By Sameeh Jubran

Algorithms and Applications in Computer Vision

FINAL PROJECT

[](https://github.com/sameehj/computer-vision---template-matching-)FINAL PROJECT

COMPuTER VISION

Contents

[OPTION 2 – TEMPLATE MATCHING 2](#_Toc428579245)

[Configurations and Usage: 2](#_Toc428579246)

[A. Template matching methods implemented in this project 3](#_Toc428579247)

[i) SIFT for template matching: 3](#_Toc428579248)

[ii) Set of filters (generated for a given template) with convolution: 6](#_Toc428579249)

[*Image segmentation using grabcut* 8](#_Toc428579250)

# OPTION 2 – TEMPLATE MATCHING

# PART 1

# Configurations and Usage:

Set the configurations in the configurations.m file as follows: set the path to the template image and the path to the directory where other jpeg/jpg images are, those are the images that contain (or don't) the template in them. next you need to run one of two files: sift\_template\_matching.m or brute\_force\_template\_matching.m.

#### setting sift\_template\_matching.m configurations for better results:

* **changing the RANSAC threshhold and number of iterations:**

You can do so by changing the ransac\_transformation parameters.

#### setting brute\_force\_template\_matching.m configurations for better results:

* **changing the convolution result threshold:**

The convolution result is between -1 and 1. Typically you want to aim to 0.6 - 0.7, but with different templates those values can be reduced. You can do so by changing the parameter to the applyFilters function.

* **changing the set of filters:**

You can change the set of filters that is created from the given template by changing the inside parameters of the function temlateFilters.m. you can do so by changing the resolution of the angle or the different filter size for each angle.

* **changing the Gaussian:**

the Gaussian that is applied to the template before the convolution occurs, you can do so by changing the Gaussian parameters in the file brute\_force\_template\_matching.m.

## Template matching methods implemented in this project

* + 1. SIFT for template matching.
    2. Set of filters (generated for a given template) with convolution.  
       \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### i) SIFT for template matching:

the implementation that was used is SIFT and RANSAC for finding a match for a given pair (template and target image), images with less than 4 matches - 4 corresponding points, 8 total - were ignored because homography transformation has 8 DOF.

**Algorithm:**

1) Load images and template image.

2) Run SIFT and extract the features from each image and the template.

3) Match the features for each image with the template features.

4) Run RANSAC:

4.1 choose 4 random points.

4.2 calculate homography.

4.3 count inliers.

4.4 repeat 4.1 until MAX ITERATIONS.

4.5 return homography with MAX number of inliers.

5) Apply the homography for the template and draw a box on the image.

**Pros:**

1) Fast.

2) Robust to scale, rotation.

3) Can handle significant changes in illumination.

4) Fast and efficient.

**Cons:**

1. no partial detection.

**Results:**

This is a sample of a good result, however the segmentation wasn’t that good.

****

This is an example of a perfect result:



This is an example of a very bad result, sift didn’t detect the stop sign but a partial portion of it:



**For more results please check the attachment section at the end of this document**

### ii) Set of filters (generated for a given template) with convolution:

For a given template a set of filters was calculated in various sizes and various rotations.

**Algorithm:**

1) Load images and template image.

2) Creating different angles of the template image in different sizes.

3) Applying gaussian filter to the set of images before the applying the convolution.

4) Using convolution between the the filter set and the set of images.

5) Searching for values that are above a given threshold, those are the matches.

6) Draw a box on the image.

**Pros:**

1) Robust to scale, rotation.

2) Finds more than one match in a given image that contains the template in different sizes.

**Cons:**

1) Slow.

2) Computationally expensive.

3) no partial detection.

**Results:**

This is a perfect result:



We can see that the template matching failed when there was very similar color in the picture to the template:



**Comparing results and analyzing results:**

We can see that sift can has an advantage when it comes to finding matches with pictures who has intensities that are close to the original template. However in a typical images with no similar dmoninat colors that are close to the template the “brute force” approach performed better ( better matches), more over in the “brute force” approach detects more than one template in the image. Both fail when it comes to morphed template in the image but SIFT fails less, for example the following image wasn’t matched by either:



# Image segmentation using grabcut

**GrabCut** is an image segmentation method based on graph cuts.

Starting with a user-specified bounding box around the object to be segmented, the algorithm estimates the color distribution of the target object and that of the background using a Gaussian mixture model. This is used to construct a Markov random field over the pixel labels, with an energy function that prefers connected regions having the same label, and running a graph cut based optimization to infer their values. As this estimate is likely to be more accurate than the original, taken from the bounding box, this two-step procedure is repeated until convergence.

After getting the image with the defining box of where the template was found, the grab cut algorithm performs background removal for the image. As we can see above, some results were great! Others failed miserably.

**Pros:**

1. Very good result with isolating background.
2. Minimal user interaction required (only provide a box and gamma parameter).

**Cons:**

1) Very slow.

2) Computationally expensive.

# Future Enhancements

* All the implementation above can be boosted if performed on GPU.
* The brute force approach can be enhanced after some research so we can have range of values that we know of.
* The convolution in the brute force approach is performed on grayscale images, doing it with colors will not boost the performance but may enhance the results.

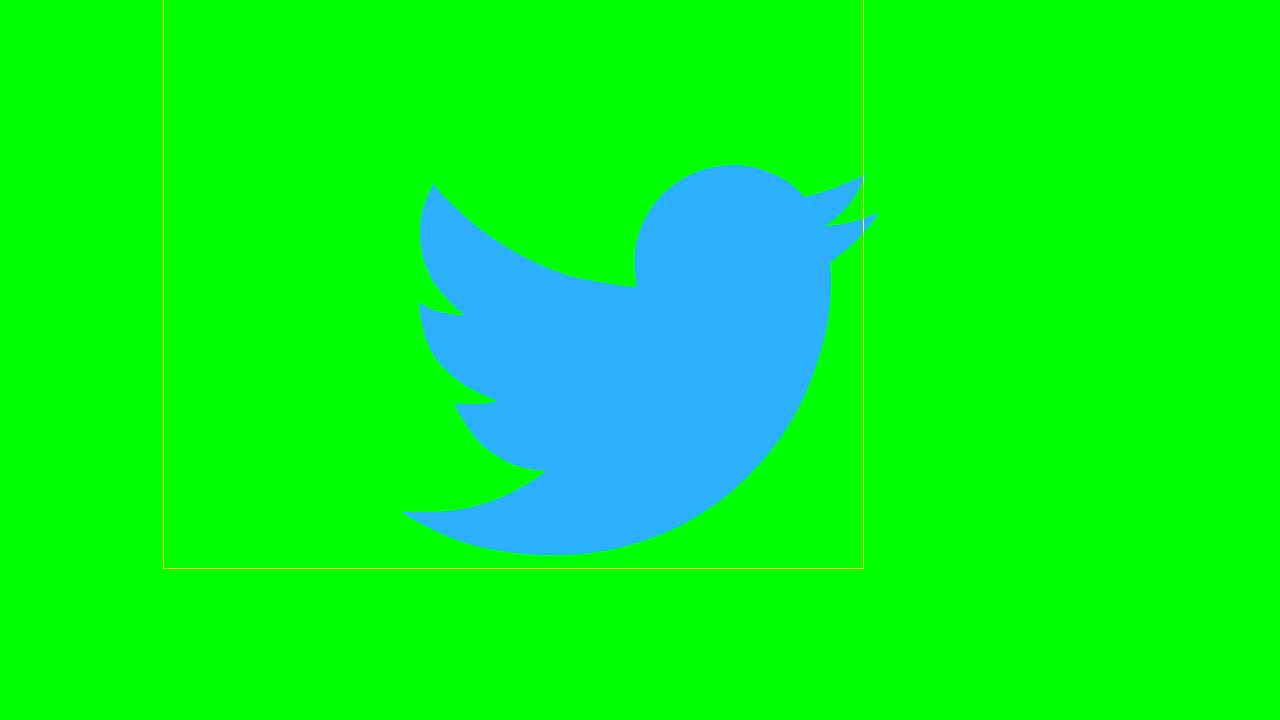
# PART 2

# Configurations and Usage:

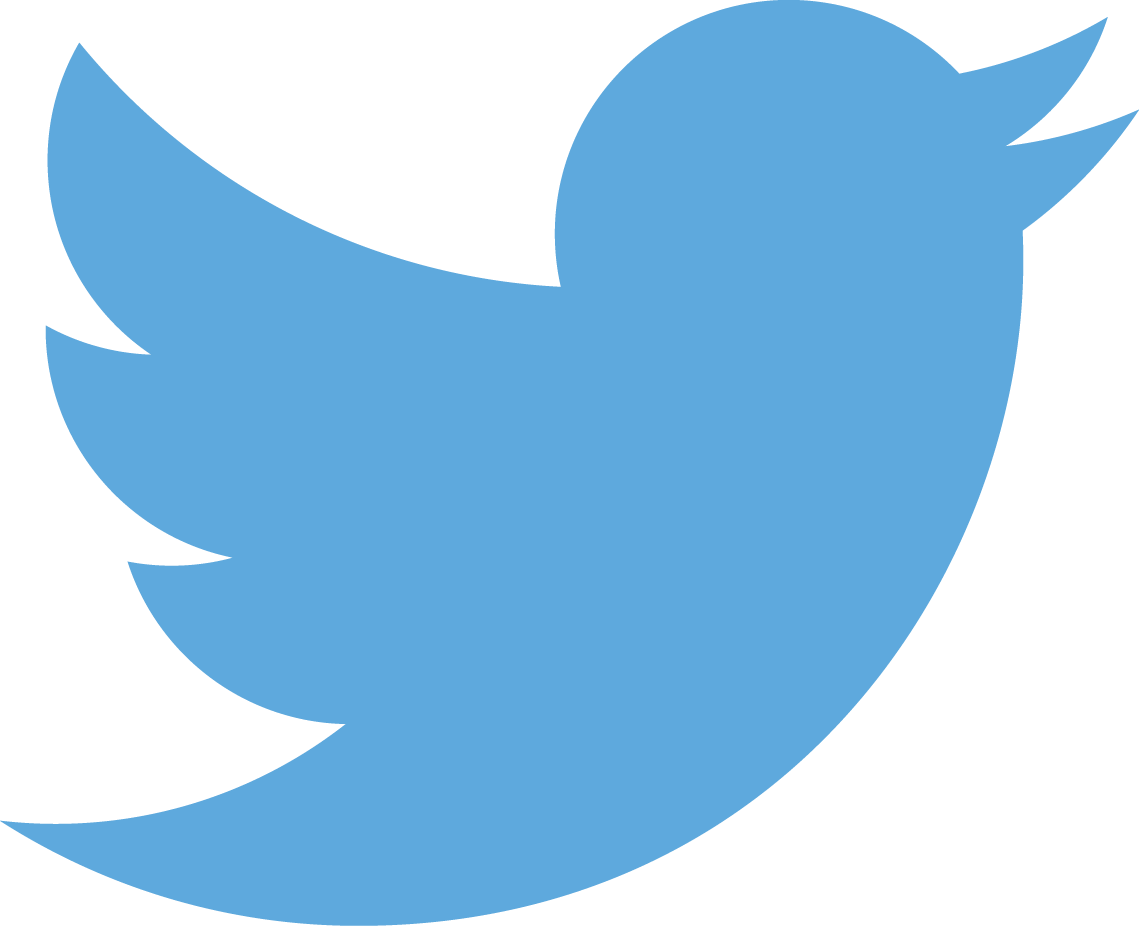
Set the configurations in the video.m file as follows: set the path to the template , replacement and the video which contains the template. Define how many frames the video should be initially skipped and for how many frames should it run.

In this PART, I took a video of twitter’s bird logo with green screen background and replaced the bird with lacoste logo using the bruteforce approach from PART A. after replacing the bird with the crocodile logo, the logo had to fit in the video thus the white background of the logo was changed to green. All of that was done in the video.m , change1colorOnRGBToAnother.m and insertImage.

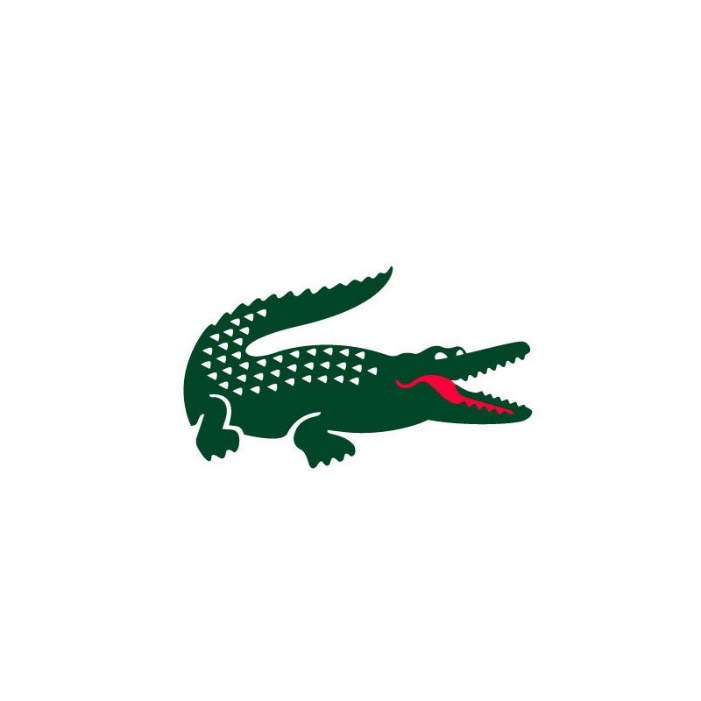
video:



Template:



Replacement:



**Algorithm:**

1. For each video frame do:
   1. Find a rectangle where the template is in the current video frame using the brute force approach as seen in PART 1.
   2. Resize and rotate the replacement image.
   3. Change main colors in the image to adapt it to the new seen.
   4. Plant the image in the frame
   5. Do so to the next frame.
2. Create the video.

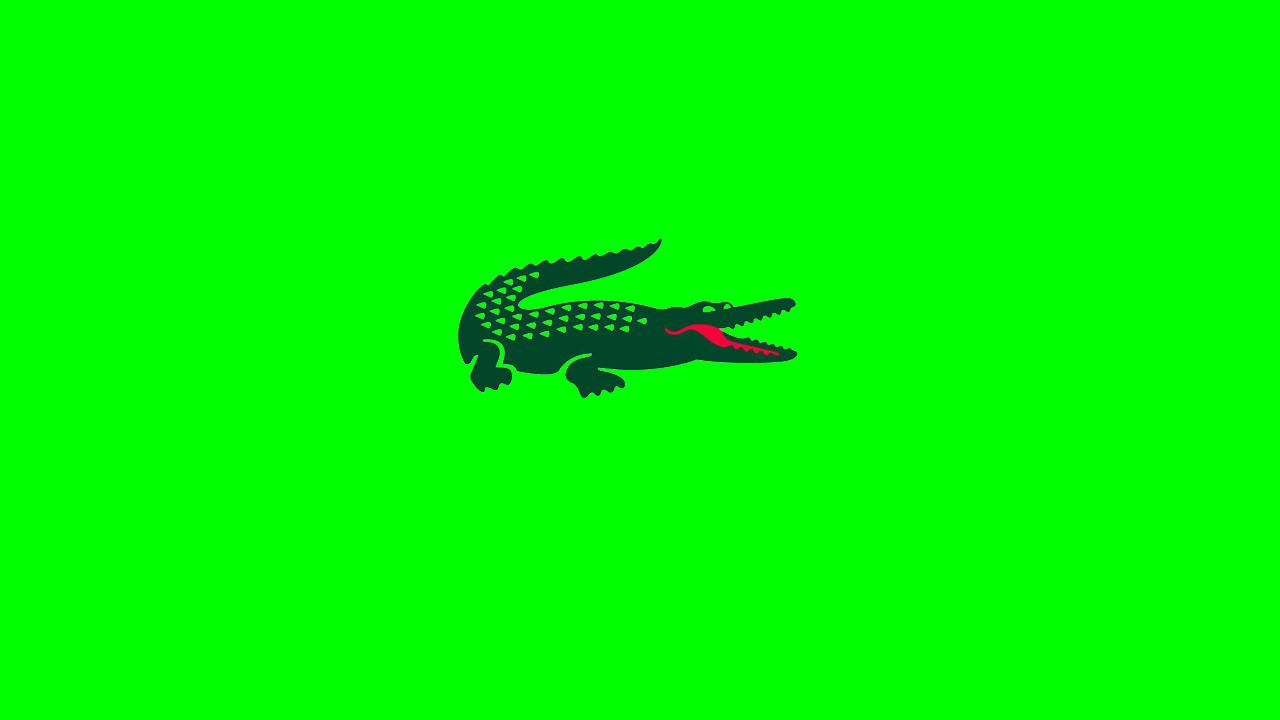
**Pros:**

1. Very good result!
2. Fast
3. Better than sift (sift would fail in this kind of footage there are very few descriptors)

**Cons:**

1. Getting the parameters to be perfect is time consuming.
2. This implementation is not generic at all.

**Results**:



As you can see the result is perfect, but then again the background is solid green and this part made it easier to merge the image.

You can find the video in the video directory of this project.

# Scripts Sources:

##### Sift:

##### <http://www.robots.ox.ac.uk/~vedaldi/code/sift.html>

##### Color changer:

##### <http://www.mathworks.com/matlabcentral/fileexchange/29137-change-1-color-to-another-of-an-image>

##### Grabcut:

##### https://github.com/xiumingzhang/grabcut