Work Review
Computer Vision Course

Yoav Sabag
Yftach Gil

Work review Agenda

- Research Field
- Referenced Literature
- Project definition
- Project Methodology
- Ground Truth Dataset
- Project's Results

Research Field

Research subject:
“Most likely object detection in a scene with orientation difference”

Our goal:

Detect the most similar or the same object in complex scenes, with a change in the object orientation, based on LightGlue and YOLO.

Motivation:

- Object detection and matching is a popular task widely used in computer vision. It can be used in many application such as 3D reconstruction, Augmented reality, Mapping, Navigation, Autonomous vehicles and more.
- Latency-sensitive applications requires very fast computing time to process video or real-time data.
- In our research we used feature matching with LightGlue and object detection with YOLO models for detecting similar objects.
- Integration of the two models had significantly improved the results.

Metrics for the integrated model:

- True/Total prediction on the dataset we have created

Referenced Literature

Articles

LightGlue:

https://arxiv.org/pdf/2306.13643.pdf

Super Glue:

https://arxiv.org/pdf/1911.11763.pdf

Deep Image Homography Estimation:

https://arxiv.org/pdf/1606.03798.pdf

Object Detection with Deep Learning: A Review

https://ieeexplore.ieee.org/abstract/document/8627998

Matching Non-Identical Objects

https://arxiv.org/abs/2403.08227

YOLO

https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html

LightGlue Model

“LightGlue: Local Feature Matching at Light Speed” (ICCV 2023) is a research done by researchers from the Microsoft Mixed Reality & AI Lab at ETH Zurich.

LightGlue is a deep neural network that match local features across images. It has a simple and effective improvements of “SuperGlue”, a state-of-the-art model in sparse matching. LightGlue is more accurate and easier to train with less memory and computation time.

One key property of LightGlue is that it adapts to the difficulty of the problem. This results inference to be much faster on image pairs that are intuitively easy to match.

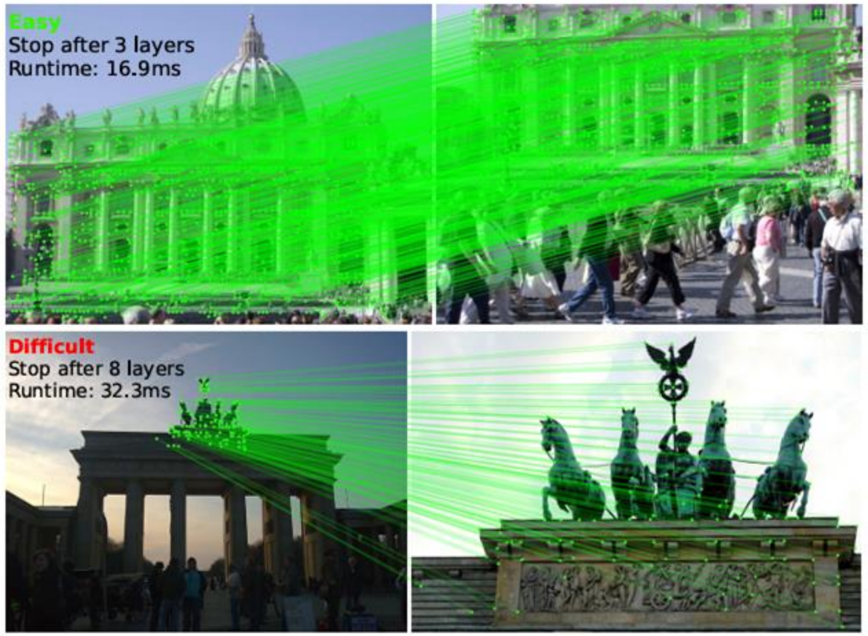
Since this model is efficient in terms of both memory and computation, it is fast and precise and fits latency-sensitive applications.

Main datasets that used for training:

- Homography estimation - with HPatch dataset
- Relative pose estimation - with MegaDepth dataset
- Visual localization - with Aachen Day-Night dataset

Metrics

- Image Pairs Per Second
- Relative Pose Accuracy [%]

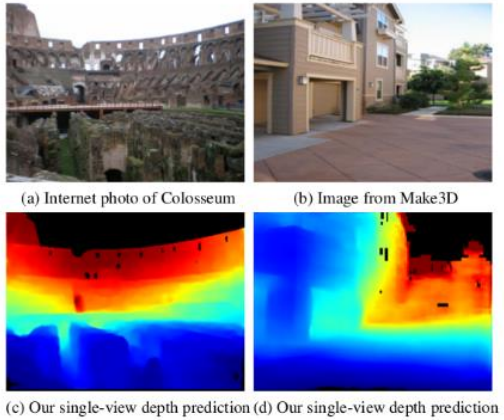


LightGlue Model - Datasets

Homography Validation:
HPatches

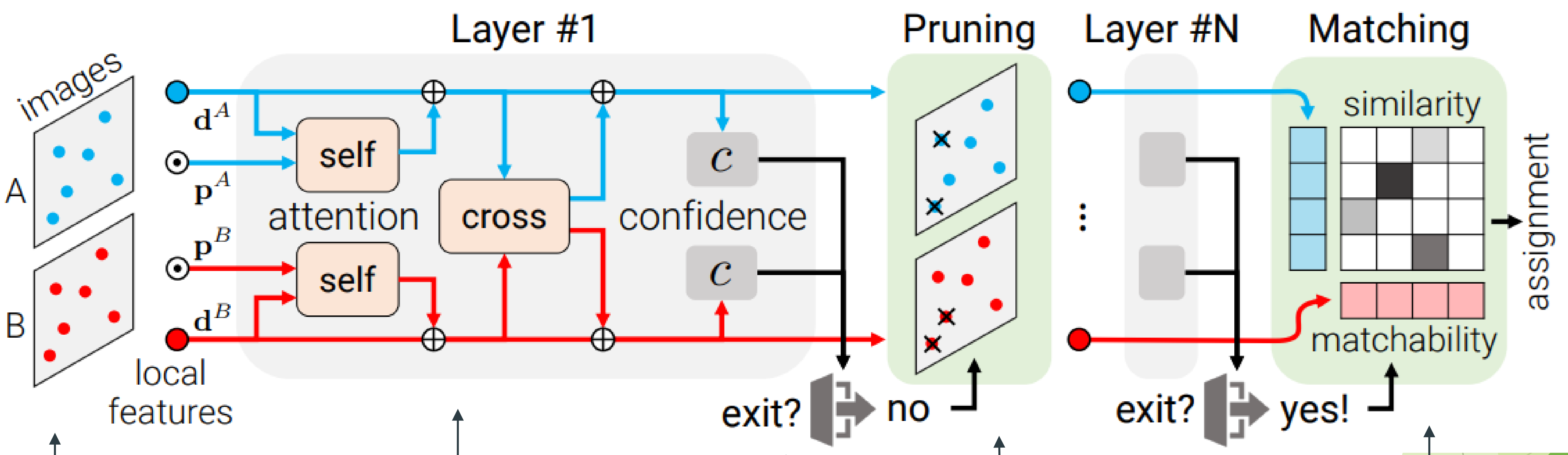
Image 1: Example image sequence. The leftmost image is the reference image, followed by 5 images with a different viewpoint.

Fine tuning:
MegaDepth: 1M crowd-sourced images depicting 196 tourism landmarks



Visual localization:
Aachen Day-Night dataset

Model Architecture



Pair images key points detection

Self and cross attention units with positional encoding

Matchable point confidence classifier determines the exit point

Prune unmatchable points

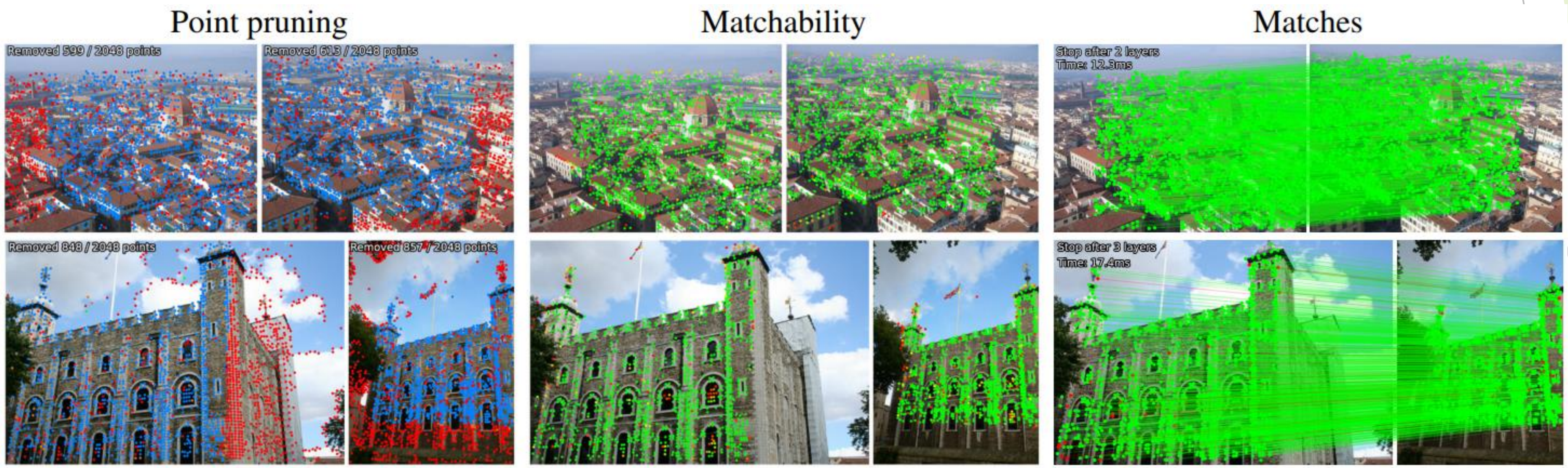
Once a confident state is reached, superGlue predicts an assignment between points based on their pairwise similarity and unary matchable

(SupePoint)

(MLP net)

(SuperGlue)

Model Architecture



key points detection

(Super Point)

Prune unmatchable points

Matchable point confidence classifier determines the exit point

Matching

(SuperGlue)

Confidence Classifier

For each point confidence classifier is computed

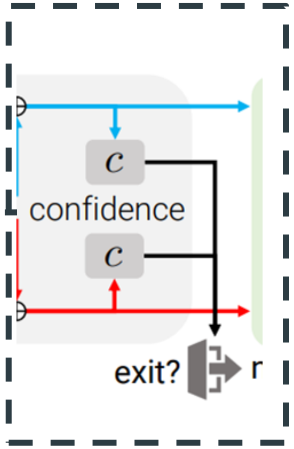
$$c_i = \mathrm{Sigmoid}\left(\mathrm{MLP}(\mathbf{x}_i)\right) \in [0, 1]$$

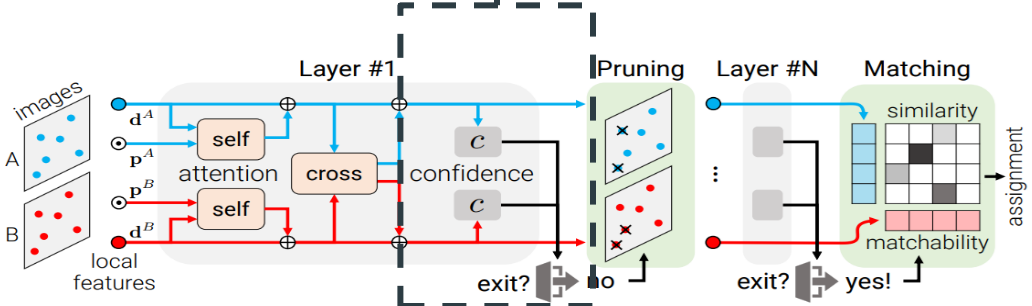
i - index

x(i) - point at index i

MLP - trained net to determine match or unmatchable point x(i)

C- the confidence interval





Exit Criterion

For a given layer ℓ, a point is deemed confident if c(i) > λ(ℓ)

We halt the inference if a sufficient ratio α of all points is confide

λ(ℓ) is decay throughout the layers until we get sufficient ratio of a

$$\text{exit} = \left(\frac{1}{N+M} \sum_{I \in \{A,B\}} \sum_{i \in \mathcal{I}} [\![c_i^I > \lambda_\ell]\!] \right) > \alpha$$

N- local features of image B

M- local features of image A

I -the association for each local feature i image

A - image A

B - image B

C(i)- confidence classifier

λ(ℓ)- For a given layer ℓ, a point is deemed confident if c(i) >λ(ℓ)

Point Pruning

A point is deemed unmatchable when its predicted confidence is high and its matchability is lower than threshold

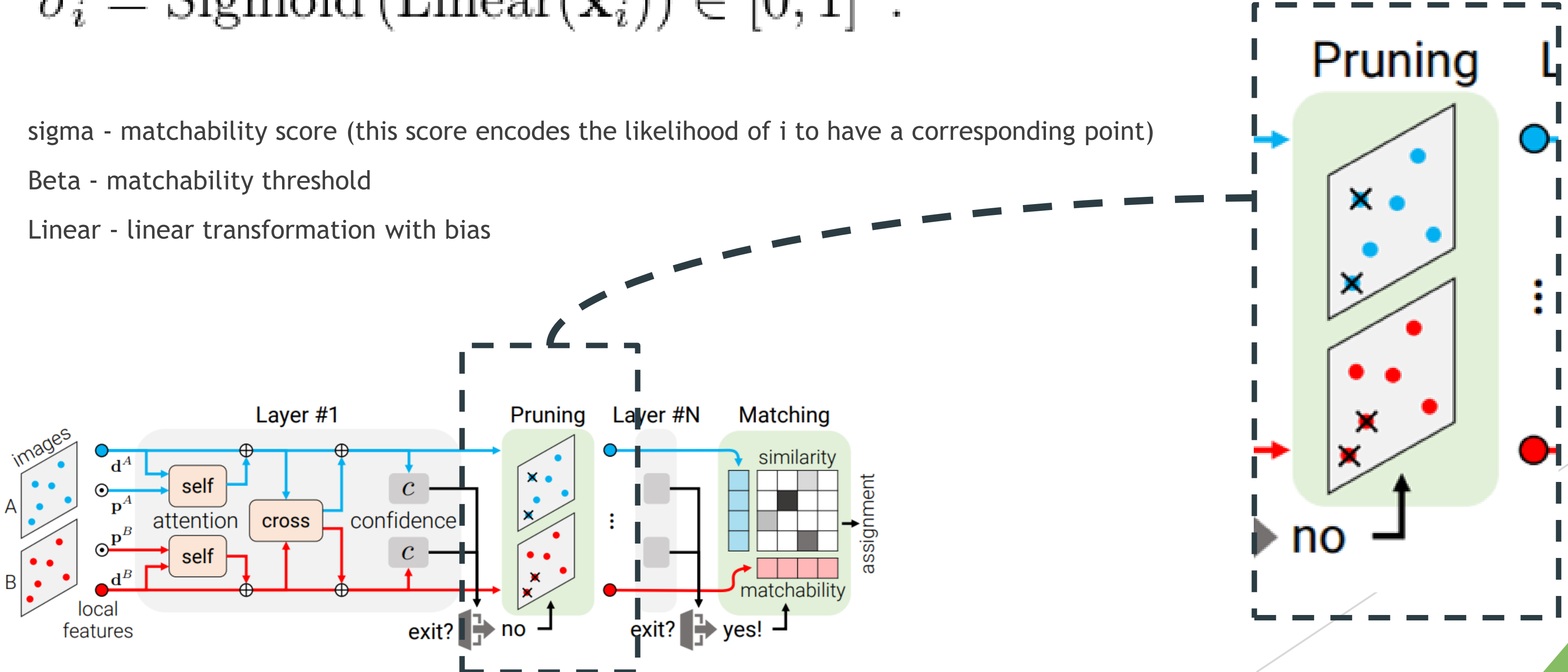
$$\text{unmatchable}(i) = c_i^l > \lambda_\ell \ \& \ \sigma_i^\ell < \beta$$

$$\sigma_i = \text{Sigmoid}\left(\text{Linear}(\mathbf{x}_i)\right) \in [0, 1] \quad .$$

sigma - matchability score (this score encodes the likelihood of i to have a corresponding point)

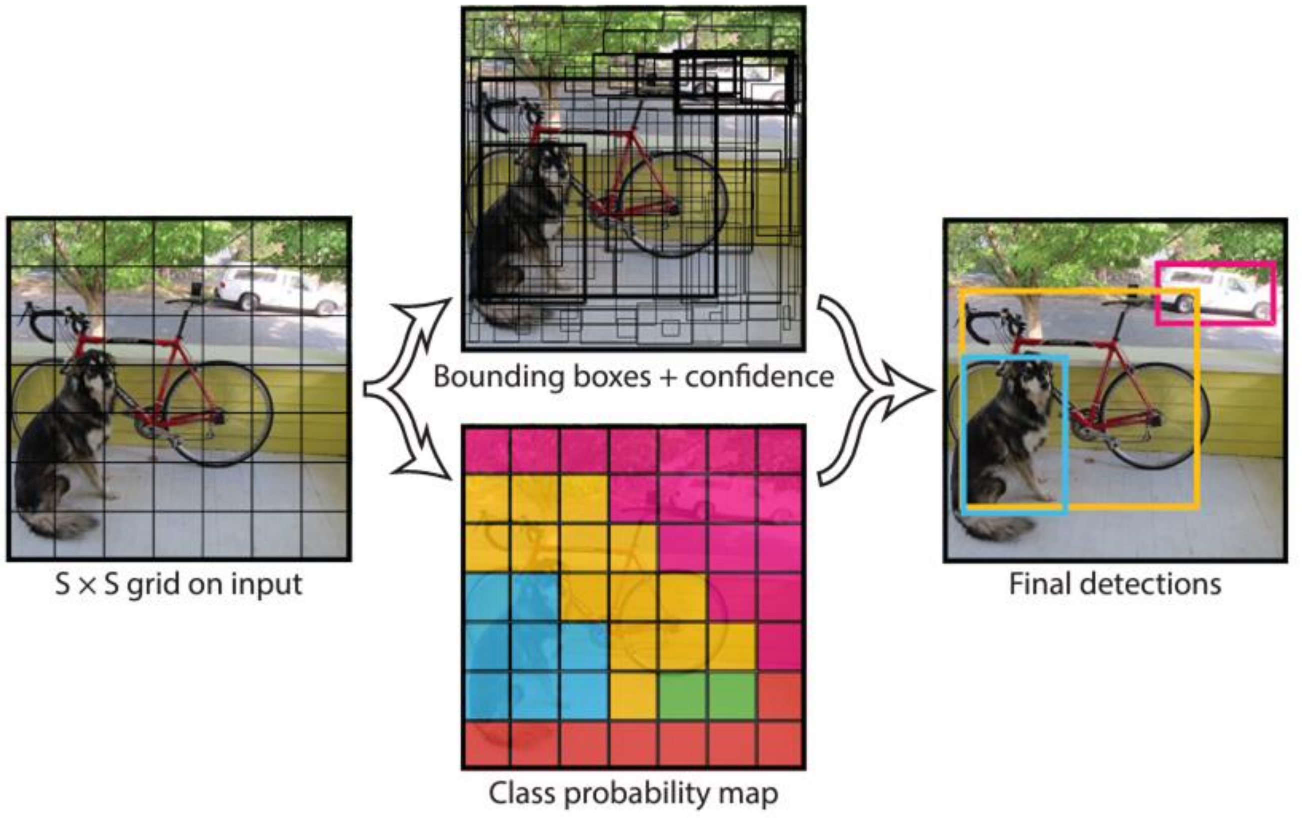
Beta - matchability threshold

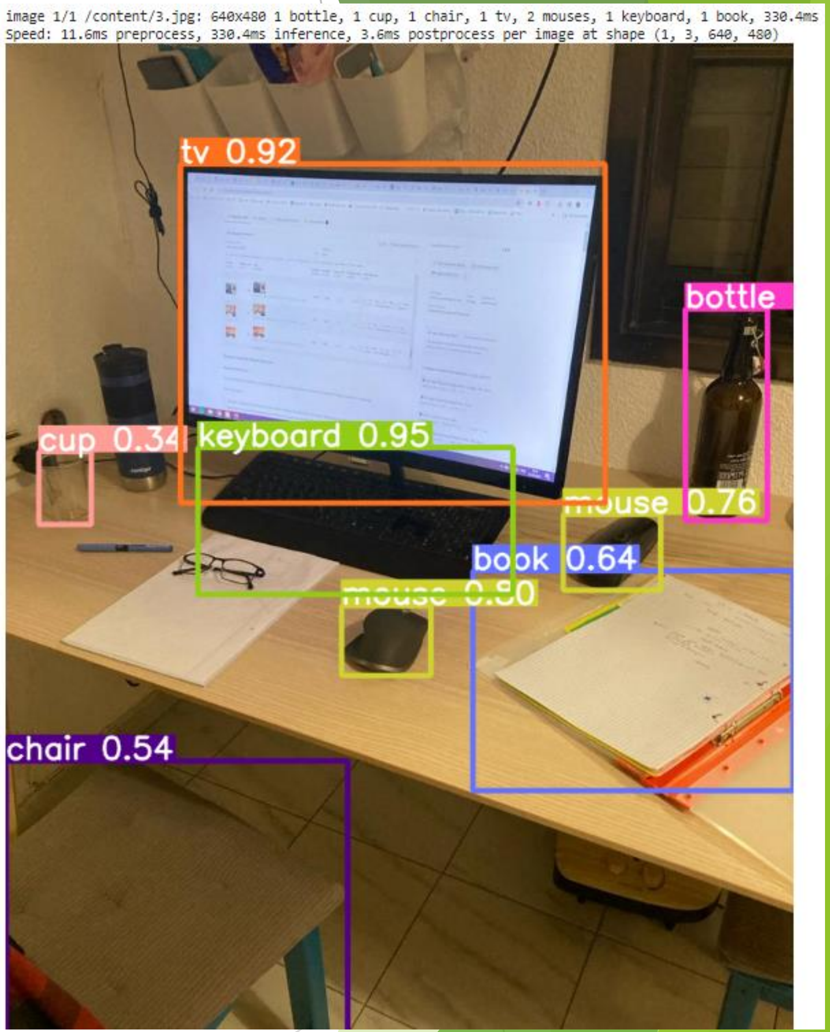
Linear - linear transformation with bias



YOLO - Object detection

- YOLO is a CNN based object detection model
- Input: Image
- Output: Bounding box, label (class), probability of the detection





First Experiment

Experiment - The problem

- LightGlue isn't very good to handle feature matching of objects in near and medium distance.
- Since LightGlue is originally trained on images of streets, views or buildings it does not work very good on smaller objects.
- There are very little matching points although the car is the same and in a rather similar posture.

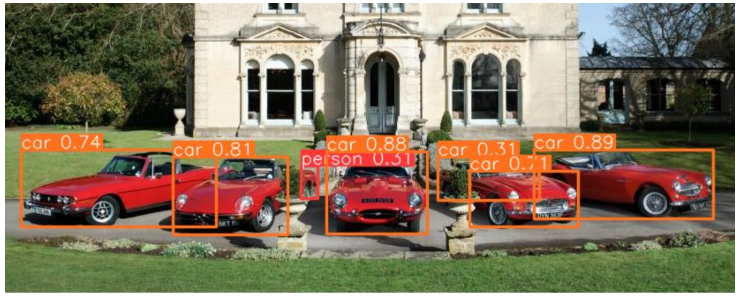
scene

object we want to search

Experiment - Suggested solution

- Use YOLO-V8 to crop the object from the complex scene.
- We assume that taking the region of interest (ROI) of each object in the scene raises the chances of LightGlue to match more keypoints and therefore to yield more confidence in the matching.

- Detect and label object in image_A and image_B
- For each object, crop the object to a new image
- Apply lightglue to check how good is the match




```
image 1/1 /content/car2.png: 448x640 1 truck, 532.8ms
Speed: 11.3ms preprocess, 532.8ms inference, 8.3ms postprocess per image at shape (1, 3, 448, 640)
```

Project Scope

Research Problem Definition:

Given 2 images {A,B} , A contains a single object and B is a multiple objects scene. Find the most likely object from image A in image B

Work intuitions

- We say the integration is “improving matching” if there are significantly more “Matched keypoints” after cropping the ROI.
- Object detection only takes prediction with a threshold of 50% probability.
- We evaluate performance of the integration of models (LG+YOLO).
- We create a ground truth dataset for evaluation of the integrated models.

Project Methodology

Methodology

- **Dataset Generation** - for performance evaluation of the integrated models we create a dataset that is considered the Ground Truth.
  - For each image in 1000 COCO images
    - detect all objects in the image using YOLO (scene B)
    - Take the first object with the highest probability (obj A) and Crop it. This object is considered as the Ground Truth.
    - Manipulate image A with homography to create image A'.
- **Model**
  - Input: 2 images, A' obj and B multi-obj scene
  - Detect objects ROI with YOLO
  - Perform lightGlue feature matcher for image A' vs each object ROI in image B and count the matches
  - Output: the object with the most matched features.
- **Model evaluation**
  - For each image in the dataset detect the object from A' in B and compare to Ground Truth
  - Calculate the score= matches/keypoint_A'_obj
  - Precision= TP/ (TP +FP)

Workflow

Create a dataset contains A and B

B-multi obj scene

A-obj[0] from B

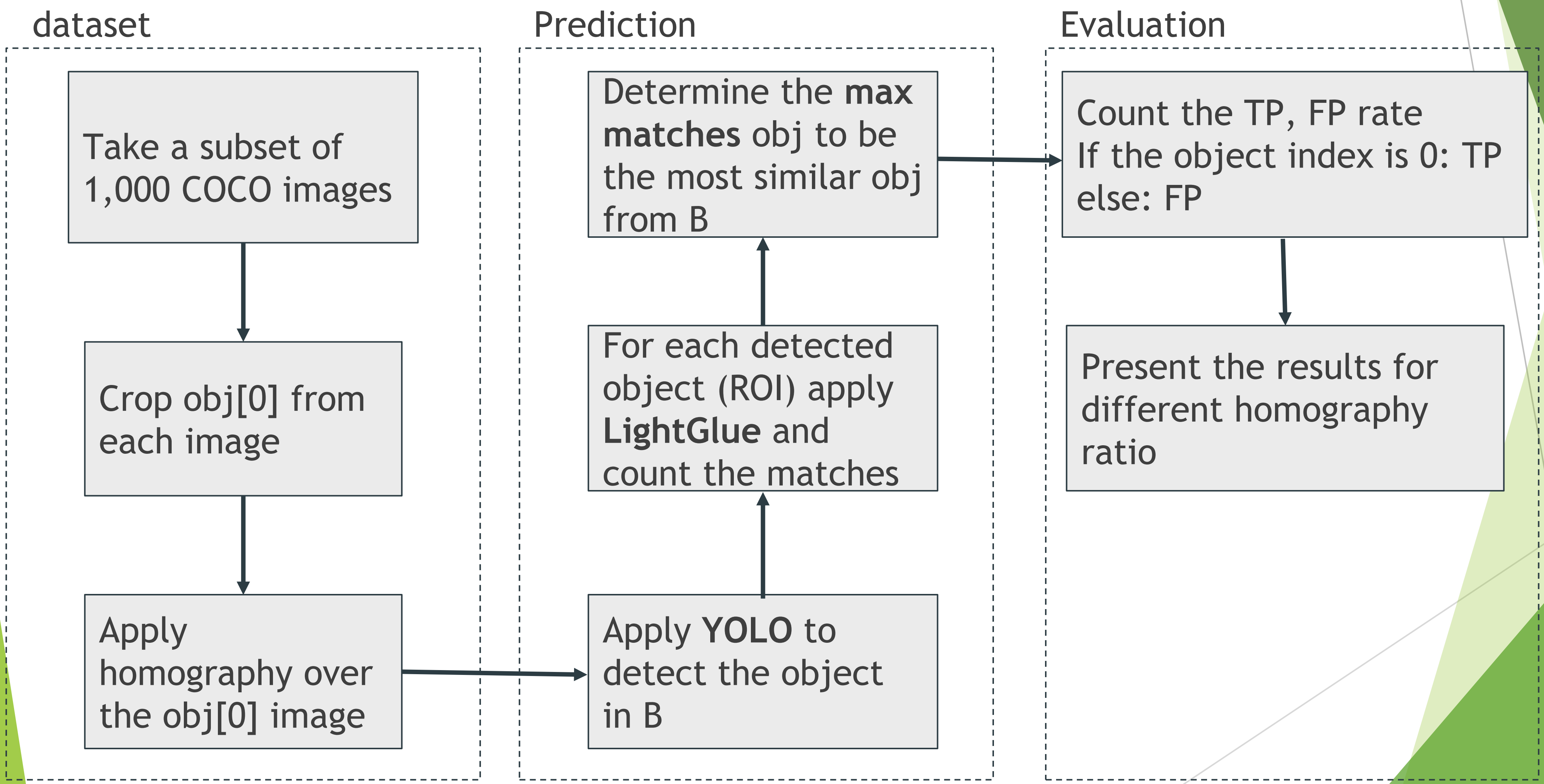
→ Define integrated YOLO + LightGlue models

→ Apply YOLO to find ROI, then apply lightGlue on ROI

→ Count the matches to determine the most similar obj in B

→ Check TP/FP+TP , TP/TP+FN and determine precision

Algorithm Architecture



Dataset

COCO Base

- MS COCO - https://paperswithcode.com/dataset/coco
  - Original COCO contains more than 200,000 images and 80 object categories
  - Each image has single object up to dozens.

- Projects Dataset - Based on COCO:
  Contains 1000 pairs of images:
  B - scenes
  A'-Homography

Create ground truth dataset

1. Take 1,000 COCO images - we'll name them image B
2. Detect (and remember) all objects with YOLO in the image B with probability>0.5
3. Choose one object (First object in the list) - remember class, ROI and prob
4. Crop and create a list of images - image A
5. Apply Homography to all A and create the "Homography image list" of A'
6. now we have a ground truth dataset containing images B and A'

Homography

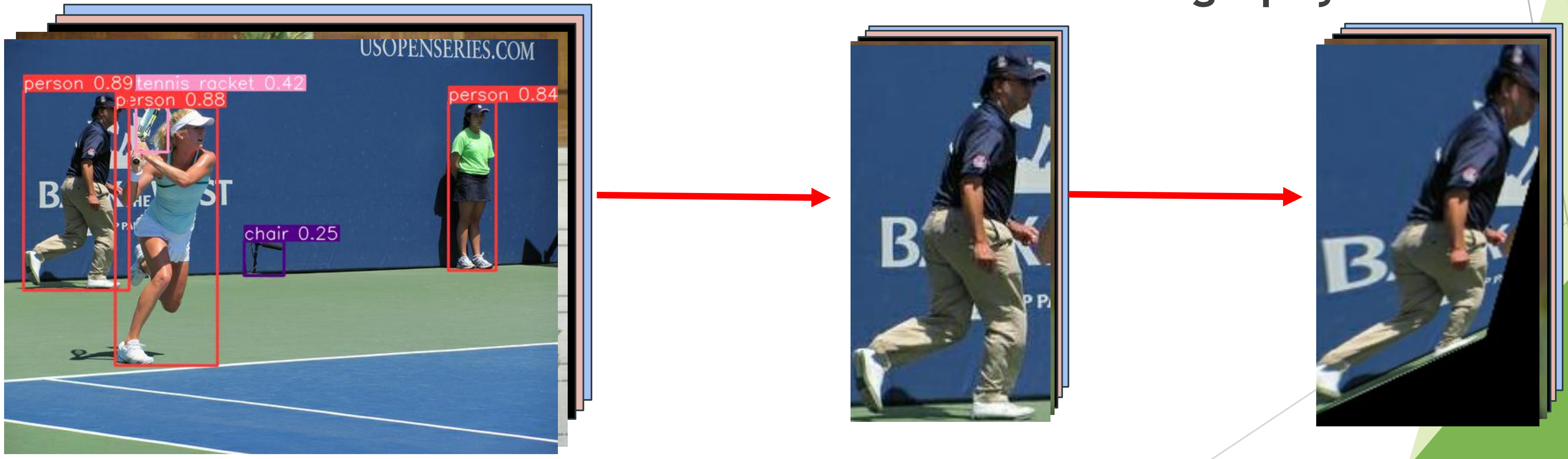
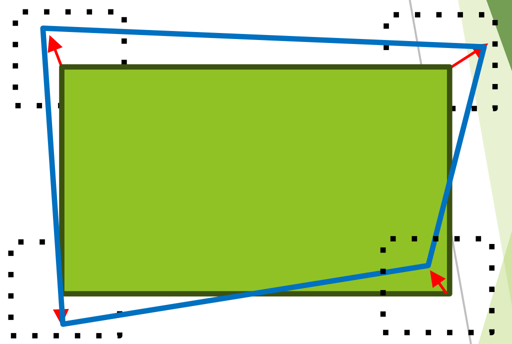
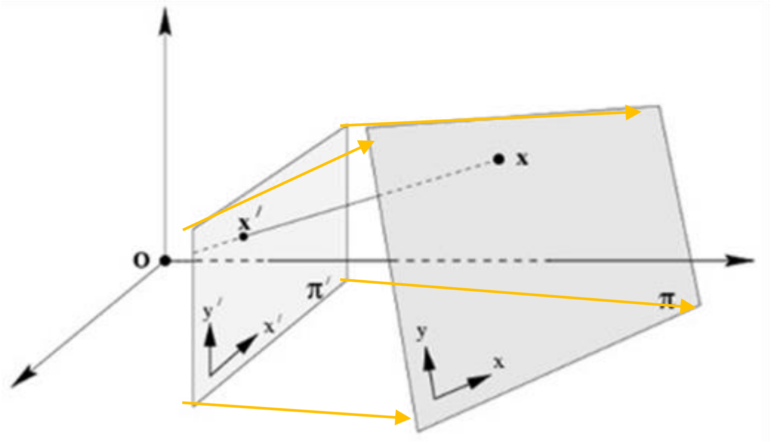


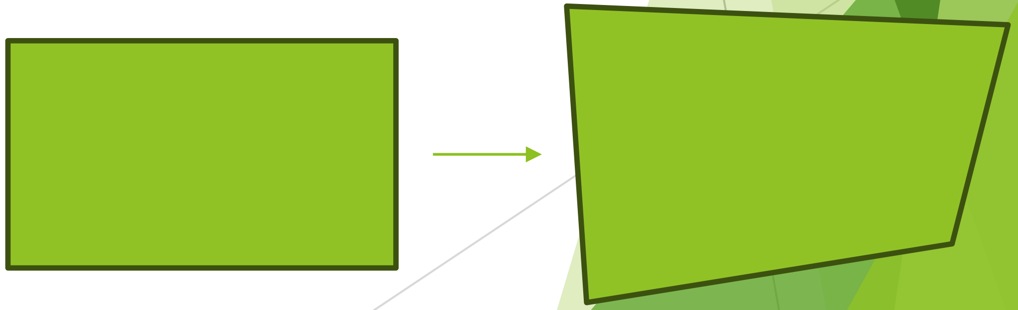
Image B **Image A** **Image A'**

Homography Transformation for creating Ground Truth "Random limited” projection

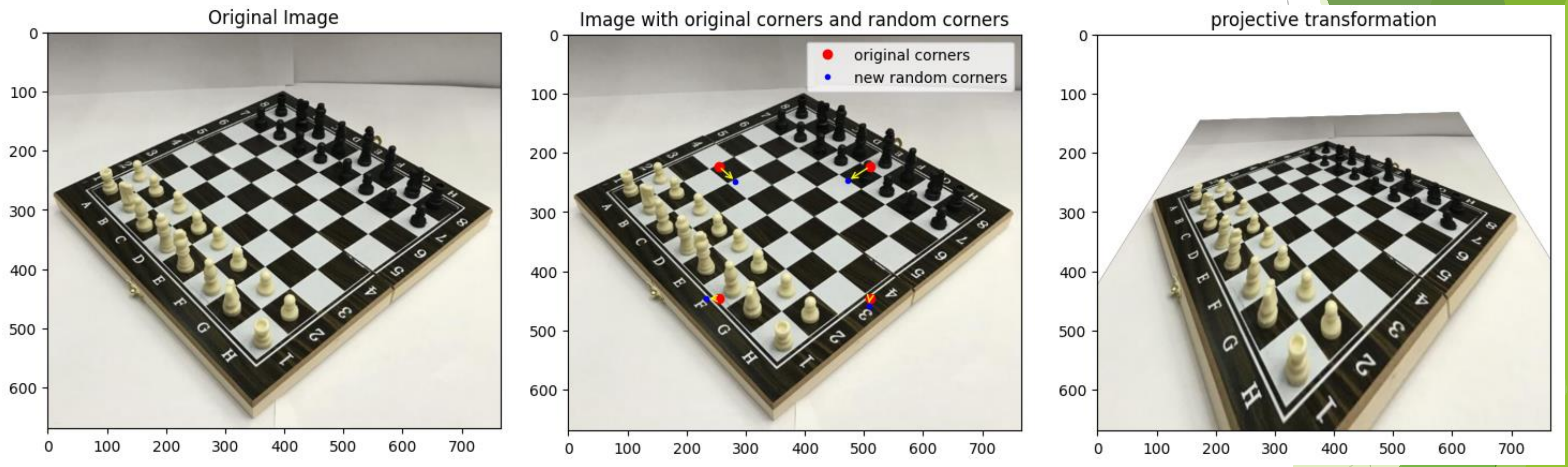
1. Take 4 central points of ROI as source.
2. Choose percent from width/height as upper limit.
3. For each corner add +/- random number to x and y.
   a. limit_X=percent*W
   b. limit_Y=percent*H
   c. new(x,y)=corner(x,y) + random(limit_X,limit_Y))
4. Transform image using “skimage transform”.
5. The results is a random homography: translation + rotation + affine+projection



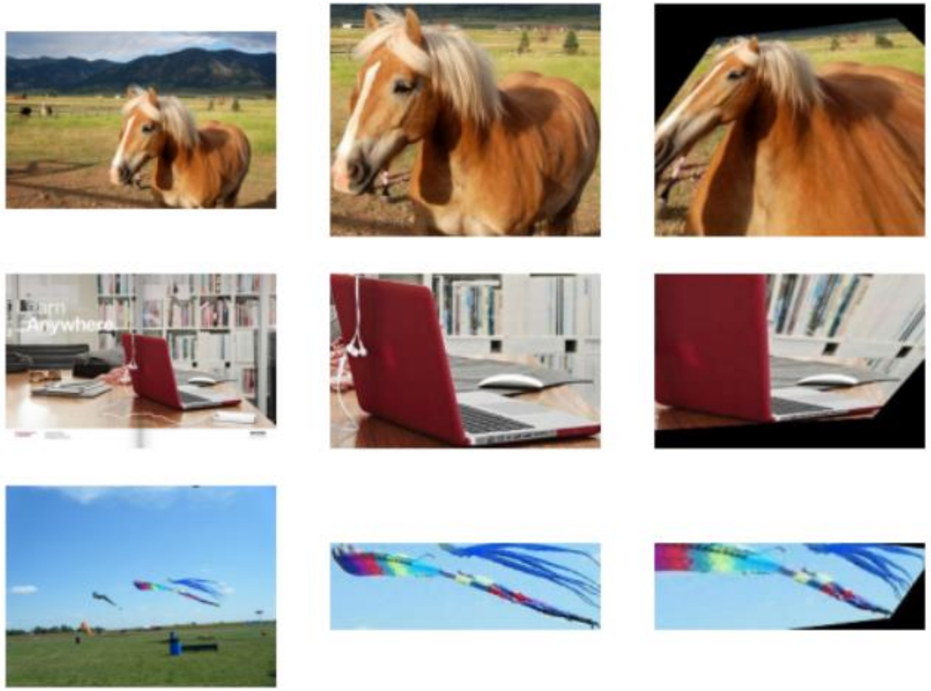




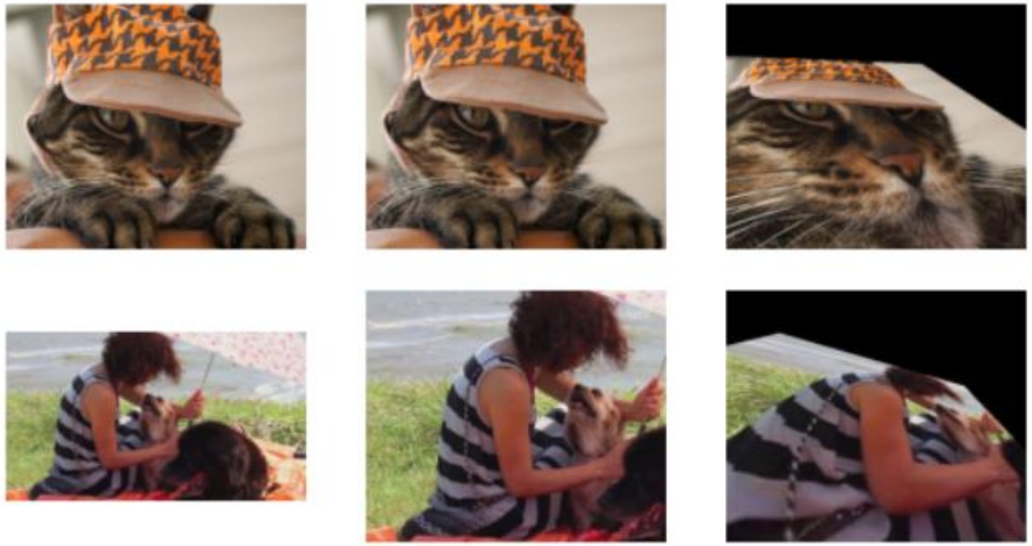
Example 5%


Original Image
Image with original corners and random corners
original corners
new random corners
projective transformation
0
100
200
300
400
500
600
700

Example 8% (after crop)



Example 12% (after crop)



Example 14% (after crop)

Example 17%
(after crop)

Upper limit

Off topic - Working with directories

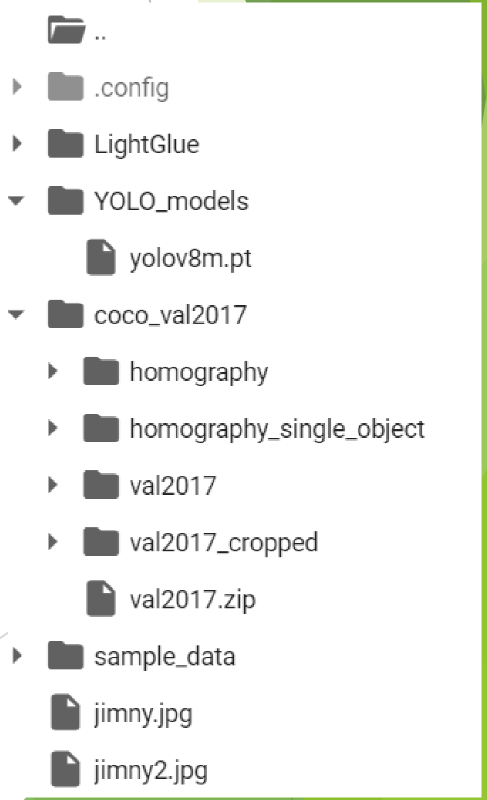
With terminal commands and “os” library you can:

- Change or get working directory
- Create folders and files
- List of files in a folder
- Save images after they have been cropped or transform with specific suffix
- Open images

```python
# list cropped image names
cropped_images_to_transform = os.listdir("/content/coco_val2017/val2017_cropped")
```

```python
len(cropped_images_to_transform)
```

```
1000
```



Off topic - Working with directories

```
percent=0.14
i=1
suffix = "_homography_single_object"
bar = progressbar.ProgressBar(maxval=len(cropped_images_to_transform)).start()

for filename in cropped_images_to_transform:
  # 1) imread file
  os.chdir("/content/coco_val2017/val2017_cropped")
  try:
    image=imread(filename)
    # 2) transform
    image_transformed = transform_image(image, percent)
    # 3) save with new file in homography folder
    new_filename = filename.split(".")[0] + suffix + ".jpg"
    os.chdir("/content/coco_val2017/homography_single_object") # work in homography img folder
    # cv2.imwrite(new_filename, image_transformed)
    pyplot.imsave(new_filename, image_transformed)
  except:
    continue
  bar.update(i)
  i=i+1

bar.finish()
print("\nFinished transforming images A to create A' in folder:\n", getcwd())
```

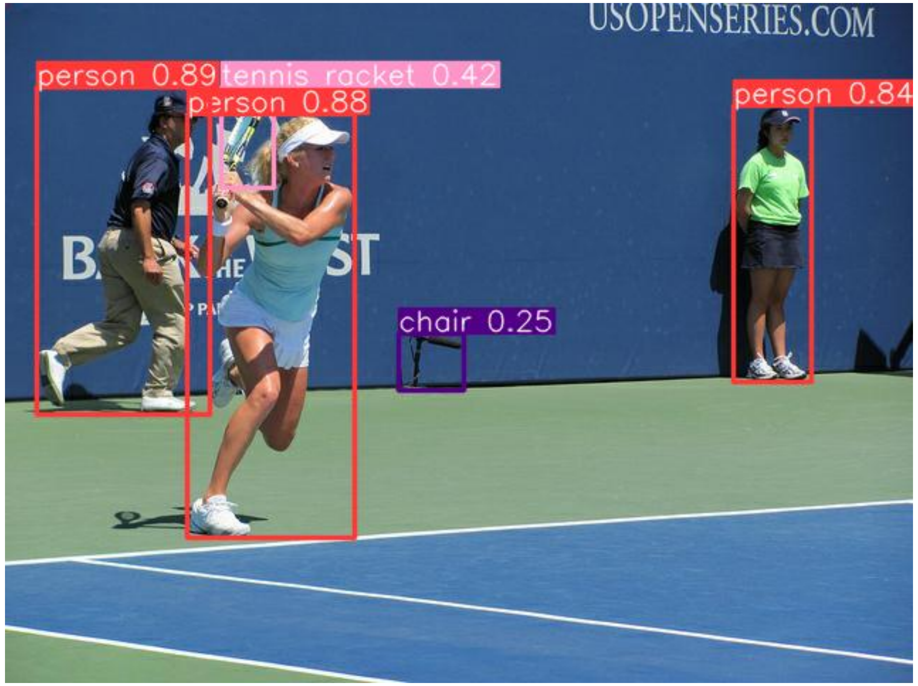
```
100% (1000 of 1000) |####################| Elapsed Time: 0:00:11 Time:  0:00:11

Finished transforming images A to create A' in folder:
 /content/coco_val2017/homography_single_object
```

Prediction

Create list of object in image B

The list includes data such as: bbox, class, class prob



Ground Truth Dataset

Take the first obj and manipulate it with homography

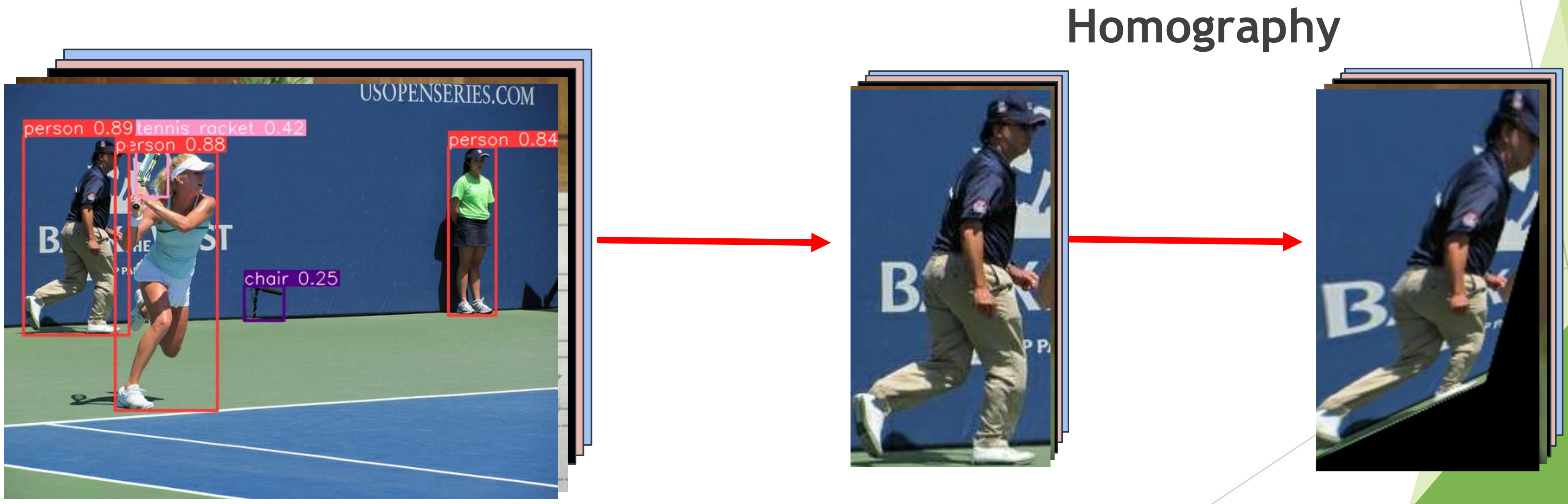


Image B

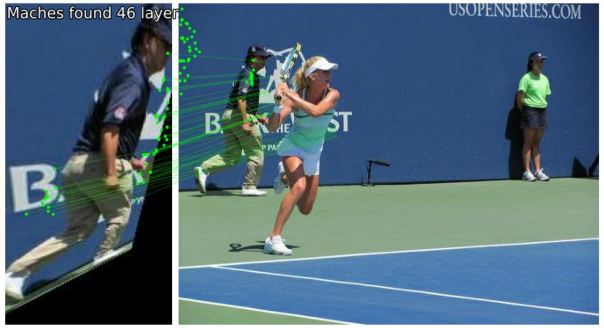
Image A

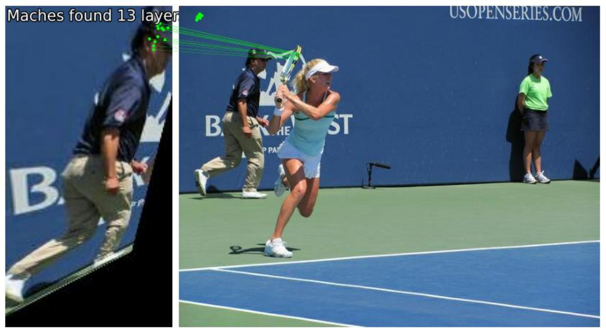
Image A’

Apply Lightglue for A’ vs ROI of B objs

for each object ROI in B apply lightglue with A’

1. detect keypoints
2. matching
3. count the matches





Evaluation

Score

- For each ROI in each Image we define a score:

  obj_score = matches/keypoints(image_A’_obj)

- than we create a list for all images:
  list_score
- For each image we check:

  If the first object has the maximum obj_score → True

  else: False

For example:

obj_score =[**obj1=0.44**, obj2=0.30, obj3=0.09, obj4=0.07] → 0.44 is max → **True Positive**

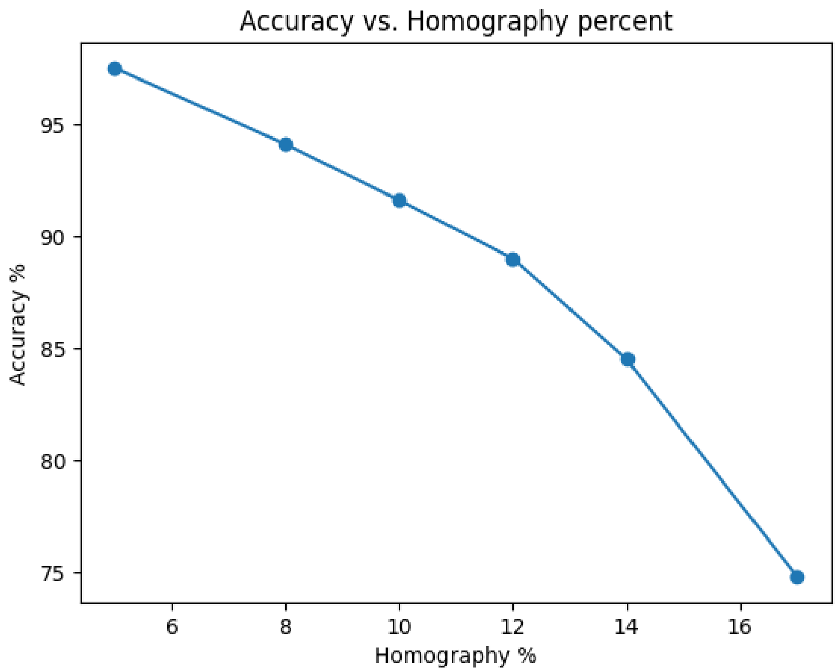
Define Precision

Precision= TP/ (TP +FP)

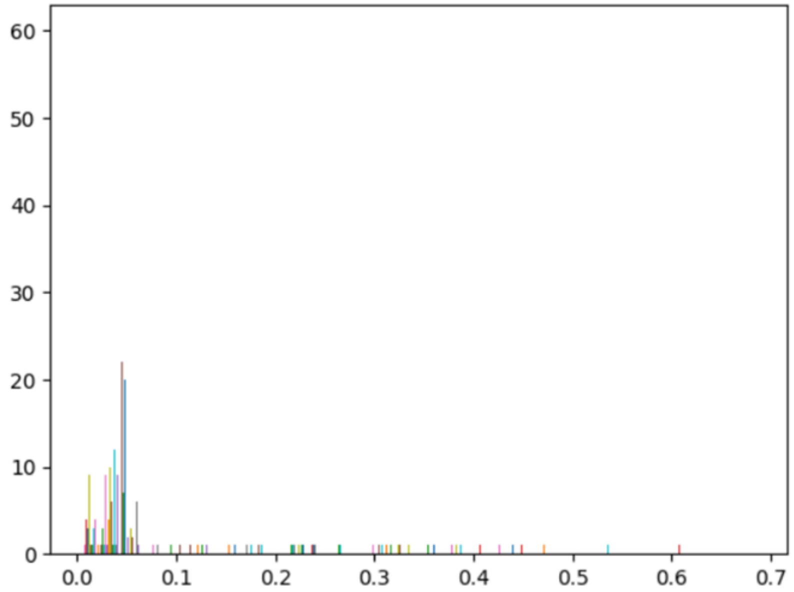
Results

Precision

- In this project we define the Precision as TP/TP+FP
- The ground truth obj index is 0 so TP determined accordingly
- We measure the results for 1,000 images



The graph present Precision vs homography rate



Example of Histogram of the score_list for 8% Homography. Most ROI's gets super low matches/keypoint ratio.

Further work suggestions

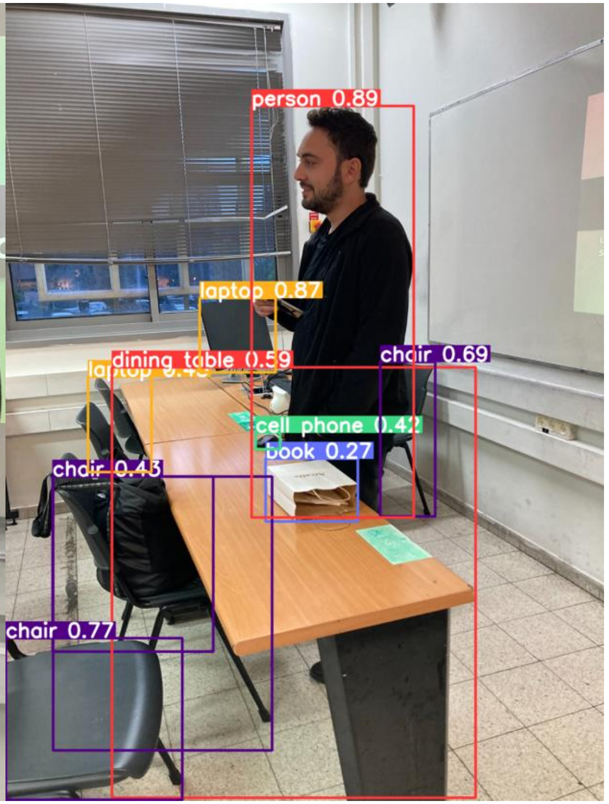
- Use the labels for saving time - for example first try to match first by label.
- Raise the probability limit.
- Input image with more than one object to detect.

Find Lior in compex scene

Object detection Detection

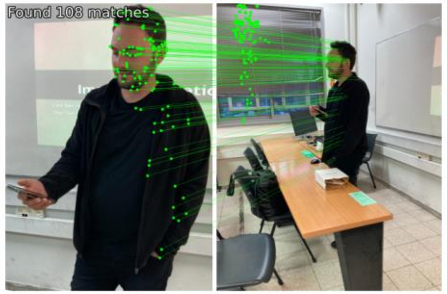
Obj to detect

Complex scene



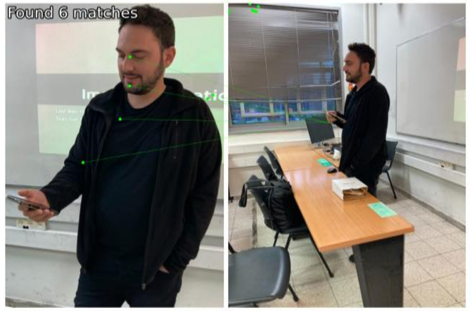
Feature Matching

Lior with Lior



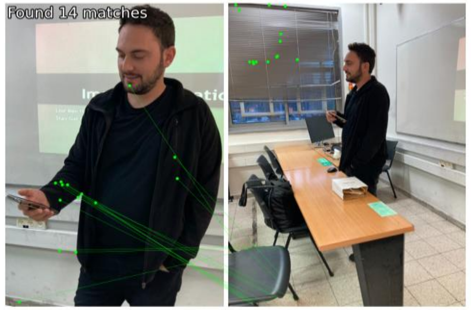
108 matches

Lior with PC



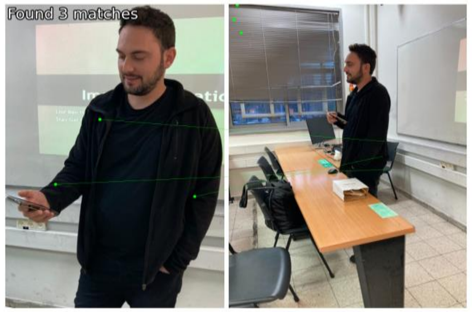
6 matches

Lior with Chair(1)



14 matches

Lior with Chair(2)



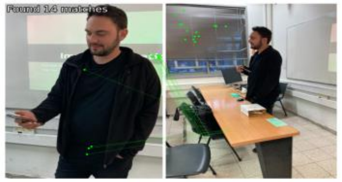
3 matches

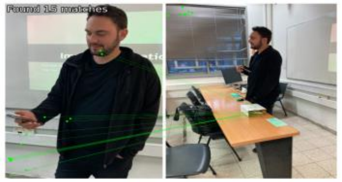
All other achieved less than 20 matches









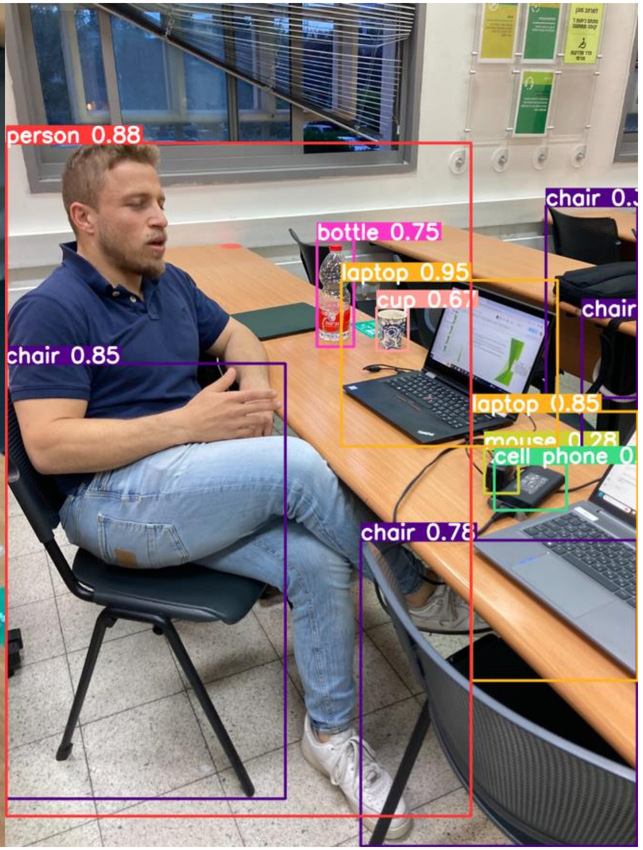


Find Bottle of water in compex scene

Object Detection

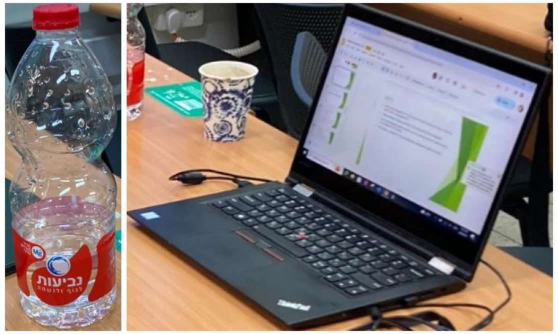
Obj to detect

Complex scene



Experiment Results

Bottle with PC

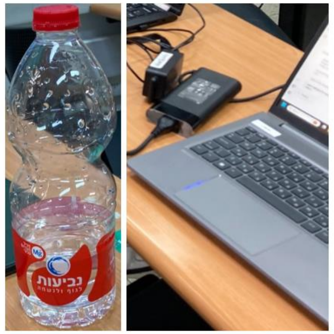


16 matches

Bottle with Yoav

6 matches

Bottle with Charger

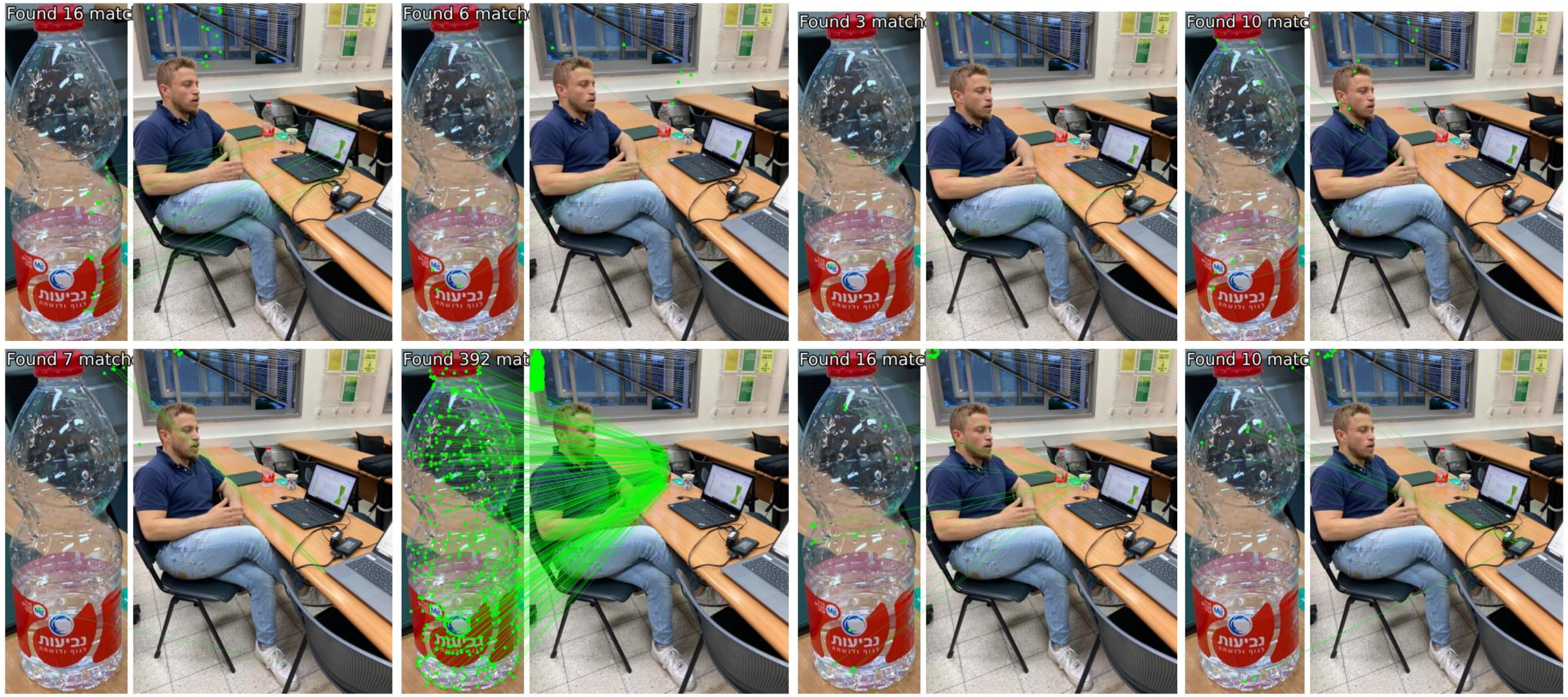


3 matches

Bottle with Chair

10 matches

Experiment Results



Feature Matching

Bottle with chair

7 matches

Bottle with Bottle

392 matches

Bottle with Cup

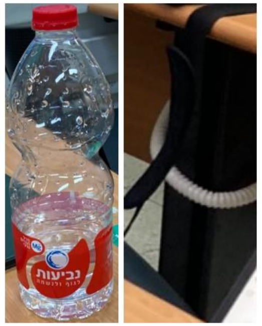
16 matches

Bottle with Charger

10 matches

Feature Matching less than 20

Bottle with Leg



Bottle with Bag

Bottle with Charger

- Object ind: 0 matches count: 16
- Object ind: 1 matches count: 6
- Object ind: 2 matches count: 3
- Object ind: 3 matches count: 10
- Object ind: 4 matches count: 7
- **Object ind: 5 matches count: 392**
- Object ind: 6 matches count: 16
- Object ind: 7 matches count: 10
- Object ind: 8 matches count: 7
- Object ind: 9 matches count: 16
- Object ind: 10 matches count: 3