CS5330: Project 3: Real-time Object 2-D Recognition

Created by Sida Zhang, last modified on Mar 07, 2022

@Sida Zhang and @Hongyu Wan

March 4, 2022

Introduction

This project allows user to do multiple matching operations via GUI(Graphical User Interface) that we implemented. The project is about 2-D object recognition. We have accomplished to have a computer to identify a set of objects that are placed on a white surface. Each object was taken from three different angle. The project recognizes objects that are placed under the camera and identify the object from the dataset by its label in real time (video sequence available).

Our operating system is Windows11 and the IDE is Visual Studio Code. We have also used cmake to create header files. For our GUI extension, we built the project with an outside source lib, CVui.

Please feel free to post any comment or suggestion on this page, and thank you for viewing our Wiki Page 👍.

Over all thinking

I think the biggest challenge we had for this project is that we couldn't find a way to draw the major axis on the connect component at first. The first couple tasks were pretty straight forward but it got a lot harder in the next tasks. I spent a tons time to figure out where went wrong on calculating the Euclidean distance metric because I made a huge mistake when reading image function was reading in non-numeric order but I didn't know about it, I thought that was pretty hilarious. 😃

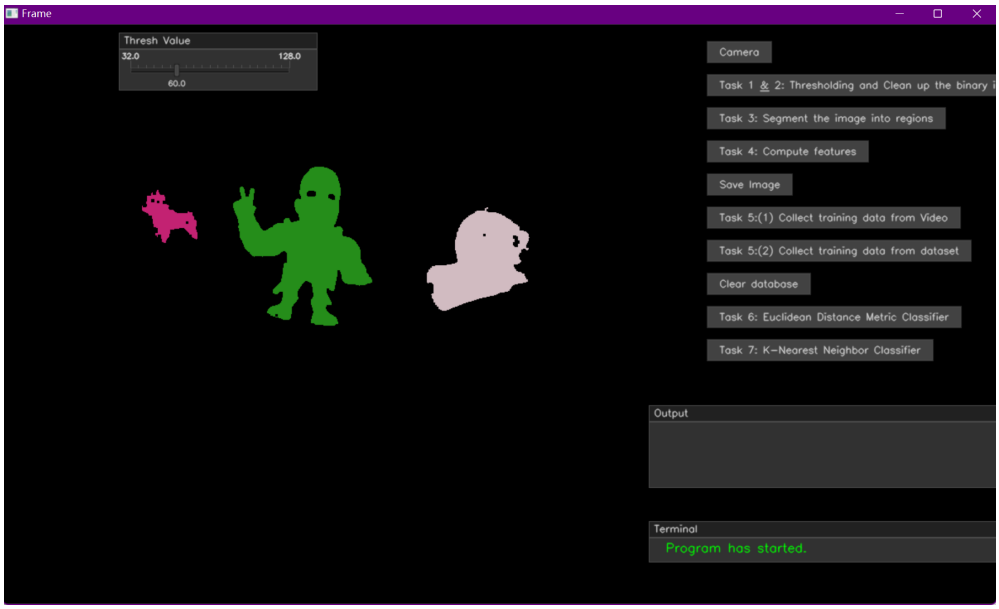
Another issue was I took pictures of many objects with my iPhone and somehow they are not showing correctly in OpenCV, and later on we figured out it was due to quality of the camera. All the images are around 4032x3024 pixels.

I am very proud that our OS gives pretty good accuracy detection and our multi-objects detection works great as well.

The project is definitely a lot harder than the previous one but I think I gained a lot more knowledge in CV by working on the tasks one by one.

I will now demonstrate each task and functionality of the project in below. Please scroll down to Task 9 or Extension 1 for a video demonstration of the project.

Graphical User Interface Instruction



Functionalities including: user friendly thresh value, original camera button, thresholding and cleanup button, find connect component region button, compute features (also draws bounding box, centroid, and axis around objects), save image to the dataset button, store feature from video stream button, store features from the dataset button, clear database button, calculate the Euclidean Distance Metric button, calculate the K-nearest neighbor button, a output window for matching results, and a terminal reader.

Tasks and functionalities:

We have first stored ten objects in our dataset and later on we added functionalities to add more objects to the CSV and the ability to save image in the dataset folder on GUI.

A little advise on objects picking, do not use banana as your object, they going bad much quicker than you know 😃

Task 1: Threshold the input video (code from scratch)

For the first task, we successfully separated objects from the background (white background) and turned the RGB image into a binary image. We first calculated the saturation value of each pixel by adapting below formular:

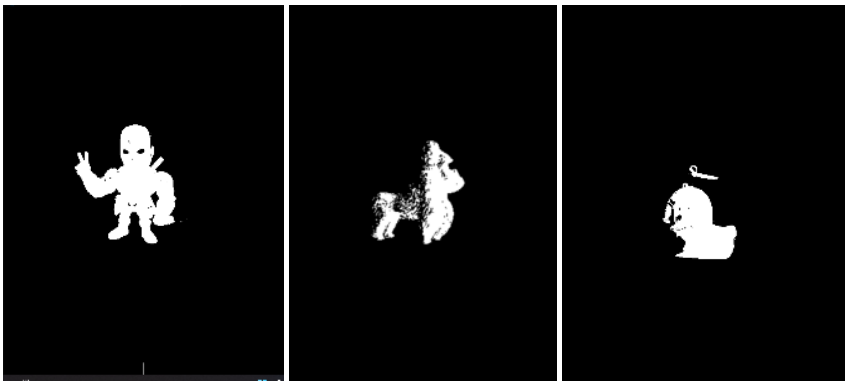
$$V \leftarrow max(R,G,B)$$

$$S \leftarrow \begin{cases} \frac{V - min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Then we find the pixels with high saturation value to be darker.

We finally get the pixels that are higher than the thresh value to be black and pixels that are lower than the thresh value to white.

All examples below show thresholed image.



Task 2: Clean up the binary image (code from scratch)

For this task, we implemented a function to clean up the binary image by adapting the morphological operations:

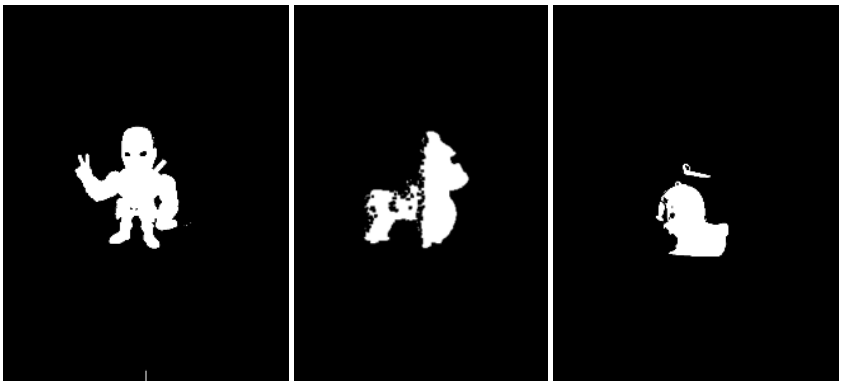
$$\texttt{dst}(x,y) = \min_{(x',y'):\, \texttt{element}(x',y') \neq 0} \texttt{src}(x+x', y+y')$$

$$\texttt{dst}(x,y) = \max_{(x',y'):\, \texttt{element}(x',y') \neq 0} \texttt{src}(x+x', y+y')$$

We first adapted growing operation by darken white pixels that are adjacent to a dark pixel, this operation helps to fill holes.

Then we adapted shrinking operation by whiten dark pixels that are adjacent to a white pixel, this operation helps to remove noises.

All examples below show cleaned binary image.



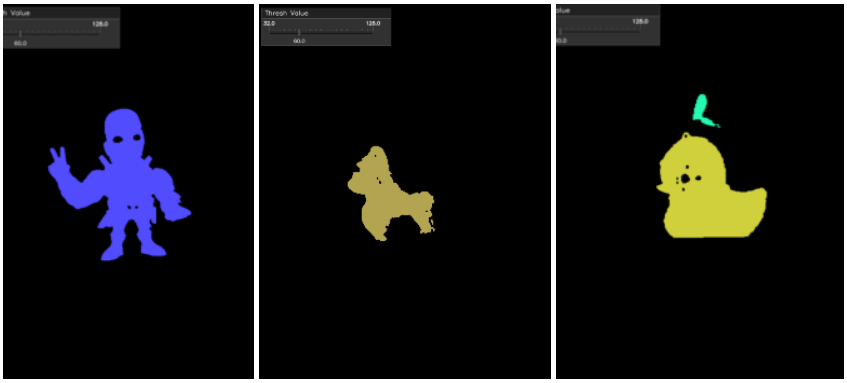
Task 3: Segment the image into regions

For this task, we used OpenCV built-in function to run connected components analysis on the cleaned regions.

Our system can recognize run connected components analysis simultaneously, up to three objects.

We first eliminated regions that are two small, then we take the biggest three objects as our final segmentations.

All examples below show cleaned objects with centroid, axis, and its bounding box.



Task 4: Compute features for each major region

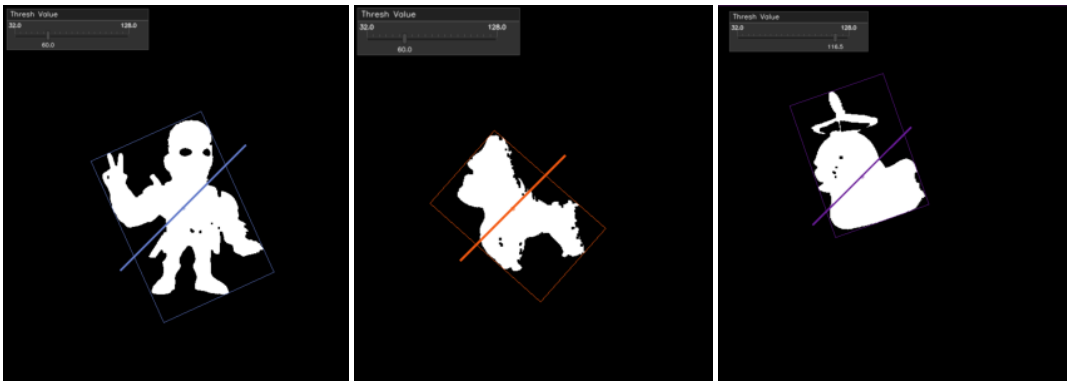
For this task, we use moments() to find spatial moments and central moments, and minAreaRect() to draw the bounding box.

Then we adapted the alpha formular to calculate the angle and drew axis on an object.

$$\text{alpha} = 1/2 \ \text{atan2}(2\ m\_11,\ m\_20 - m\_02)$$

Now we have the information to store features to the database: ratio and percentage filled in an object as our two features.

All examples below show cleaned objects with centroid, axis, and its bounding box.



Task 5: Collect training data

In this task, we ask users to input label names for each object and stored the features into a database, a CSV file.

```
Please label object 1 from the dataset.
Deadpool
Please label object 2 from the dataset.
Wallet
Please label object 3 from the dataset.
Mouse
Please label object 4 from the dataset.
BicLighter
Please label object 5 from the dataset.
Banana
Please label object 6 from the dataset.
Pen
Please label object 7 from the dataset.
CreditCard
Please label object 8 from the dataset.
RubberDuck
Please label object 9 from the dataset.
Orange
Please label object 10 from the dataset.
Bracelet
Please label object 11 from the dataset.
MrApe
Please label object 12 from the dataset.
AppleWatch
```

Below are the features that we stored into the database.

```
Deadpool,1.3904,0.5685,1.4065,0.5667,1.4011,0.5700
Wallet,1.3508,0.9656,1.3397,0.9709,1.3329,0.9728
Mouse,1.6173,0.8361,1.6123,0.8325,1.6193,0.8261
BicLighter,3.3385,0.8547,3.3466,0.9280,3.3119,0.8791
Banana,1.8225,0.4372, 1.8234, 0.4287, 1.7929, 0.4182
Pen,8.9680,0.8505, 8.8936,0.8652,10.5285,0.5808
CreditCard,1.5948,0.9730,1.5856,0.9578,1.5668,0.7755
RubberDuck,1.2291,0.6091,1.0786,0.6822,1.1078,0.6729
Orange,1.0219,0.7857,1.0219,0.7846, 1.0203, 0.7754
Bracelet,1.1946,0.6200,1.1974,0.5568,1.0928,0.6085
MrApe,1.4643,0.4653,1.2286,0.4843,1.4318,0.5024
AppleWatch,1.3094,0.2931,1.2838,0.7628,1.1153,0.3344
```

Task 6: Classify new images

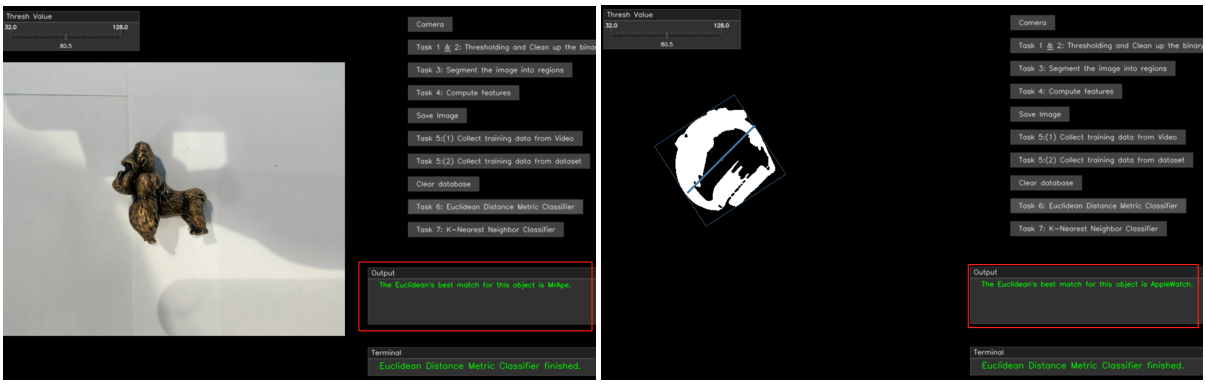
Now we have a dataset that stores all the pre-loaded features. We implemented a function to first calculate the ratio and percentage filled of objects under the camera and then compare it to the features in dataset by using Euclidean Distance Metric.

We have spent a lot of time to calculate the standard deviation

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

because somehow, we always get the incorrect values after square root the value. I believe it is because of data overflow but I'm not entirely sure. We finally managed to calculated the distance metric by putting the value into a vector after it is square rooted.

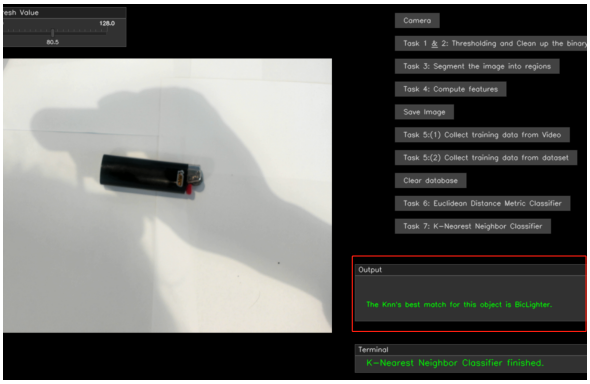
All examples below show that Euclidean Distance Metric correctly matched objects from the database.

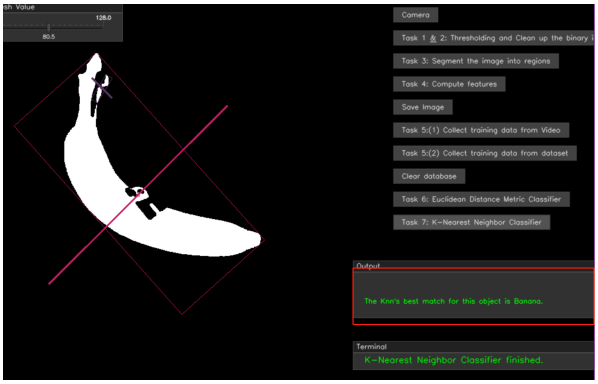


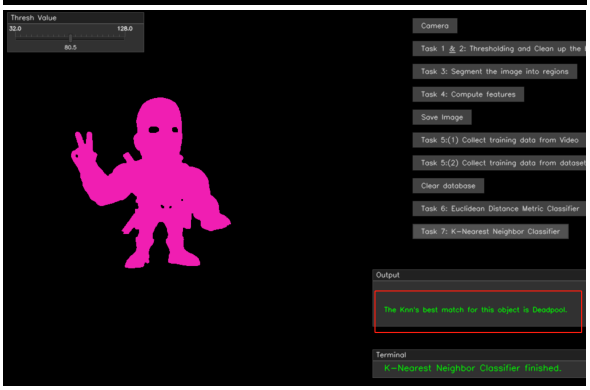
Task 7: Implement a different classifier

For this task, we implemented a K-Nearest Neighbor classifier system where parameter K is 3. This classifier takes first calculate the Euclidean Distance Metric from the training sets and sort the values from low to high. Then the classifier calculate the object with most frequency (up to 3 because k = 3) by adapting the Boyer–Moore majority vote algorithm and return the matching.

All examples below show that KNN correctly matched objects from the database.







Task 8: Evaluate the performance of your system

Below are the two-confusion matrix for the objects in my system. I have run 10 times Euclidean Distance Metric and 10 times K-Nearest Neighbor. I believe both of the results are pretty accurate.

Euclidean Distance Metric:

	Deadpool	Wallet	Mouse	BicLighter	Banana	Pen	CreditCard	RubberDuck	Orange	Bracelet	MrApe	AppleWatch
Deadpool	8										1	
Wallet		9	3	1			1					
Mouse			7				1					
BicLighter				9		2		1				
Banana					8			1				
Pen						8						
CreditCard		1					8					
RubberDuck								5		1	1	
Orange								1	10	1		
Bracelet					1					8		1
MrApe	2				1			2			7	
AppleWatch												9

EDM Accuracy: ~80%

For K-Nearest Neighbor classifier, we updated the frequency of each objects and picked the object with the most frequency (3 as the most frequency since K is 3) as the matching object.

K-Nearest Neighbor where K = 3:

	Deadpool	Wallet	Mouse	BicLighter	Banana	Pen	CreditCard	RubberDuck	Orange	Bracelet	MrApe	AppleWatch
Deadpool	9							3			1	
Wallet		9	1			1						
Mouse			9							1		
BicLighter				9								
Banana					10	1						

Pen				1		8						
CreditCard		1					9					
RubberDuck							1	7				
Orange									10			
Bracelet										8	1	
MrApe	1										8	
AppleWatch										1		10

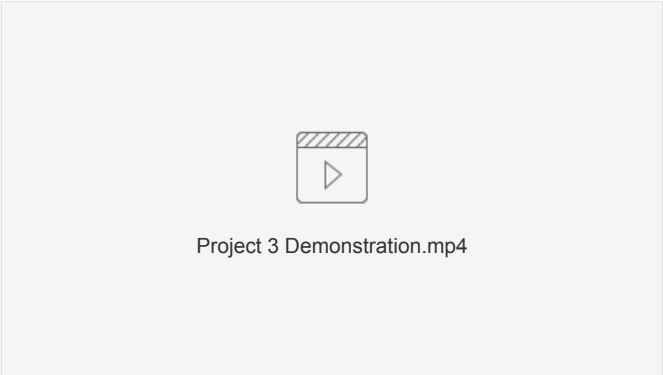
Knn Accuracy: ~88.33%

By comparsion, I noticed that the K-Nearest Neighbor tends to be a little bit more precisely.

The data is probably a little bit off because I took the pictures of the objects in a different environment (with better light sources).

Task 9: Capture a demo of your system working

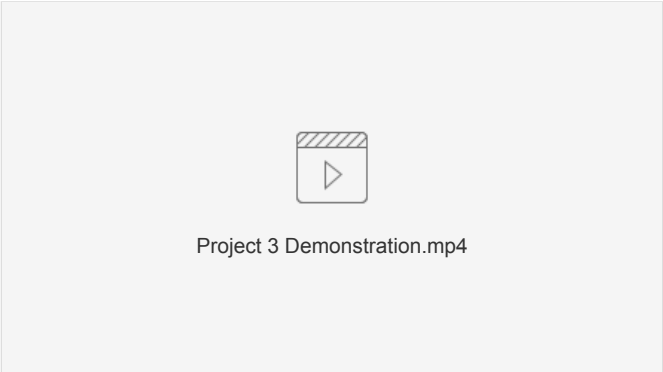
This demo also contains a GUI demonstration as well, please enjoy and feel free to post any comment relating to our project.



Note: Because I am holding the camera, the objects are flickering sometimes. It is perfectly fine when you have a static camera setup.

Extentions

Extension 1: GUI:



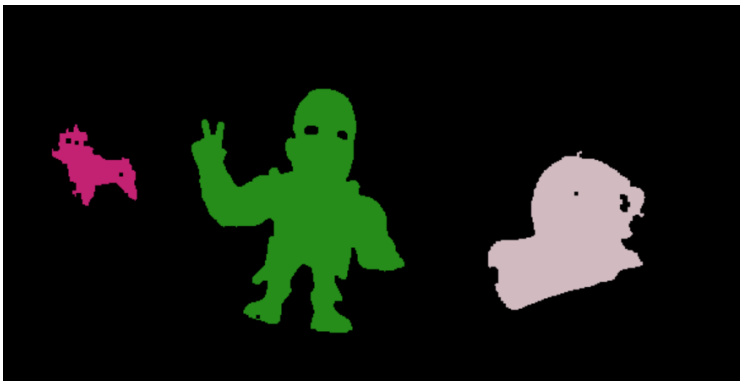
Extension 2: Allows costomized thresh value

Our GUI function allows customized thresh value by adjusting the thresh value box on the left side of the GUI system.

If an user is going to customized the thresh value, please remember to first clear the database and re-label all the pre-loaded image from the dataset to keep consistency for both of the database and the camera image.

Extension 2: Enable to recognize more than one object

Our GUI can recognize up to 3 objects simultaneously.

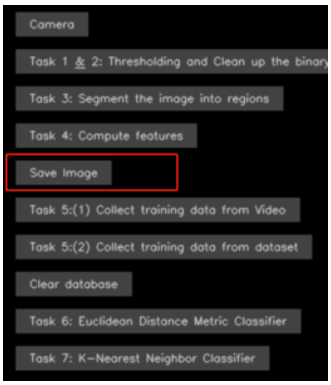


Extension 3: More than required ten object in the DB

For this project, we have 12 objects in the dataset.

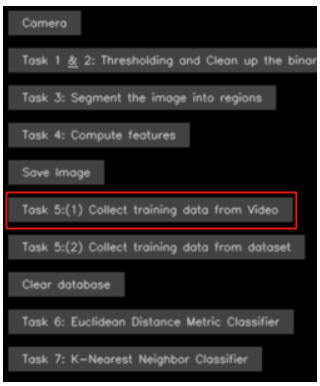
Extension 4: Add new objects into the image dataset folder

Our GUI allows user to capture images from the camera and save to the dataset folder.



Extension 5: Add new feature to the Database

Our GUI allows user to calculate new features from the camera and to save to the database, the CSV file.



Project Reflection:

Compare to the previous two projects, I think I have learned a lot more by doing the algorithm by myself. This project was definitely a lot harder than the previous ones but I think we have also enhanced our knowledges on the concept of the field. I believe the purpose of this project is very important because it helps us to get familiar with image recolonization by processing and manipulating images at a pixel level. I think it was also pretty cool to see our works to manage to match

objects under camera. I believe if we have a bigger dataset, we can do a lot better on matching as well; and I now learned the importance of big data because the more data you have the more accuary you get.

Overall, I thought this project was very challenging compares with previous two projects. But the results are quite pleasant I believe. I am very excited for our next project, with AR technology.

Acknowledgement:

Thanks to Dr. Maxwell and all the posts and discussions on Piazza.

Dr. Maxwell's CV lecture notes

Color Conversion: https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html

Morphological operators:

https://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm (a very comprehensive introduction on image morphology).

https://docs.opencv.org/4.x/d4/d86/group__imgproc__filter.html#gaeb1e0c1033e3f6b891a25d0511362aeb

https://docs.opencv.org/4.x/d4/d86/group__imgproc__filter.html#gaeb1e0c1033e3f6b891a25d0511362aeb

Connected Component:

https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga107a78bf7cd25dec05fb4dfc5c9e765f

Moments: https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga556a180f43cab22649c23ada36a8a139

CVui:

https://github.com/Dovyski/cvui

https://dovyski.github.io/cvui/components/sparkline/

K-nearest-neighbours: https://www.geeksforgeeks.org/k-nearest-neighbours/

No labels