
3. blur logo
4. leaning logo

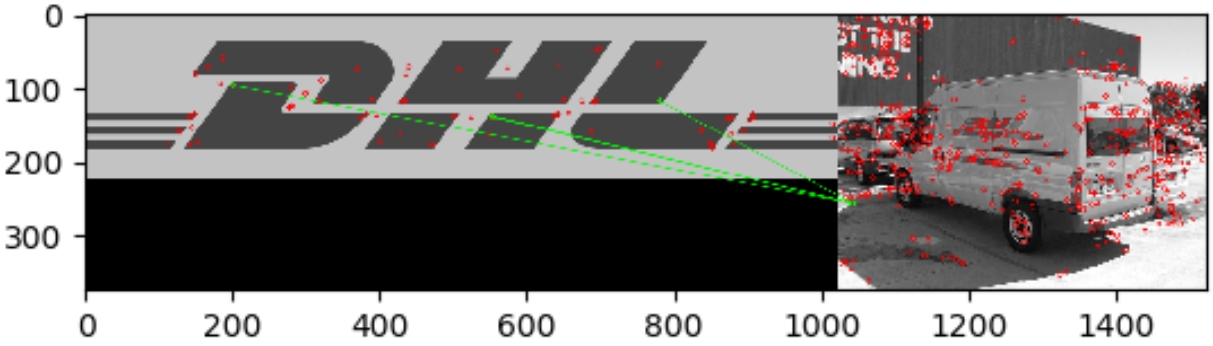

Figure 1: Result for small logo

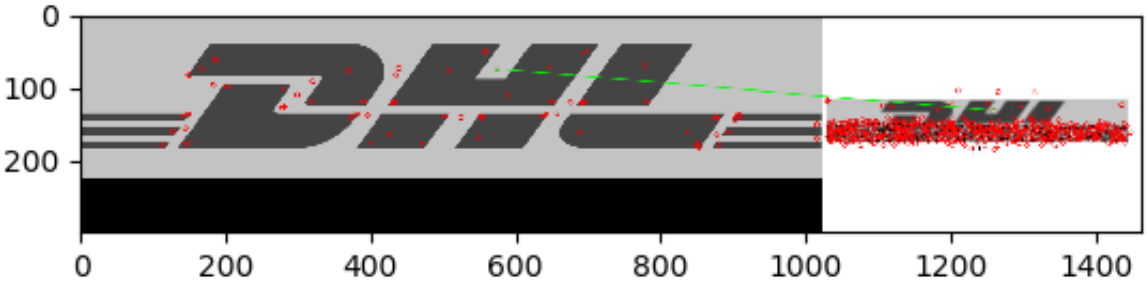

Figure 2: Result for uncompleted logo

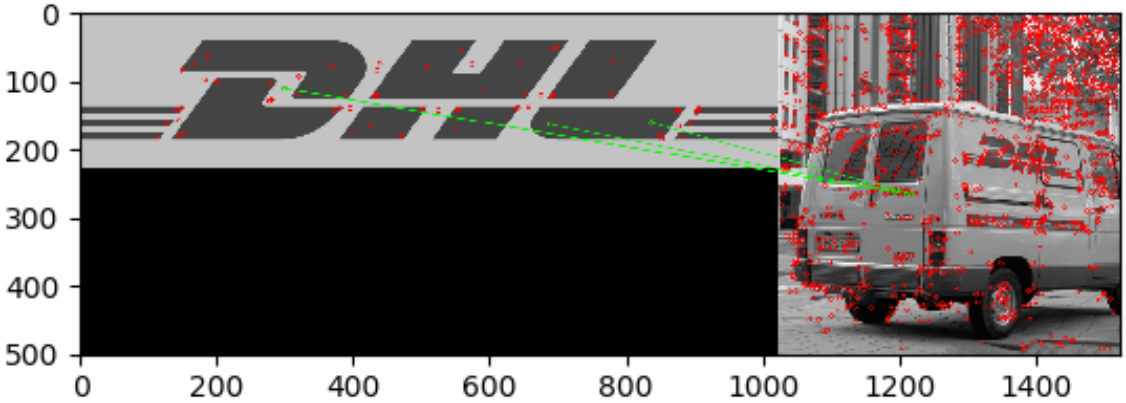

Figure 3: Result for leaning logo

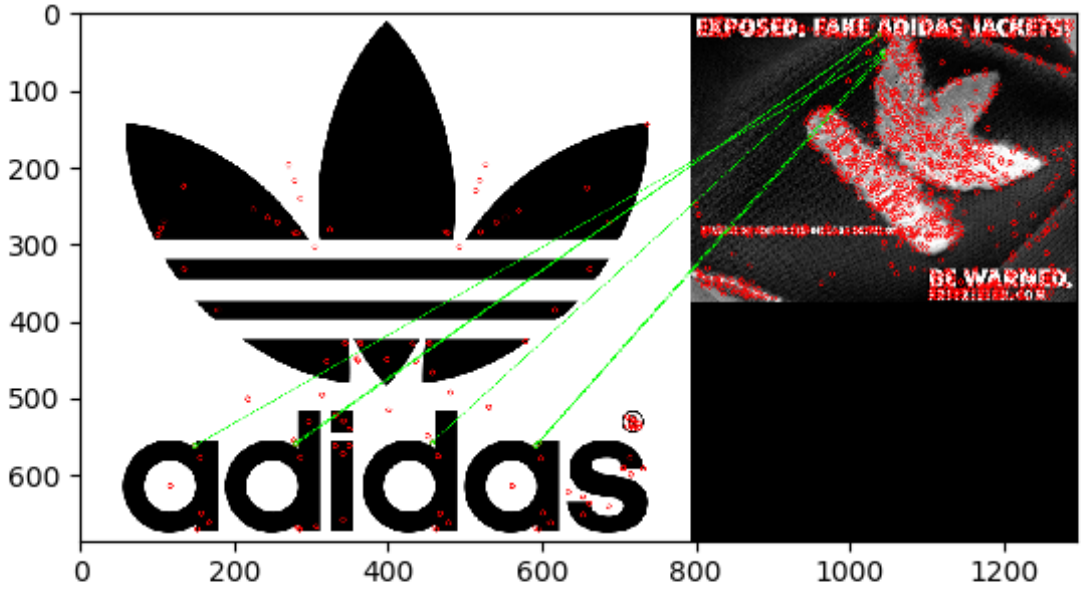

Figure 4: Result for blur logo