Similar or identical object matching in complex scenes

Yoav Sabag
205382690

Yftach Gil
301594990

Abstract

Object matching of image pairs is a wide area of research in the field of computer vision. In this paper we present an integration of state-of-the-art models to perform similar or identical object matching.

We insert a pair of images, of object and scene. Inside the scene the same or similar object can appear with different perspectives, pose, or lightning. The main model used to match the image pair is LightGlue [1], a state-of-the-art feature matching model. For increasing performance it is integrated with YOLO, an object detection model. The assembled model of LightGlue and YOLO detects the objects and iterates over the regions of interest in the scene. Then Lightglue is applied and finds the number of matched features, accordingly the matched object is determined.

Implementing this integration into one framework achieves greater results than Lightglue alone, while tested on a customized dataset based on COCO dataset and on live experiments.

1. Introduction

Feature matching is a computer vision task which detects keypoints in two images and then matches them according to a matchability score. Counting the matched keypoints of the image pair indicates the similarity. The similarity needs to be compared with some ground truth dataset or trained models so it can be quantified.

Keypoint detection or Local features detection finds the unique interest points in the images, it is a fundamental building block of many computer vision applications like camera tracking, 3D mapping, Simultaneous Localization and Mapping (SLAM), Structure-from-Motion (SfM), camera calibration, and image matching.

For feature matching we have used LightGlue model, a deep neural network that is trained to match local features across 2 images efficiently. LightGlue is based on SuperGlue and is a plug-and-play replacement to SuperGlue. It has higher matching speed in terms of “images per second" and improved accuracy. Thanks to its efficiency and speed LightGlue is suitable for latency sensitive applications like SLAM or other real-time applications.

LightGlue was basically trained on scenes that mostly contains buildings or street view and not on smaller items such as objects. In preliminary experiments made in this work it was noticed that LightGlue is not performing well trying to detect a small object in a large scene but is excellent for matching when images content is with similar scale.

To tackle this issue, we have used YOLO-v8 [2] object detector for detecting the region of interest of each object in the image. YOLO object detection uses a neural network to locate (or segment) an area in the image that contains an object. In addition, the object is also labeled and the label's probability is shown. In our work we used the bounding box from YOLO as ROI so LightGlue was focused to a specific area and performed better relative to a full scene. This combination yields a good solution for the complicated task of object matching in a multi-object scene.

Problem Definition: Given 2 images {A, B}, A contains a single object and B is a multi-objects scene. Match the most likely object from image A in image B.

2. Related Work

The task of similar object detection can be addressed with various approaches. In this work, we have chosen to address it by employing feature matching using LightGlue.

“LightGlue: Local Feature Matching at Light Speed” (ICCV 2023) is research done by researchers from the Microsoft Mixed Reality & AI Lab at ETH Zurich. It is a deep neural network that matches local features across images. It has 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 in inference to be much faster on image pairs that are intuitively easy to match. This adaptation is achieved using keypoints pruning and after that matching for each layer. Pruning reduces the unmatchable keypoints. Keypoint that were detected as confident and unmatchable are unlikely to aid the matching of other points, such points are for example in areas that are clearly not covisible across the images, it is therefore discarded at each layer. This reduces the computation needed. After the pruning an exit criterion is applied on the layer. If a confident state is reached, the model will match the keypoints based on their pairwise similarity. This logic repeats several times until the confidence state is reached. In this way LightGlue optimizes the number of layers to a minimum so no excess computation will occur.

Lightglue was trained on the following datasets: for homography estimation with "HPatch dataset", for relative pose estimation with "MegaDepth dataset", and for Visual localization with "Aachen Day-Night dataset". The model metrics were "Image Pairs Per Second" and "Relative Pose Accuracy [%]".

Another study for matching objects is **"Efficient Object Detection and Matching using Feature Classification"** [4]. This study presents object detection and matching using feature extrusion with SIFT. After the feature extrusion, feature classification is executed using SVM and then feature matching using Nearest Neighbor classifier. Classification is made using the binary classification based on the SVM. This study showed good results with short computation time.

The study **"Image Feature Matching and Object Detection using Brute-Force Matchers"** [5] is another research in the field of object matching with good experimental results. It considers the same object detection with feature matching by exploiting concurrent algorithms for feature detection and descriptor extrusion. The feature matching is accomplished by Brute-Force combined with KNN.

3. Methodology

In this section we describe the proposed ensemble algorithm. The section is divided into the algorithm and the evaluation process. Given a trained YOLO and LightGlue models we aim to create an algorithm for detection of the most similar object. For performance evaluation we create a customized ground truth dataset from the large-scale COCO [3] dataset. The methodology flow presented in the scheme.

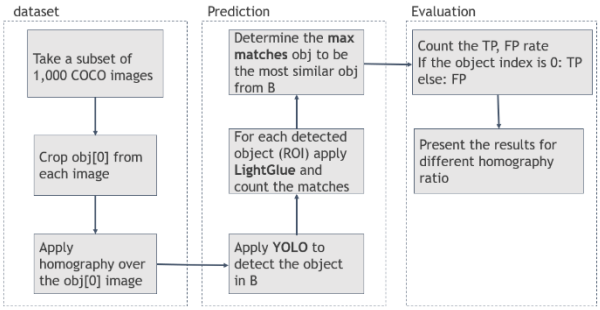

Figure1 . methodology workflow scheme: 1) create a ground truth dataset. 2) match the objects from B with A'. 3) evaluate the results.

3.1. Evaluation

Since this work is an integration of trained models an evaluation is needed for checking how good is the integration. To evaluate the performance, we created an evaluation dataset that was considered the ground truth. This new dataset was created by cropping one object from images of COCO dataset. Each cropped object was manipulated with homography and then matching with LightGlue was applied to predict the most likely object (further description on the dataset creation in section 4).

The general steps of evaluation:

1. Load 2 images from the evaluation dataset.
2. Apply YOLO on scene image (B) and get a list of object's ROI's.
3. Apply LightGlue on image (A') and each ROI from the list of scene image (B).
4. Calculate the score of each object in the scene image. The object with the maximum score is to be the most similar object.
5. Compare the maximum score object with known ground truth and calculate the precision for all images as:
   P = TP/TP+FP.

Creating Dataset Algorithm
1. input: subset of 1000 images of COCO dataset 2. Iterative thru images: detect objs in img A = objs[0] A' = homography(A) 3. output: Transformed dataset of 1000*{B, A'}

Table 1.creating dataset algorithm

3.2. Ensembled model

The ensembled model receives a pair of {A', B} images that contain an object and a scene respectively.

First, we apply YOLO over the scene image (B), to detect the objects and define the bounding box as ROI (region of interest), then, we implement LightGlue for each ROI vs the object image (A'), and we export the number of matches each object from the scene image (B) achieved with the object image (A'), the maximum matches object is determined to be the most similar object.

Object matching
1. input: pair of A',B images 2. YOLO - detect all objs in B 3. Iterative thru obj in B: a. SuperPoint - keypoints detection(A', obj) b. LightGlue - keypoints match and count(A', obj) c. similar_obj = max(matches_count(A',obj)) 4. output: similar_obj

Table 2 object matching algorithm with ensembled models.

3.3. Score

The score is the based metric to determine which object from the scene is the most likely to be the object from image A':

$$score = \frac{matches\{A', obj\}}{keypoints\{A'\}}$$

4. Creating the Evaluation Dataset

The original COCO dataset contains more than 200,000 images and 80 categories. Each image contains a single object up to dozens.

As mentioned, we have created a dataset as ground truth to evaluate performance of the ensembled model, we did so by taking a subset of 1000 images from the COCO validation dataset. These images are called "B" and contain the full scenes. For each image B YOLO model has detected the object's bounding box, label, and probability (fig. 2).

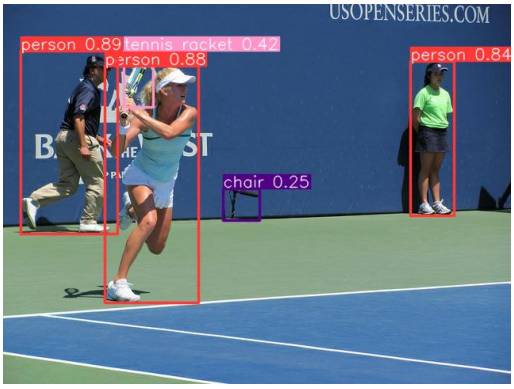

Figure 2. YOLO prediction of an image from COCO. YOLO creates a list of the object labels, probability, and bounding box.

The object with the highest probability was taken and cropped using its region of interest (ROI) and was called A. Images that had objects with probability under 50% were skipped.

After the cropping, each cropped image A was transformed using a limited random homography transformation to create images A' (fig. 3).

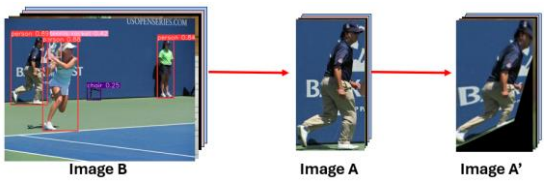

Figure 3 transformed dataset creation: we crop the object with the highest probability from YOLO list and manipulate it with homography transformation.

The homography transformation was defined by 4 source points and 4 new destination points. The source points are the corners of a rectangle in the middle of the image.

The transformation function projects the image according to the new destination points. Each destination point was randomly chosen under a certain limit box. The limit box size was taken as a percentage from the height and width of the image. The image and percentage are the input of the homography function (fig.4):

$$limit_X = percent * W$$
$$limit_Y = percent * H$$
$$corner_{new}X = cornerX + random(-limit_x \div limit_x)$$
$$corner_{new}Y = cornerY + random(-limit_y \div limit_y)$$

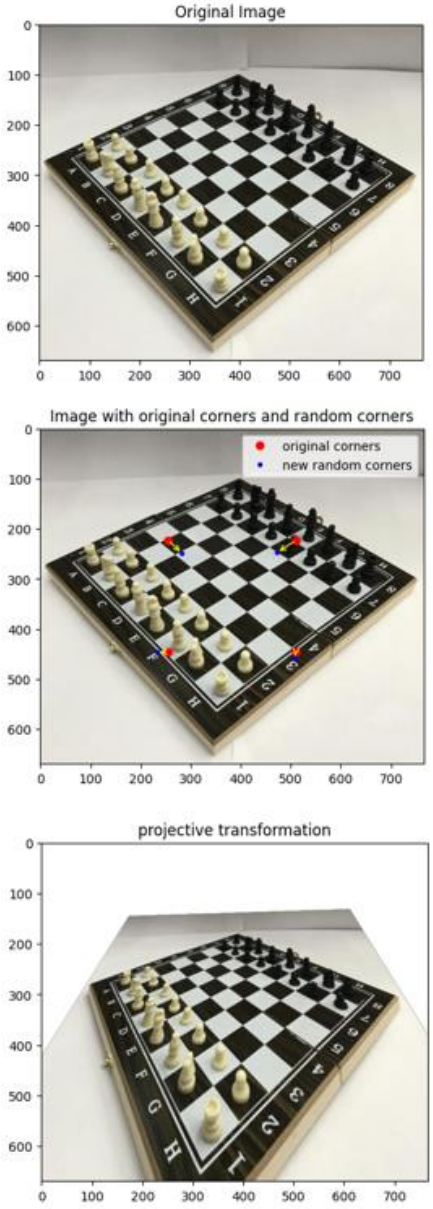

Figure 4. example of homography transformation.

The homography was applied several times starting with 5% up to a limit of 17%. At this limit the transformed images were no longer relevant since they were too much distorted for the eye and did not represent a sensible orientation.

5. Evaluation

The evaluation of the integrated models was performed over the “evaluation dataset” by comparing the model's decision to the ground truth. The decision was made by a defined score. The score is the number of matches divided by keypoints of image A'. Since the matches are equal for both images and the keypoints of each ROI differ, we take the number of keypoints from image A'. In This way we use a rather good method because the score is relative for all objects. For each object in image B we apply Lightglue, detect and count the matches with A'. If the first object gets the maximum score, we set the prediction as "True Positive" since we know that it is the ground truth. After iterating over the entire dataset, we can calculate the precision as:

$$Precision = \frac{TP}{TP + FP}$$

6. Results

The evaluation process was done several times with different homographies percentage from 5% to 17%. The following graph shows the results (fig. 5):

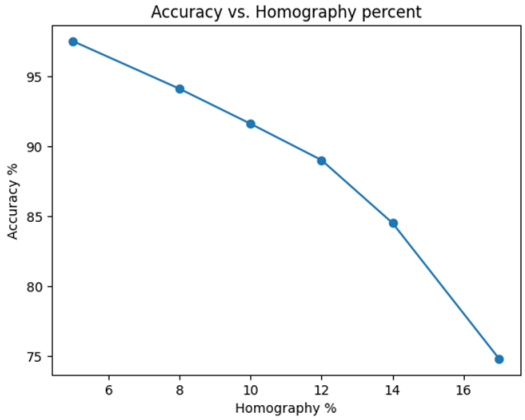

Figure 5. Results graph for homography vs Accuracy validation

As expected with a smaller homography percentage the precision is high. For 5% homography the precision was 97.5% and the lowest was 74.8% for 17% homography.

7. Experiments

We now demonstrate the effectiveness of our model. First, we present a qualitative and then quantitative result of our methodology performance. The experiments were conducted on google Colab with NVIDIA T4 and V100 GPUs.

First experiment

In the first hands-on experiment, LightGlue model was straight forward applied over the image pair. We tried to detect a specific red sport car in a scene including the same car and other similar red cars, as presented in fig. 6. The matcher matched small amount of keypoints so it was clear that the matching was insufficient. It can be observed in the figure that the matches were found mostly in the wheel area of another object.

Figure 6. First experiment: LightGlue performance for full image - top image. LightGlue performance for a region of interest -bottom image.

Under the primer knowledge that LightGlue was trained over images with different characteristics then the car images, we encountered that it was hard to match. We decided to crop the cars from the scene and apply LightGlue over again. This time the matching gave much more matches. This method was repeated for some other image pairs and the results laid a foundation for the assumption that cropping is significantly improving the matcher.

Experiment 2: find the man

In this section we took a closeup picture of our colleague in the lab in a certain pose as image A', then we took a picture of him with a scene with 8 more objects and a different pose, as demonstrated in fig. 7:

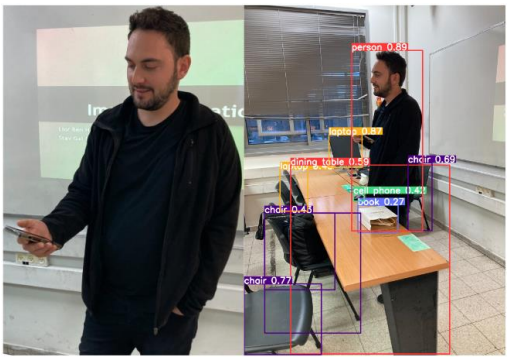

Figure 7. experiment 2: Right image - scene image with multiple objects detected with YOLO. Left image - object to detect.

We used YOLO to detect the objects in the scene and then LightGlue for feature detection and matching, last we computed the score. The object from image B (scene) that got the most matches with the single object, is the most similar object, as presented in fig. 8:

Figure 8. Qualitative performance of the ensembled model to match every object ROI from the scene with the object image. We can observe that the top right image represents the true match.

The quantitative matches results for each object from fig. 8 are presented in the following table:

Object image	Object from scene	Number of matches
Man	**Man**	**108**
Man	Laptop	6
Man	Chair (1)	14
Man	Chair (2)	3
Man	Table	9
Man	Laptop	3
Man	Chair (3)	18
Man	Cell Phone	1
Man	Book	15

Table3 .Experiment 2: quantitative results matching the object image with every object in the scene image. We observe the result and see obviously that the man with itself have received much more matches from the other objects in the scene

The same object (the man) got 5-10 times more matches than the others. All other objects received 1-18 matches. This distribution gives us confidence that the detection is clear.

Experiment 3: find the bottle in the scene

In this experiment we took a closeup picture of a bottle of water in a certain pose. Then we took a picture of a scene containing the bottle with 11 more objects.

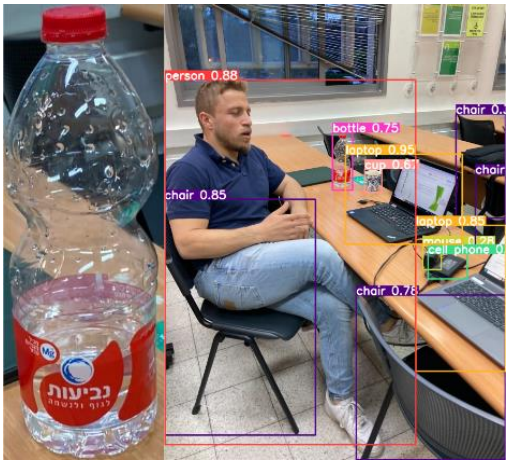

Figure 9. Experiment 3: right image - scene image with various objects detected with YOLO. Left image - object image that we try to detect.

As we did in experiment 2, we have detected the objects in the scene, and matched each object in the scene relative to the bottle and computed the score, qualitative results presented in fig. 10:

Figure 10. qualitative performance of the ensembled model to match every object ROI from the scene with the bottle image. Here we can see that the bottle to bottle match got highest number of matches.

In this experiment the model manages to define the bottle as the most similar object with the greatest number of matches and therefore the highest score.

object image	object from scene	Number of matches
Bottle	Man	6
Bottle	Chair (1)	10
Bottle	**Bottle**	**392**
Bottle	Laptop	10
Bottle	Cup	16
Bottle	Chair (2)	7
Bottle	Laptop	16
Bottle	Mouse	10
Bottle	Cell Phone	3
Bottle	Chair (3)	6
Bottle	Chair (4)	10

Table 4 . experiment 3 quantitative results. matching the object image with every object in the scene image. The object with maximum matches is marked, it has more than 20 times more matches than the other objects.

From the quantitative results present in the table, we get the understanding that the true object (bottle) has received much more matches relative to the other object in the scene which gives a reliable decision.

8. Further research

Combination of objects detection

In this work the ensembled model goal is to detect a given single object in the scene that might contain many objects. Future research may include detection of a combination of objects in the scene. For example “man+bag+hat”. This task will not be just more computational complex but will require more properties to be defined, such as relative position between the objects, and intersection over union (IoU) threshold.

Use object labels to reduce computation time

In the current work the object label was ignored. We suggest using the predicted label to improve calculation time. If the matching will be applied for objects from the same label, it is possible that a high number of matched points will be received. This will yield high confidence and iterating for more objects will be unnecessary, this might save computation time.

Raise the object detection probability limit

Raising the probability limit will result in taking less objects for matching stage and therefore less computations will be made. We assume that higher probability objects can improve the precision or at least not affect it. We suggest adding probability limits to the evaluation stage and conduct it again for values such as 0.6 to 0.8.

9. Conclusion

In this study we tackled the task of object matching using object detection and feature matching. As preliminary experiments showed, detecting an object using only feature matching can be difficult. With further research we integrated LightGlue with YOLO. We significantly improved the matching by first applying object detection to get the object's ROI and then using the feature matcher. As experiments showed, applying the matching on the object's ROI in the scene resulted a high success rate and successfully achieved our task. The integrated model was backed by evaluation using a customized homography ground truth dataset based on the COCO dataset.

References

[1] Lindenberger, P., Sarlin, P. E., & Pollefeys, M. (2023). Lightglue: Local feature matching at light speed. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 17627-17638).

[2] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788) .

[3] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13 (pp. 740-755). Springer International Publishing.

[4] Dornaika, F., & Chakik, F. (2010, August). Efficient object detection and matching using feature classification. In 2010 20th International Conference on Pattern Recognition (pp. 3073-3076). IEEE.

[5] Jakubović, A., & Velagić, J. (2018, September). Image feature matching and object detection using brute-force matchers. In 2018 International Symposium ELMAR (pp. 83-86). IEEE .