Digital Imagination

Project for course

Automation, Representation, Transmission

Yan Ka Man, Kevin

3028693

Zhou Fangli , Sam

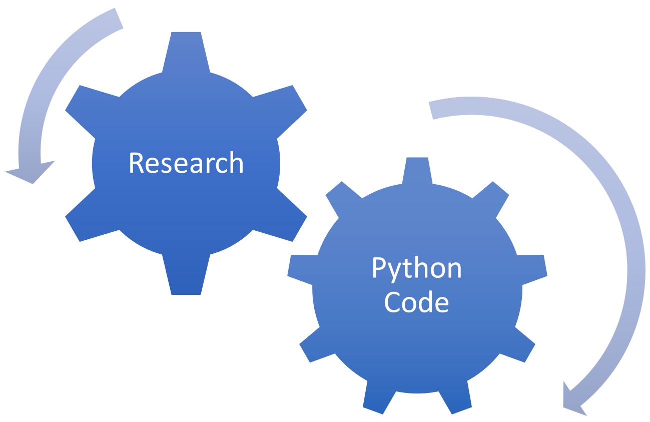
3028691

Goal

The goal of the project is trying to explore the possibility of whether the computer can function like human being, use human-like imagination and instinct to transfer the contemporary daily objects in the pictures to the ancient characters. In this case, we tried Pictogram to do the experiment. We believe that our attempt can answer our question.

Documentation method

To achieve and insist our core concept which has already been considered from the last semester, we started our experiments by searching different current technological methods. We knew that the project might be a bit ambitus and need further untouched computer knowledge, but we still set it as a long-term research of computer science in the field of Artificial Intelligence, machine learning and Deep Dream Development. The ultimate goal is to lead to a question about the comparison of the imagine ability between ancient human being and computer, as well as to distinguish the fact of how computer and natural being think. As a result, we will use a half theoretical, half practical form to document and present our work. We wish you can enjoy our journey.



Experiments and preparation

In short, our technological goal was to detect any daily objects from an image, try to enable the computer can associate any graphical information (certain patterns, colour or shape) with a cluster of ancient pictograms. Then represent the graphic to us through its “Imagination”.

Methods below we have tried to apply:
(All the coded used integrated into the ‘Experiment’ file on Github.

1. OpenCV, Feature Matching
2. Haar Cascade training
3. Amazon’s Label matching system
4. Deep Dream - Caffe
5. Deep Dream online generator (we decided to adapt)

1. OpenCV, Feature Matching

We all looked deep into each method we found online, initially we learned the OpenCV, which is a powerful visual library providing multiple graphical manipulation functions. Feature Matching was the one we found it quite suitable for our project goal. It can compare two images and match the objects look like almost the same.

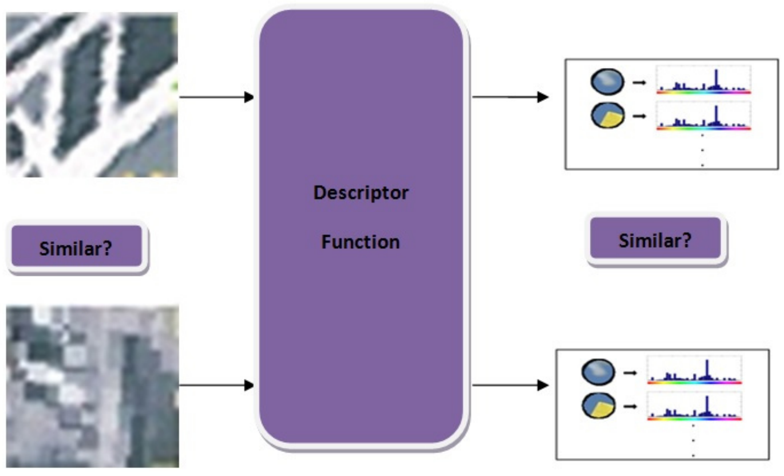
Feature Matching includes

ORB detector:

1. A sampling pattern: where to sample points in the region around (for example, use Harris corner measure to find top N points among them).
2. Orientation compensation: It aims to find the centroids of the patch to measure the orientation of the key point.
3. Sampling pairs: the pairs to compare when building the final descriptor.

Descriptor:

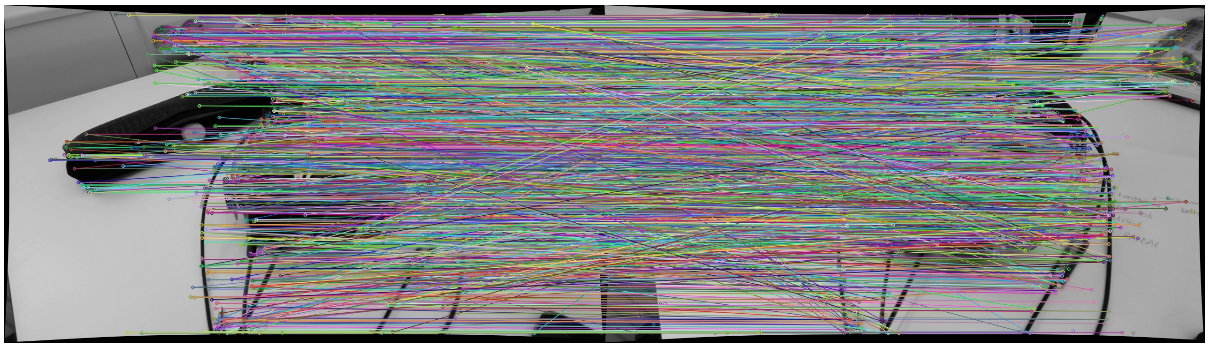
1. Compare the key points between the two images to find matches.
2. Use the matches to find the general mapping between the images (For our project, we use homography mapping to deal with the perspective warp)
3. Apply the mapping on the first image to align it to second image.



The above illustration shows that Descriptor is actually objectively transfer the patch of the image to data, then comparing to similarity of the two arrays of the data.

To do so, we need to measure the distances among key points by Brute Force Algorithm, which proves that if the distances among key points on two pictures are the similar, hence, the actual visual pattern will be probably similar too.
(Brute Force algorithm explanation:
https://www.youtube.com/watch?v=QD9zkrcMoOg

Be that as it may, follow its principle, if we can just adjust/lower the accuracy of the matching. It theoretically fulfils the goal of our project. We also attached our code which was amended in GitHub. However, the outcome of the generate image seems not like in our expectation.



2. Haar Cascade, OpenCV

For our interest to discover more about our concept about the ancient intelligence and the contemporary artificial intelligence. We also do an experiment of making our own Haar Cascade, which is to train our computer to recognize certain images or objects by feeding large amount of data, then the computer supposes to detect the objects quickly and accurately.

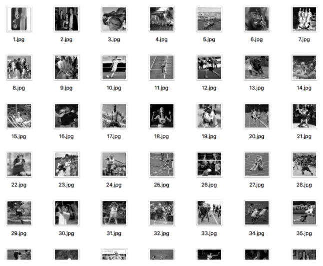
Haar Cascade was widely used to build up the technique of face recognition and it was developed by Intel. It was open-sourced recently on the Internet for people to play around but not for commercial usage.

(e.g. Face Swap)

Next, we tried to train the computer to recognize Kevin's watch:

Train our own Haar Cascade for our project

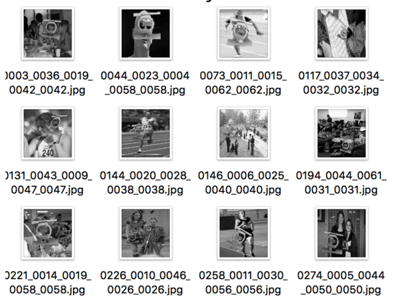
Step 1 Collect negative or irrelevant image database. (The more images we have, the more accuracy of object detection)



Step 2

Create samples which contain the positive and negative pictures.

Create description files, vector files which indicate the x, y and width and height within the images.

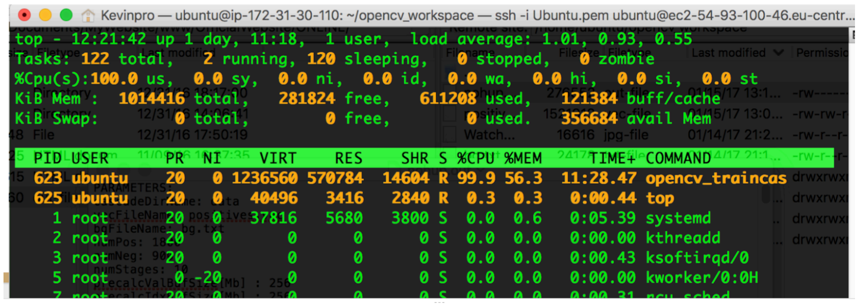


)003_0036_0019_0042_0042.jpg
0044_0023_0004_0058_0058.jpg
0073_0011_0015_0062_0062.jpg
0117_0037_0034_0032_0032.jpg
)131_0043_0009_0047_0047.jpg
0144_0020_0028_0038_0038.jpg
0146_0006_0025_0040_0040.jpg
0194_0044_0061_0031_0031.jpg
)221_0014_0019_0058_0058.jpg
0226_0010_0046_0026_0026.jpg
0258_0011_0030_0056_0056.jpg
0274_0005_0044_0050_0050.jpg

Step 3, Train the Cascade

Finally we train the computer to learn those samples (~2000 pictures) to create xml files (contain data of image features)

Normally, the learning processes should have at least ten stages and of course better CPU RAM … to shorten the waiting time…



Kevinpro — ubuntu@ip-172-31-30-110: ~/opencv_workspace — ssh -i Ubuntu.pem ubuntu@ec2-54-93-100-46.eu-centr...

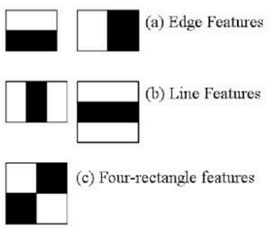
```
top - 12:21:42 up 1 day, 11:18,  1 user,  load average: 1.01, 0.93, 0.55
Tasks: 122 total,   2 running, 120 sleeping,   0 stopped,   0 zombie
%Cpu(s):100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1014416 total,   281824 free,   611208 used,   121384 buff/cache
KiB Swap:        0 total,        0 free,        0 used.   356684 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  623 ubuntu    20   0 1236560 570784  14604 R 99.9 56.3  11:28.47 opencv_traincas
  625 ubuntu    20   0   40496   3416   2840 R  0.3  0.3   0:00.44 top
    1 root      20   0   37816   5680   3800 S  0.0  0.6   0:05.39 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.00 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:00.43 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
```

Step4

run the xml file in OpenCV python, the machine should recognize the objects were learned.

The algorithm uses the Viola Jones method of calculating the integral image and then performing some calculations on all the areas defined by the black and white rectangles to analyze the differences between the dark and light regions of a face.

https://www.youtube.com/watch?v=hPCTwxF0qf4

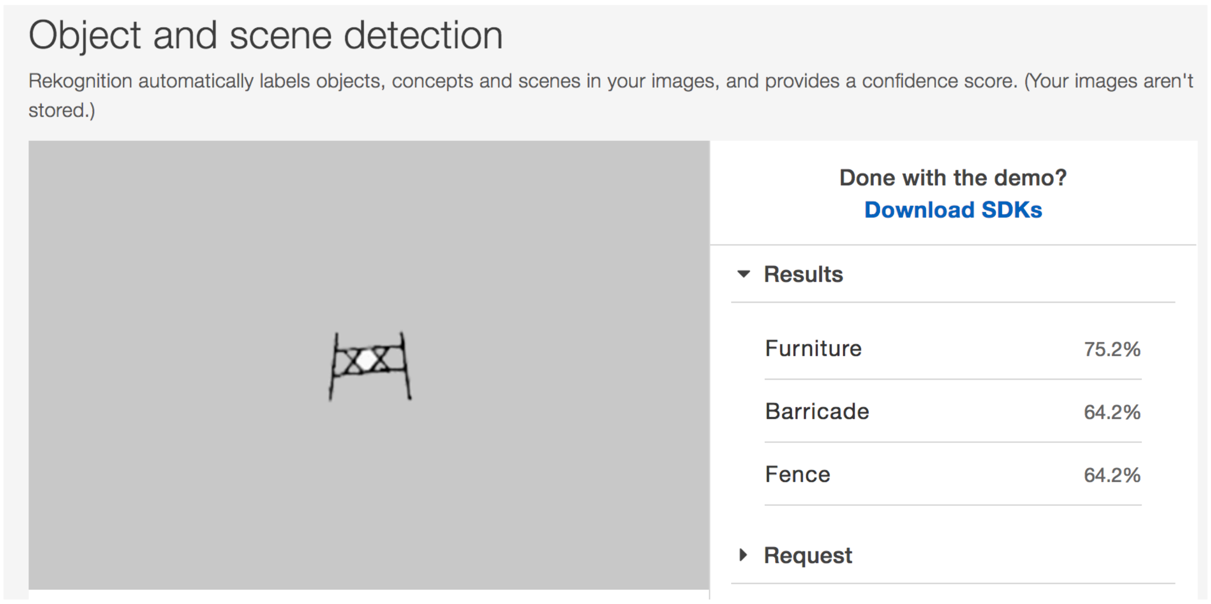


We gave up continue adopt this method to develop our project, because just training one object need almost 12 hours through our hardware, it costs too much time.

#*Considered our abilities of computer programming knowledge, we eventually decided to look for some premade product invented from some famous and matured technology companies like Google and Amazon. They recently keen to put their open source code online for public, we think that it might be a more accessible approach for our project.*

3. Amazon Web services's product – Amazon Rekognition

This is one of the product they provide from the Amazon Web Service. It is designed to use deep learning technology to analyse billions of images daily. Rekognition automatically labels objects, concepts and scenes in our images, and provides a confidence score. Amazon provide the users the trial version of their artificial interllegine one year for free. We also tried to upload the ancient pictogram we collected at the moment to try, it works quite well, and we did find out little linkage of between the image and the labels.


Object and scene detection

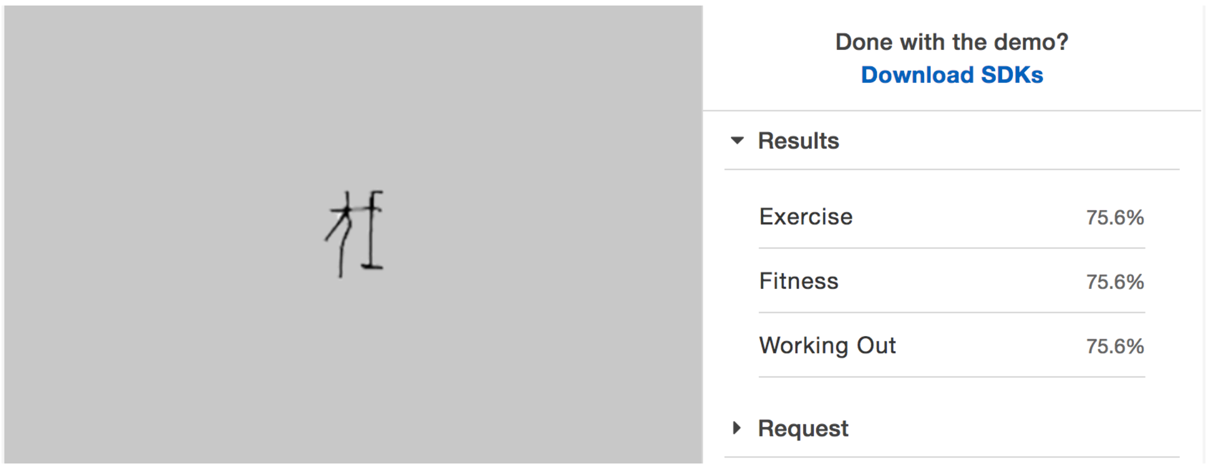
Rekognition automatically labels objects, concepts and scenes in your images, and provides a confidence score. (Your images aren't stored.)

Done with the demo?
Download SDKs

Results

Furniture	75.2%
Barricade	64.2%
Fence	64.2%

Request



Done with the demo?
Download SDKs

Results

Exercise	75.6%
Fitness	75.6%
Working Out	75.6%

Request

It is surprising that the label results we got was quite relevant to their linguistic meanings behind the Pictograms. Like the Chinese “網” was evolved by the first pitogram example, which means fence in English. We can see that the meaning exactly is on the third result. And the second word is “伐”, which is a verb means assaulting something. Then we also can find out those three results both are quite similar with the meaning of the original meaning of the pictogram. We can further prove that the pictogram has strong connect with the shape of the object itself through an objective perspective, the Amazon Rekognition system.

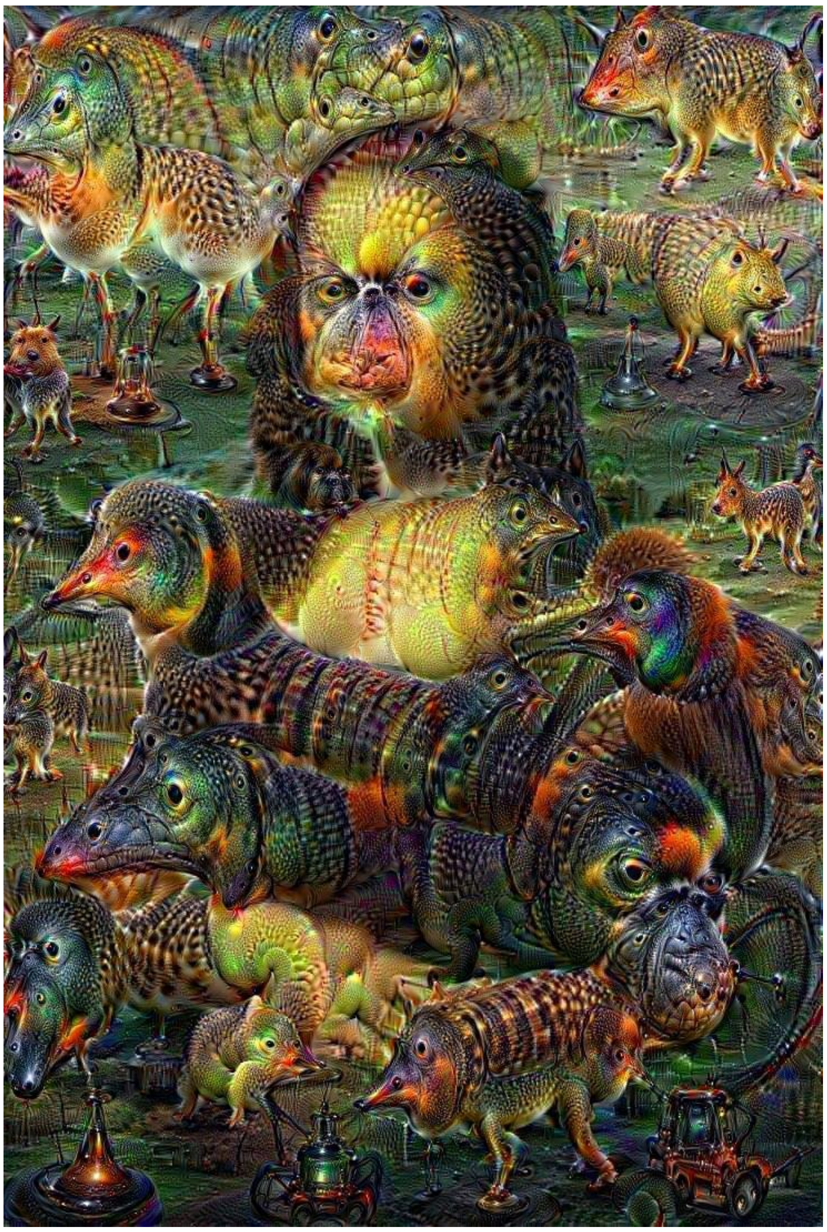
The Amazon also provide SDKs for Python for public to install, which allows Python developers to write software that makes use of Amazon services like S3 and EC2. S3 is a cloud storage service from Amazon. Then we downloaded their source code and installed the environment needed. It works pretty well, popping out various labels with confidence score. However, we finally abandoned it as we find out the output is a bit irrelevant to our goal. The outputting labels are actually some words, but our aim is to purely evolve the lines/patterns to another form of lines/patterns in the image instead of classifying the certain objects in the picture to certain nouns. We also thought that we change the idea a bit, just search the labels in Google engine, then we can get a different picture. It might be an easier approach, but it does not qualify our original concept of our project.

4. DeepDream from Google, Caffe

Finally, we also played with another computer vision program created by Google Research Blog. Its idea really fits our original concept, but the implementation is undoable so far. Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors. Their website http://caffe.berkeleyvision.org/installation.html provides the public with open-source code to DIY Deep Learning for vision with Caffe. However, the hardware of running Caffe is quite demanding, like Berkeley Vision runs Caffe with Titan Xs, K80s, GTX 980s, K40s, K20s, Titans, and GTX 770s including models at ImageNet/ILSVRC scale. It is because of the hardware limitation from us (Only two notebooks), we decided to hold on learning and installing the modules.

5. Online Deep Dream Generator, https://deepdreamgenerator.com/

To achieve our goal, we finally adopt an online platform where we can transfer photos using a powerful AI algorithms. Although their code is hidden because it just provides a user interface, at least it helps transforming our image as a stepping stone for our project progress.



(Generated by Deep Dream)

Steps to imagine from pictogram:

1. Use the Deep Dream Generator to recreate pictogram image collage
2. Dream deeper until level 3[1]
3. Import OpenCV module to change the image to the thresholding image in order to make the image accessible for CNC plotting machine
4. Transfer the image to path as GCode file by using Python.

[1] The system provides 6 levels to "hallucinate" more details from the Deep Dream, the deeper level, the higher complexity of the output image.

5. Send the Gcode to the Gcode sender to manipulate the Arduino drawing machine[2]
6. Draw the graphic which is imagined by computer.

Conclusion:

As we mentioned before, the purpose of our project is not listing out the existing AI products/source codes and praising how intelligent the machine learning, deep learning or OpenCV is. The reason why we keep spending time to find different kinds of computer technology is that we once believed that computer can replace and even surpass human being one day, if we just apply a technology which is smart enough. One year ago, we came up with the doubt that whether computer can think creatively by using its imagination. Then we continuously tried those advanced computer theories like Neural Network etc. We think that our final product for this course is not only just handing the outcome, but it should also consist of recording the process of our exploration.

Does computer have imagination?

Although we used different kinds of AIs and visual libraries, until now, we will say we are still unsatisfied with the outputs of what computer gave us. Computer did excellent job in terms of feature matching and object detection, but the premise is human being provides data for them. The outputs can be predicted, because so-call how computer think is just how computer stores a large amount of data, which is classified by human through Mathematic algorithm or is costumed by human for certain functions. If we want to endow our human instinct to the computer like the purpose of this project, it is nearly impossible as they are not individuals to really think, dream or imagine.

Some might say that those people can have creativity because they experience things and learn knowledge from various areas, then they can quickly associate with interdisciplinary knowledge when they need to design or create something. Human being also cannot come up with something they have never encountered before. The

[2] The project we have done for last semester

principle is the same as the Ai to store fragmental data, then stitch, arrange and compare the data through certain algorithms. Once the computer system has same amount of data, they could also think, create and imagine like human. This is an interesting idea we are looking forward to seeing what will happen next in the futural technology development.

Link for Video Documentation:

https://www.youtube.com/watch?v=7BFGpEtXgjs

GitHub Repository:

https://github.com/Kenterese/Digital-Imagination

Reference:

https://pythonprogramming.net/loading-images-python-opencv-tutorial/
https://gist.github.com/alexcasalboni/0f21a1889f09760f8981b643326730ff
http://docs.aws.amazon.com/rekognition/latest/dg/get-started-exercise-detect-labels.html

```
@article{jia2014caffe,
  Author = {Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev,
Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell,
Trevor},
  Journal = {arXiv preprint arXiv:1408.5093},
  Title = {Caffe: Convolutional Architecture for Fast Feature Embedding},
  Year = {2014}
}
```

https://deepdreamgenerator.com/about