Deep Homography Estimation Report

Centro de Investigación en Matemáticas A.C.

Esteban Reyes–Saldaña[1],
esteban.reyes@cimat.mx[1]

June 1, 2023

Abstract

In the original paper [2] it was introduced a deep convolutional neural network for estimating homography in a relative way between a pair of images. The original convolutional network has 10 layers. The input is a pair of grayscale images that produces an 8 degree of freedom vector associated to a homography, useful to mp the pixels from the first image to the second. The main contrubution is that, using two types of convolutional neural: a regression network and a clasification network, it can be produce a 4-point homography witch maps the four corners from one image into the second. The model is compared with traditional homography estimator based on ORB features using Mean Corner Error Metric.

1 Introduction

2D Feature points are the basis of most modern Structure from Motion and SLAM techniques. These 2D features are tipically know as corners. A classical aproach for estimating homography bewteen images rely on the error on corner-detection method.

Estimating a 2D Homograpy (or projective transofrmation) from a pair of images is a fundamental task in computer vision. The homography is an essential part of monocular SLAM systems in scenarios such as

- Rotation only movements.
- Planar Scenes
- Scenes in witch objects are very far from the viewer.

In class, we saw that the homogeneous coordinates of points in a 3D plane are related with the homogeneous coordinates of their images thought a homography

$$p_{i_2} \propto H^{12} p^{i_1}$$

where $H12 = H^2(H^1)^{-1}$.

1.1 Classic Homography Estimation

To perform the estimation of a homography, the basic material are correspondences among the two images

$$p_k, p'_k$$

that sould correspond to the same 3D point.

The tradictional homography estimation pipeline is composed of two stages: corner estimation and robust homography estimation. Robustness is introduced by returning a large and over-complete set of points. For the second stage a RANSAC(Random Sample Consensus) technique is used for que sauqred loss function.

RANSAC is for cases where, among all the available matchings, some of theme are correct (inliners), other incorrect (outliners). A useful code seen in the course is

```
import cv2
import numpy as np

MAX_FEATURES = 500
GOOD_MATCH_PERCENT = 0.15

# Convert images to grayscale
```

```
im1Gray = cv2.cvtColor(im1, cv2.COLOR_BGR2GRAY)
im2Gray = cv2.cvtColor(im2, cv2.COLOR_BGR2GRAY)

# Detect ORB features and compute descriptors.
orb = cv2.ORB_create(MAX_FEATURES)
keypoints1, descriptors1 =
orb.detectAndCompute(im1Gray, None)
keypoints2, descriptors2 =
orb.detectAndCompute(im2Gray, None)

# Match features.
matcher = cv2.DescriptorMatcher_create(cv2.
,DESCRIPTOR_MATCHER_BRUTEFORCE_HAMMING)
matches =
list(matcher.match(descriptors1, descriptors2, None))

# Sort matches by score
matches.sort(key=lambda x: x.distance, reverse=False)

# Remove not so good matches
numGoodMatches =
int(len(matches) * GOOD_MATCH_PERCENT)
matches = matches[:numGoodMatches]

# Draw top matches
imMatches =
cv2.drawMatches(im1, keypoints1,
im2, keypoints2, matches, None)
cv2.imshow("Matches", imMatches)

# Extract location of good matches
points1 = np.zeros((len(matches), 2),
dtype=np.float32)
points2 = np.zeros((len(matches), 2),
dtype=np.float32)
for i, match in enumerate(matches):
points1[i, :] = keypoints1[match.queryIdx].pt
points2[i, :] = keypoints2[match.trainIdx].pt
# Find homography
h, mask = cv2.findHomography(points1, points2,
cv2.RANSAC)
```

The autors claims that that they want a single robust algorithm that, given a pair of images, simply returns the homography relating the pair without the corner-ish features, line-ish features, etc.

As seen in the last section, the simpletst way to parametrize a homography is with a $3x3$ matrix and a fixed scale. The homography maps $[u, v]$, the pixels in the left image to $[u', v']$, the pixels in the right image and it is define (up to scale) by

$$\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \sim \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad (1)$$

and it is well know that

$$\begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix} \quad (2)$$

refers to the rotational terms in the homography, while

$$[H_{13}, H_{23}] \quad (3)$$

is the translational offset.

1.2 The 4-Point Homography Parametrization

Suppose that

$$U_i^k = (u_i^k, v_i^k)^T \quad (4)$$

for $k = 1, 2, 3, 4$ are 4 fixed point in the image i and

$$U_j^k = (u_j^k, v_j^k)^T \quad (5)$$

are variable points in the image j such that

$$H_{ij} U_i^K = U_j^k \quad (6)$$

for each pair of corresponding points, we have

$$\begin{aligned} h_1 x_i + h_2 y_i + h_3 - x_i x_i' h_7 - y_i x_i' h_8 - x_i h_9 &= 0 \\ h_4 x_i + h_5 y_i + h_6 - x_i y_i' h_7 - y_i y_i' h_8 - y_i' h_9 &= 0 \end{aligned}$$

and if we know 4 pair of points, considering $h_9 = 1$, the system can be solved using Direct Linear Transoformation [3].

1.3 Corner Location Parameterization

Balancing the rotational and translational terms as part of an optimization problem is difficult. The main reasoning of this is explained in [4]. The rotation and shear components tend to have a much smaller magnitude than the translation component (angles vs location) and as a result although an error in their values can greatly impact H.

The impact in the $L2$ loss will be minimal. In addition, the high variance in the magnitude of the elements of the 3×3 homography matrix makes it difficult to enforce H to be non-singular. The 4-point parameterization does not suffer from these problems since there are all magnitudes refering to a position rate.

4.1 The Direct Linear Transformation (DLT) algorithm 91

Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the 2D homography matrix $\mathtt{H}$ such that $\mathbf{x}'_i = \mathtt{H}\mathbf{x}_i$.

Algorithm

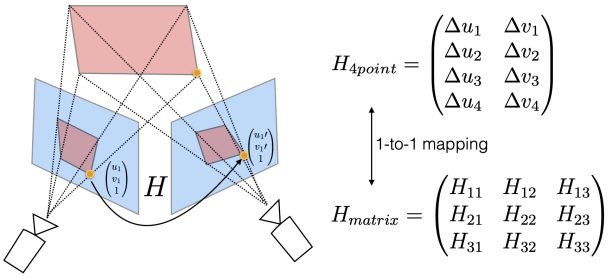
(i) For each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ compute the matrix $\mathtt{A}_i$ from (4.1). Only the first two rows need be used in general.

(ii) Assemble the n 2×9 matrices $\mathtt{A}_i$ into a single $2n \times 9$ matrix $\mathtt{A}$.

(iii) Obtain the SVD of $\mathtt{A}$ (section A4.4($p585$)). The unit singular vector corresponding to the smallest singular value is the solution $\mathbf{h}$. Specifically, if $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^\mathsf{T}$ with $\mathtt{D}$ diagonal with positive diagonal entries, arranged in descending order down the diagonal, then $\mathbf{h}$ is the last column of $\mathtt{V}$.

(iv) The matrix $\mathtt{H}$ is determined from $\mathbf{h}$ as in (4.2).

Figure 1: DLT Pseudocode



$$H_{4point} = \begin{pmatrix} \Delta u_1 & \Delta v_1 \\ \Delta u_2 & \Delta v_2 \\ \Delta u_3 & \Delta v_3 \\ \Delta u_4 & \Delta v_4 \end{pmatrix}$$

$$H_{matrix} = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix}$$

Figure 2: Homography Map

An alternative parametrization, based on a single kind of location variable, namely **corner location** [1]is more suitable for our deep homography estimation.

Let

$$\Delta u_1 = u'_1 - u_1 \tag{7}$$

be the u-offset for the first corner,the 4 point parametrization represents a homography as follows

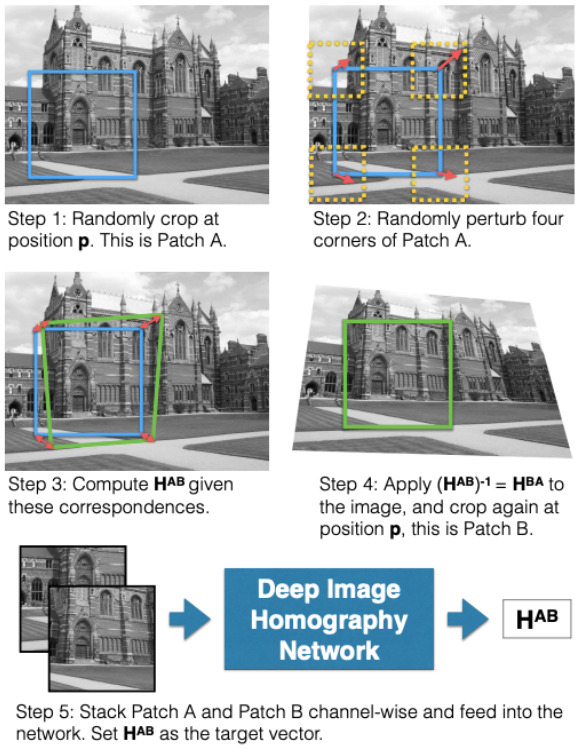
$$H_{4points} = \begin{pmatrix} \Delta u_1 & \Delta v_1 \\ \Delta u_2 & \Delta v_2 \\ \Delta u_3 & \Delta v_3 \\ \Delta u_4 & \Delta v_4 \end{pmatrix} \tag{8}$$

and once the displacement of the four corners is known, one can easily convert $H_{4points}$ to H using several methods, such as DLT or the function $getPerspectiveTransofrmation()$ in OpenCV.

2 Method

2.1 Data Generation for Homography Estimation

Deep Learning algorithms requires a large amount of data. To meet this requirement, it is generated a almost unlimited number of labeled training examples by applying random projective transformations to a large dataset.



The process is list as follow

1. Randomly crop a square patch from a large image I at position p (avoiding borders) called I_p.
2. Four corners of Patch a are randomly perturbed by values within the range$[-\rho, \rho]$.
3. The four correspondeces define a homography H_AB.
4. Compute $H^{BA} = (H_{AB})^{-1}$.
5. Apply H^{BA} to the large image to produce image I'.
6. A second patch I'_p is cropped from I' par position p.

7. The training example is a tuple. The first element is a stacked channelwise image containing I_p, I'_p and the second element is $H_{4points}$.

2.2 Convnets Models

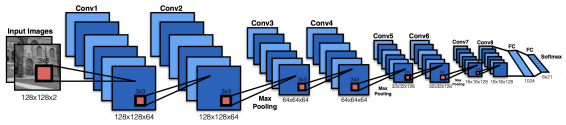
The networks use 3×3 convolutional bnloks with BatchNorm and ReLUs activations. The input for both network configuration are a grayscale pair of images of size

$$128 \times 128 \times 2 \tag{9}$$

and they have 8 convolutional layers with filtering

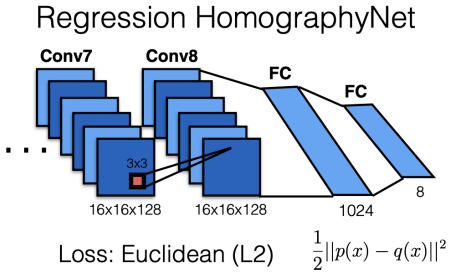
$$64, 64, 64, 64, 128, 128, 128, 128 \tag{10}$$

and finally it is connected to a two fully connected layers. The first fully connected layer has 1024 units. Dropout with a probability of 0.5 is applied after the final convolutional layer and the fisrt fully connected layer.



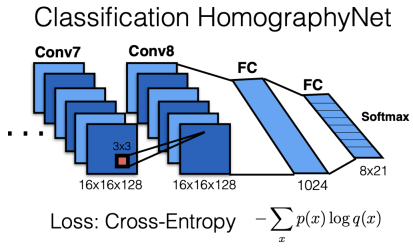
2.2.1 Regression Network

it produces directly 8 real values numbers and Euclidean L2 loss is applyed during training.

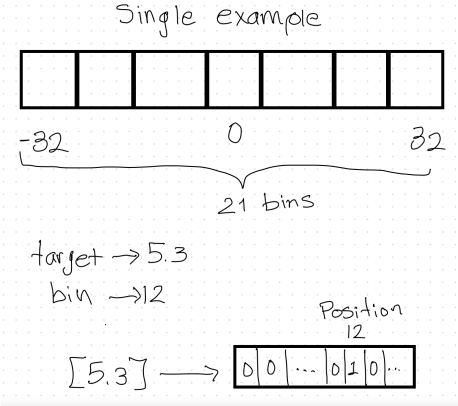


2.2.2 Classification Network

Use a quantization scheme, has a softmax at the last layer and we use the cross entropy loss function during training. 21 bins was chosen for each of the 8 outputs, which results in a final layer with 168 output neurons.



The process for creating bins starts with defining the number of bins in which the interval will be divided. In this case, if $\rho = 32$, the range will be $[-32, 32]$. Then each bin will correspond to a subinterval range. So we take the original label and transform it to the corresponding one hot vector acording to the bin representation.



3 Experiments

The optimizer for all the network was **SGD** with a learning rate of 0.005 and momentum of 0.9. The learning rate was decrease by a factor of 10 after every 30,000 iterations.

The network are trained for 90,000 total iterations using a batch size of 64.

For creating the dataset, MS coco 2017 was used. The main configuration is

Item	Value
Img_size	$(320, 480)$
patch_size	$(128, 128)$
ρ	32
num_samples	500000
Iterations	90000

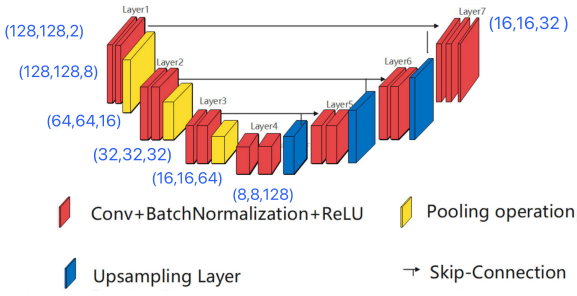
and for the testing dataset

Item	Value
Img_size	$(640, 480)$
patch_size	$(256, 256)$
ρ	64
num_samples	5000

It is compared with the traditional Homography Estimation using ORB descriptor + RANSAC $+GetPerspectiveTransform()$ OpenCV Homography Computation. When $ORB + RANSAC$ estimate is too extreme it return identity estimate.A second baseline is perform where for every data in the test set a 3×3 matrix is returned.

3.1 Unet Backbone

In the aim of reduce the number of parameters using a more complex network architecture, it is use a UNet for feature extracting.



3.1.1 Results

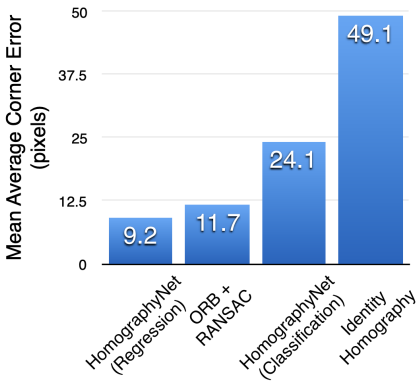
The training parameters and the time for a single epoch for each model are present in the table

Model	num params	Time per Epoch
Reegression	34,193,800	1.63 min
Classification	34,357,800	1.62 min
Unet Reg	8,899,592	1.21 min

According to the Mean Average Corner Error, we get

Model	MSE Training	MACE
Unet Reg	0.016	9.98
Reegression	0.020	7.44
Classification	0.019	44.00
ORB/RANSAC	-	12.07
Identity	-	52.11

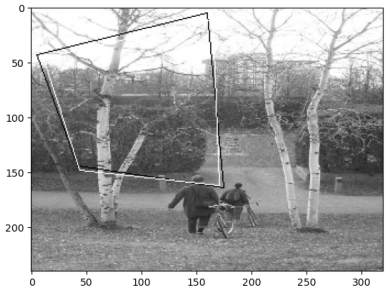
that is similar to the results of the original paper



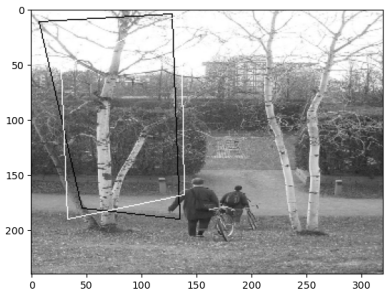
3.2 Image Results

For visual comparation, it is shown some examples from the testing set

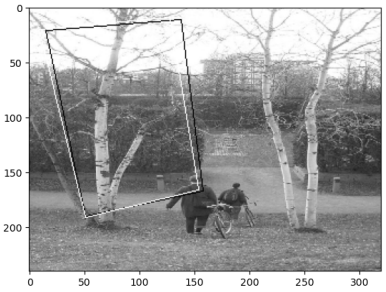
Regression Model



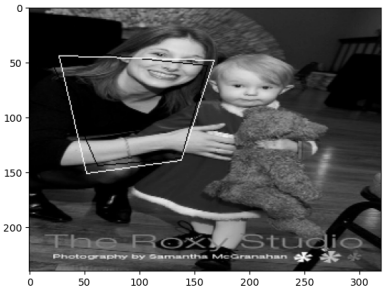
Classification Model



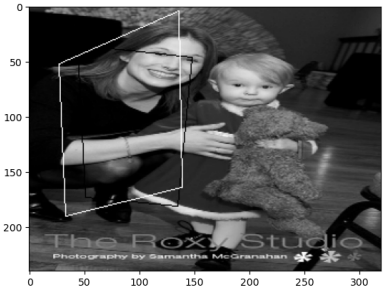
Unet Reegression



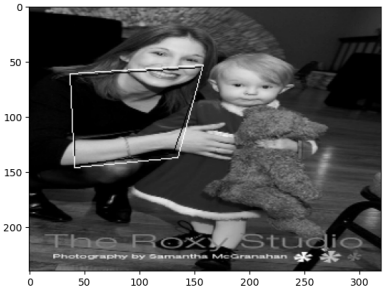
Regression Model



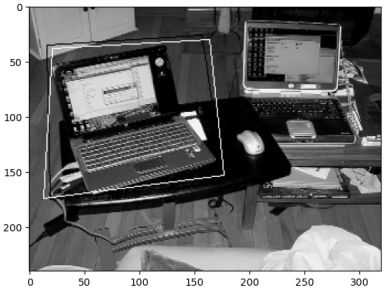
Classification Model



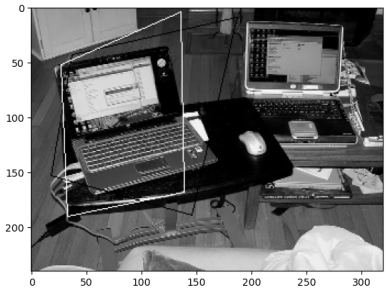
Unet Reegression



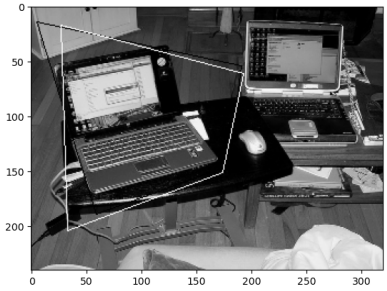
Regression Model



Classification Model



Unet Reegression



4 Conclusions

In this paper, it was introduced a new method for estimating homography between two gray scales images. Two convolutional newt works were present fro extracting image spatial features. Additionally, it was introduced an algorithm for creating a synthetic dataset which is a tuple of two image grayscale images and the homography given 4 corners.

This paper is from 2016 and it was the first for estimating homographies from a deep model. Many suggesting can improve the model

- Introduce a metric that takes into account spatial features.
- Convolutions only sees local context, maybe some attention mechanism could work.
- The dataset is syntetic, it must be compare with real life examples.
- Investigate another metrics for a better comparatation.
- Adding a unet shows that a small network works better for extracting features, maybe a residual scheme could work better.

References

[1] Baker, S., Datta, A., and Kanade, T. Parameterizing homographies. Tech. Rep. CMU-

RI-TR-06-11, Carnegie Mellon University, Pittsburgh, PA, March 2006.

[2] DeTone, D., Malisiewicz, T., and Rabinovich, A. Deep image homography estimation, 2016.

[3] Hartley, R., and Zisserman, A. *Multiple View Geometry in Computer Vision*, 2 ed. Cambridge University Press, New York, NY, USA, 2003.

[4] Nguyen, T., Chen, S. W., Shivakumar, S. S., Taylor, C. J., and Kumar, V. Unsupervised deep homography: A fast and robust homography estimation model, 2018.