Vehicle Speed Estimation Using Computer Vision And Evolutionary Camera Calibration

Hector Mejia
School of Mathematical and Computational Sciences
Yachay Tech University
Urcuqui, Ecuador
hector.mejia@yachaytech.edu.ec

Esteban J. Palomo
Department of Computer Languages and Computer Science
University of Málaga
Málaga, Spain
ejpalomo@lcc.uma.es

Ezequiel López-Rubio
Department of Computer Languages and Computer Science
University of Málaga
Málaga, Spain
ezeqlr@lcc.uma.es

Israel Pineda
School of Mathematical and Computational Sciences
Yachay Tech University
Urcuqui, Ecuador
ipineda@yachaytech.edu.ec

Rigoberto Fonseca-Delgado
Department of Electrical Engineering
Metropolitan Autonomous University
Mexico, Mexico
rfonseca@izt.uam.mx

Abstract

Currently, the standard for vehicle speed estimation is radar or lidar speed signs which can be costly to buy and maintain. However, most major cities already implement networks of traffic surveillance cameras that can be utilized for vehicle speed estimation using computer vision. This work implements such a system using homography estimation, YOLOv4 object detector, and an object tracker capable of vehicle speed estimation. The homography component uses world plane-image plane point correspondences, located by humans. Moreover, a new method is developed specifically for this use case, using the estimation of density evolutionary algorithm. It aims at correcting the points misalignment in between planes. In addition, a basic direct linear transformation (DLT) and a random sample consensus robust version of DLT are implemented for comparison. Finally, the results show that the proposed homography method reduces the projection error from world to image point by 97%, when compared to the other two methods, and the complete workflow can successfully estimate speed distributions expected from vehicles on urban traffic and handle dynamic changes in vehicle speed.

1 Introduction

Each year, approximately 1.35 million people die on roadways worldwide due to speeding drivers; public entities have to deal with the cost of damage control and lidars for speed estimation are expensive to buy and maintain. In addition, most major cities have implemented a grid of surveillance cameras in areas of interest to tackle this problem.

This work[1] proposes a three-component workflow to develop a speed estimation system using only images and point correspondences between the image plane and world plane. First, homography estimation is employed, assuming a flat road to back-project image coordinates to longitude-latitude coordinates. A human operator defines a set of correspondences between the two planes to estimate the homography matrix for each scene. Then, an object detector locates vehicles on the scene. Later, an object tracker receives those vehicle detections and uses intersection over union (IoU) to assign them to vehicle tracks, and estimate speed by computing haversine distance between two points on a fixed window of time. Moreover, three implementations of homography estimation were compared. The first is a basic Direct Linear Transformation (DLT) method, the second is a robust version of DLT using Random Sample Consensus (RANSAC), and the third is the proposed novel method based on the Estimation of Density evolutionary algorithm (EDA). This proposed method modifies the longitude-latitude inputs to minimize the projection error loss function. Therefore, correcting the misalignment between a point in the image plane and the world plane.

2 Previous Work

There are many works that try to address the problem of vehicle speed estimation using machine vision. For instance, Tang *et al.* [13] introduced a methodology that consists in the calibration of a pinhole camera model based on vanishing points, followed by object detection using YOLO9000 [10], and a customized tracking system. This tracker uses various visual a spatial features to produce reliable tracks. This work achieved the first place on the NVIDIA AI City Challenge 2018, achieving a RMSE of 4.0963 mi/h. Then, the work of [15] proposes a similar workflow. However, the calibration is based on perspective transformation leveraging road markings. Their work assumes that markings follow standard dimensions. Furthermore, the authors used them to establish a real world coordinate system, and find a homography matrix. This method only uses a single scene and achieved an average error of 3.2 km/h.

3 Methodology

3.1 Algorithm Design

This method has three main components: homography matrix estimation, object detection, and object tracking. Figure 1 depicts the system workflow with the aforementioned components. The figure also includes all inputs and outputs, which are explained in detail in the following subsections.

3.2 Data Collection

To compute homography matrices, a video dataset with both image and world points representing the same spots is needed. At the time of writing this work, no such dataset was found. Instead, a compilation of 10 live feeds from different cameras in unconstrained conditions were recorded using the Department of Transportation of Seattle portal (`https://web6.seattle.gov/travelers/`). In addition, street addresses from each scene were extracted. Then, those addresses were visited on Google Maps to find the scene and manually extract world points that matched the image plane. Moreover, as the dataset was built from scratch, no beforehand speed annotations of the vehicles are available. Table 2 of Appendix A presents the metadata of all the videos.

[1]The code for this work can be fount at `https://github.com/hector6298/titulacion_vehice_speed_estimation`.

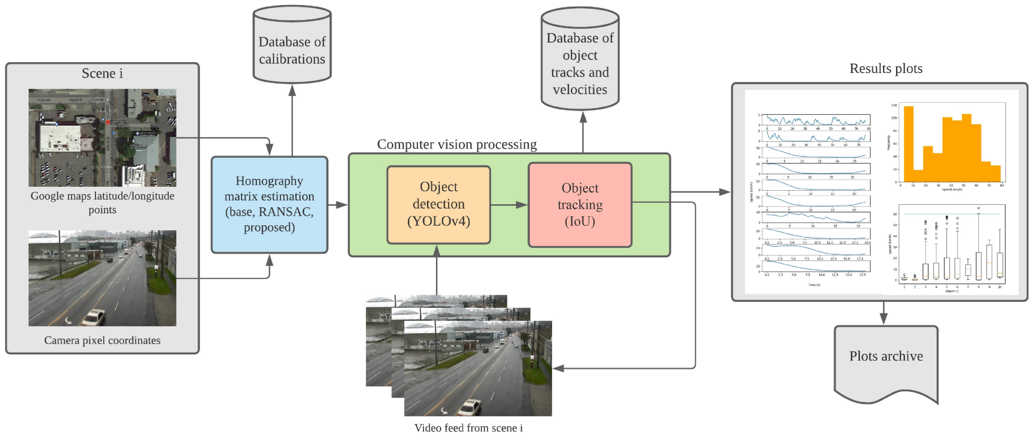

Scene i
Google maps latitude/longitude points
Camera pixel coordinates
Database of calibrations
Homography matrix estimation (base, RANSAC, proposed)
Computer vision processing
Object detection (YOLOv4)
Object tracking (IoU)
Database of object tracks and velocities
Video feed from scene i
Results plots
Plots archive

Figure 1: Proposed speed estimation workflow.

3.3 Homography Estimation for Camera Calibration

A critical component for a system, capable of measuring the speed of the objects in the scene using only visual information, is camera calibration. Assuming a flat road, let $s \in \mathbb{R}$ be a scale factor, $\boldsymbol{p'} = \begin{pmatrix} x' & y' & 1 \end{pmatrix}^T$ and $\boldsymbol{p} = \begin{pmatrix} x & y & 1 \end{pmatrix}^T$ be the homogeneous coordinates of a point in the image and world plane, respectively. Then, the camera model [12] is expressed as:

$$s \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{1}$$

The parameters of **H**, the homography matrix, can be obtained with Direct Linear Transformation (DLT) [4] with a minimum of four point correspondences. In this work, given a scene i, a set of n points are manually identified both for the pixel coordinates from a reference image of the scene and for the latitude/longitude coordinates taken from Google Maps. Each correspondence between two points should represent the same location on the scene. Furthermore, three methods are evaluated: a base DLT homography extraction with no robust method for error minimization, a RANSAC [5] DLT robust method, and a proposed technique, designed specifically for roadside camera calibration, based on evolutionary algorithms.

Given the initial manual annotations, the proposed calibration method consists of an iterative, localized correction of each starting latitude-longitude point on a scene to estimate image coordinates that match the annotations as much as possible. Intuitively, this procedure tries to alleviate the miss-alignments between real-world and image point correspondence introduced by human operation. It follows an Estimation of Density evolutionary algorithm (EDA) [1], guided by the projection error:

$$\epsilon = \frac{1}{n} \sum_{i=1}^{n} ||p_i - \hat{p}_i||_2, \tag{2}$$

where p_i is a point selected by a human operator on the image plane, and $\hat{p}_i$ is the estimated point given a longitude-latitude coordinate. The EDA algorithm works by establishing an initial population N of point correspondences with a pre-defined range where a random variation of the original points can be generated. Then, generation by generation, K individuals are selected with the least projection error. Each generation, the mean and variance of the survivors are calculated to generate offspring and the process repeats until error convergence. Note that, although all points from both planes are manually selected, the reasons for subjecting longitude-latitude points to correction instead of the image coordinates is that projection error uses image coordinates as target values and, most importantly, the world plane has no perspective. This allows a uniform initial search range for every point.

3.4 Object Detection with YOLOv4

To be able to detect vehicles on the scene, YOLOv4 detection model, created by Bochkovskiy *et al.* [3], was utilized. It is a deep convolutional neural network that consists of three main components: backbone, neck, and head. First, a network called CSPDarknet53 is embedded into YOLO as its feature extractor backbone. Based on its original version Darknet-53, it is implemented as a Cross Stage Partial Network (CSPNet) [14]. For the neck of the detector, a Spatial Pyramid Pooling (SPP) block was included over the backbone to generate a fixed-length vector regardless of image size [7]. YOLOv4 detector also performs feature extraction at three spatial dimensions to improve detection at varying object sizes. The mechanism follows a Feature Pyramid Network (FPN) [9] in which the feature maps gradually decrease in the spatial dimension but are later up-scaled with deconvolution layers. Finally, the detection head is taken from the previous version, YOLOv3 [11], in which each output feature map is subjected to a 1×1 convolutional layer with shape $1 \times 1 \times (B(5 + C))$, where B is the number of objects that can be detected on each image cell and C the number of classes.

3.5 Object Tracking and Speed Estimation

The object tracker follows a tracking by detection scheme from Bochinsky *et al.* [2], with modifications to keep track of the speed in km/h along the path of the objects. At the first image frame, the tracker assigns all detections to new active tracks. Then, the algorithm updates itself each frame by receiving new detections and assigning them to the tracks that maximize the intersection over union (IoU) using bounding boxes. In addition, an assignment threshold is defined to avoid identity hijacking or location jumps [8]. Furthermore, the speed estimation process is made every K number of frames for each track independently, using the following formula:

$$S = \frac{d(p,q)}{t} = \frac{d(p,q)}{f_s^{-1}} \times 3600, \tag{3}$$

where $d(p,q)$ is distance in kilometers, f_s is the number of frames per second taken by the camera, and f_s^{-1} represent the time spent each speed estimation. Each time the speed estimation is performed for a vehicle, p takes the value of q, and q becomes the newest location available for that vehicle. To calculate distances, recorded locations in image points have to be transformed to the real world plane using the inverse of the homography matrix $\mathbf{H}^{-1}$. Then, the distance in kilometers is computed using the haversine distance [6]:

$$d_h(p,q) = 2 \arcsin \sqrt{\sin^2\left(\frac{p_x - q_x}{2}\right) + \cos(p_x)\cos(q_x)\sin^2\left(\frac{p_y - q_y}{2}\right)}, \tag{4}$$

where p_x, q_x are latitudes and p_y, q_y are longitudes.

4 Results and Discussion

4.1 Homography Estimation

Figure 2 presents quantitative results for the base, RANSAC, and proposed version of the calibration. On average, both RANSAC and base algorithms got an error of 7.99 pixels, while the proposed returned only 0.24. This represents a reduction of 97% in projection error. In particular, video 4 had the highest error on both base and RANSAC algorithms, with estimations significantly deviating from targets. On the other hand, with the proposed methodology, the estimations match the annotations correctly, as can be seen in figures 3a and 3b, which depict a graphical comparison with the calibration methodologies. This pattern repeats for all the other videos, except for videos 6 to 10. These videos had negligible error for all the three methodologies, and they all had only 4 point correspondences, placed at the corners of the region of interest.

4.2 Speed Estimation

Distributions of speed across all the region of interest were computed for every video instance to assess the performance of the speed estimation methodology. These distributions are depicted in

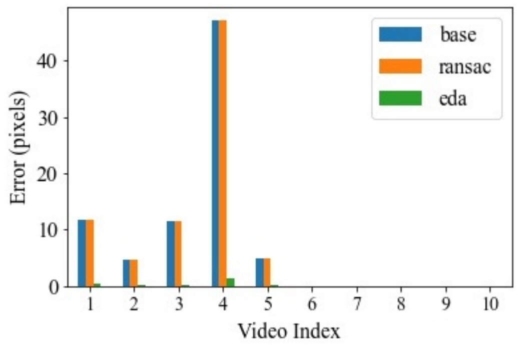
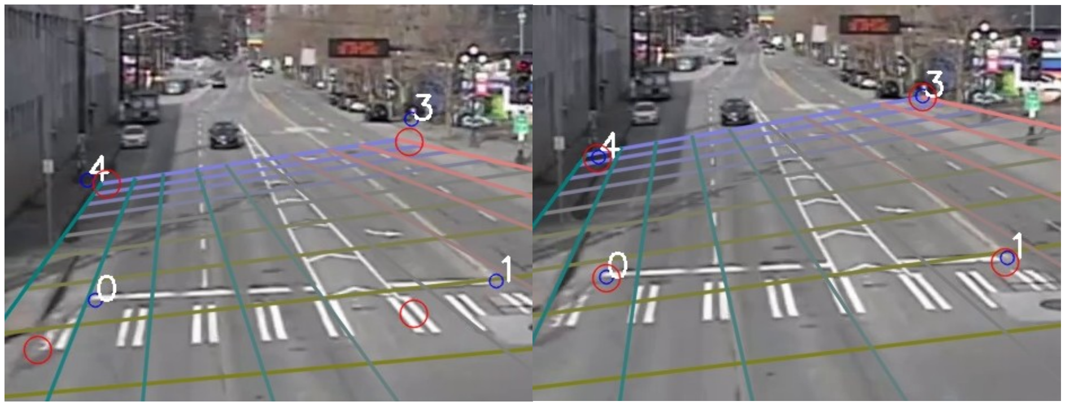


Figure 2: Projection error on all videos.



(a) Base Calibration.

(b) EDA calibration.

Figure 3: Calibrations on video 4. Blue circumferences represent annotations, red circumferences represent estimations.

Figure 4 using boxplots. Also, Table 1 shows statistics computed on the same videos: number of samples, mean speed, median speed, and standard deviation.

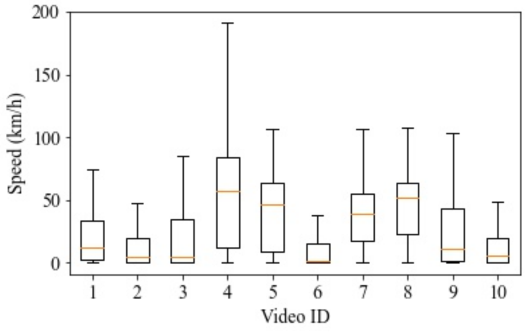


Figure 4: Boxplots for the speed distribution on all videos.

Most of the distributions per video tend to be skewed towards 0 km/h. This is due to traffic lights that are placed next to the cameras that were filming the area, and cars often had to stop, or were parked. Moreover, the trackers were still measuring vehicles at full stop. Then, the majority of speed values that were obtained, per video, are less than 70 km/h, as can be seen in Table 1. However, there is still some exceptions. For instance, Figure 4 shows that video 4, have its fourth quartile on ranges greater than 100 km/h. As this methodology estimates the speed of each vehicle independently, these measures were stored as time series. Figure 5 shows the speed timeline for ten different vehicles for video 7, which were randomly chosen.

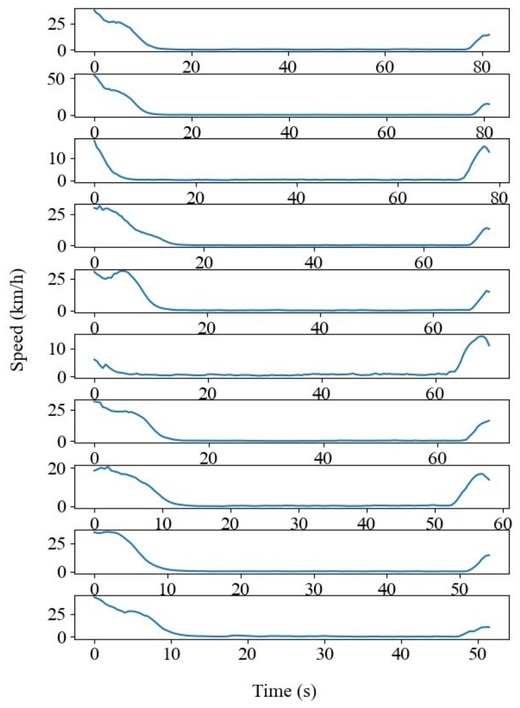


Figure 5: Timeseries with measurements of speed for 10 vehicles in video 7.

It can be seen that the tracker algorithm is able to measure the change in velocity of the vehicles. There are no sudden spikes in speed or noisy recordings, and the change in speed follows soft curves. These insights show that the tracker measures are on par with the real behavior of drivers on urban areas.

Table 1: Speed statistics for all videos.

Video Id	Speed samples	Mean speed (Km/h)	Median speed (Km/h)	Speed STD (Km/h)	percentage of speed samples > 70 Km/h
1	1049	19.53	12.18	19.01	0.00
2	2289	10.74	5.09	12.73	0.00
3	5948	17.26	5.26	19.97	0.00
4	2444	55.66	57.49	44.18	0.37
5	2610	41.83	47.00	28.28	0.18
6	8581	9.69	1.99	13.93	0.00
7	4814	37.00	39.52	23.01	0.06
8	6063	44.15	51.62	25.89	0.14
9	5168	23.98	10.83	26.15	0.07
10	6470	10.52	5.43	11.39	0.00

5 Conclusions

There are two main conclusions that stand out from the results of this work. The first is that the proposed calibration method, employing EDA evolutionary algorithm, resulted in a dramatic reduction of the projection error of all scenes, when compared to both the base DLT and RANSAC DLT methods.

Second, in all scenes, the speed estimation distributions along with the vehicles speed timeseries resembled the behavior of drivers on an urban city.

References

[1] Rubén Armañanzas, Iñaki Inza, Roberto Santana, Yvan Saeys, Jose Luis Flores, Jose Antonio Lozano, Yves Van de Peer, Rosa Blanco, Víctor Robles, Concha Bielza, et al. A review of estimation of distribution algorithms in bioinformatics. *BioData mining*, 1(1):1–12, 2008.

[2] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *International Workshop on Traffic and Street Surveillance for Safety and Security at IEEE AVSS 2017*, Lecce, Italy, August 2017.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[4] Elan Dubrofsky. Homography estimation. *Diplomová práce. Vancouver: Univerzita Britské Kolumbie*, 5, 2009.

[5] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[6] Kenneth Gade. A non-singular horizontal position representation. *Journal of Navigation*, 63(3):395–417, 2010.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[8] Md Zahidul Islam, Md Shariful Islam, and Md Sohel Rana. Problem analysis of multiple object tracking system: A critical review. *IJARCCE*, 4(11):374–377, 2015.

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[10] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[12] Peter Sturm and Srikumar Ramalingam. *Camera models and fundamental concepts used in geometric computer vision*. Now Publishers Inc, 2011.

[13] Zheng Tang, Gaoang Wang, Hao Xiao, Aotian Zheng, and Jenq-Neng Hwang. Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 108–115, 2018.

[14] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[15] Wan-Ping Wu, Ying-Cheng Wu, Chih-Chun Hsu, Jenq-Shiou Leu, and Jui-Tang Wang. Design and implementation of vehicle speed estimation using road marking-based perspective transformation. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–5. IEEE, 2021.

A Appendix

A.1 Experimental setup

All experimentation was performed on an Acer Aspire-E5-576G laptop running Ubuntu 20.10 with Intel® Core™ i5-8250U CPU, Nvidia GeForce MX150 GPU with Compute Unified Device Architecture (CUDA) enabled, 12 Gb of DDR4 RAM, and 500 GB SSD M.2 NVME. Although CUDA is enabled, it could not be used for deep learning acceleration due to the limited graphics memory available. Thus, CPU had to be utilized to make the computations of the object detection model, significantly increasing the processing time. For future work, more robust hardware is needed to assess the performance of this workflow in real-time.

A.2 Metadata of the dataset

All the addresses, date of recording and number of points extracted can be seen in Table 2.

Table 2: Overview of the videos captured from Seattle Dept. of Transportation.

Video Id	Date	Address	Points Num.	Resolution	Duration
1	February, 22, 2021	2nd Ave & Marion St	8	1280×720	8:21
2	February, 22, 2021	E Marginal Way S & S Idaho St	8	1920×1080	7:52
3	February, 22, 2021	23rd Ave E & E Madison St EW	7	1280×720	10:17
4	February, 22, 2021	1st Ave S & S Royal Brougham Way	7	1280×720	10:08
5	February, 22, 2021	Airport Way S & S Industrial Way	7	1280×720	10:31
6	March, 10, 2021	23rd Ave S & S Jackson St	4	1920×1080	10:22
7	March, 10, 2021	Airport Way S & S Lander St	4	1920×1080	9:44
8	March, 10, 2021	E Marginal Way S @ Hudson St	4	1920×1080	9:34
9	March, 10, 2021	1st Ave & Seneca St	4	1920×1080	10:25
10	March, 10, 2021	Fairview Ave & Denny Way	4	1920×1080	10:04

A.3 Algorithm and hyperparameters used for the proposed homography estimation method

The homography estimation process using estimation of density evolutionary algorithm (EDA) follows Algorithm 4, which also uses auxiliary functions defined in algorithms 1, 2, and 3. Furhtermore, the hyperparameters are:

- Initial population (N): 20000
- Selected population per generation (K): 100
- Number of generations (τ): 20
- Initial variability range (γ): 10%

Algorithm 1: EDA population initialization for world point sets (EDAPopInitialization).

Input : γ : Percentage of variation, N: Initial Pop., X': Initial set of world points.
Output : WorldPtSets

```
WorldPtSets ← {};
for i=0 to N, step=1 do
    for x in X do
        $\hat{x} \leftarrow randomVariation(x, \gamma)$ ;
        $\hat{X} \leftarrow \hat{X} \cup \hat{x}$;
    $WorldPtSets \leftarrow WorldPtSets \cup \hat{X}$
```

Algorithm 2: EDA population initialization from statistics (genPopFromStats).

Input : σ : World point set Variance vector, N: Initial Pop., μ: World point set Mean vector.
Output : WorldPtSets

```
WorldPtSets ← {};
for i=0 to N, step=1 do
    for i=0 to (num points on original set), step=1 do
        $\hat{x} \leftarrow randomVariation(\mu, \sigma)$ ;
        $\hat{X} \leftarrow \hat{X} \cup \hat{x}$;
    $WorldPtSets \leftarrow WorldPtSets \cup \hat{X}$
```

Algorithm 3: Mean and variance estimation (getMeanVariancePtSets).

Input : WorldPtSets: A set of sets of world points.
Output : μ, σ: Mean and variance vectors.

```
$\sigma_x, \sigma_y \leftarrow$ compute variance vector per coordinate of for all point sets;
$\sigma \leftarrow concatenate(\sigma_x, \sigma_y)$;
$\mu x, \mu y \leftarrow$ compute means vector per coordinate of for all point sets;
$\mu \leftarrow concatenate(\mu x, \mu y)$;
```

Algorithm 4: Calibration with estimation of density algorithm.

```
Input  : $X$: Original set of world points, $X'$: Original set of image points, $(N, \tau, \gamma, \alpha)$: EDA
         params
Output : $\hat{X}_{out}$: Corrected set of world points, $\hat{H}_{out}$: estimated calibration matrix.
1  $WorldPtSets \leftarrow EDAPopInitialization(X, \gamma, N)$;
2  while $i < \tau$ do
3      for $\hat{X}$ in WorldPtSets do
4          $\hat{H} \leftarrow$ DLT_homography($\hat{X}$, $X'$);
5          $\hat{X'} \leftarrow \{\}$;
6          $Errors \leftarrow \{\}$;
7          for $\hat{x}$ in $\hat{X}$ do
8              $\hat{x'} \leftarrow \hat{H}\hat{x}$;
9              $\hat{X'} \leftarrow \hat{X'} \cup \hat{x'}$;
10         $\epsilon \leftarrow$ calcProjectionError($\hat{X'}$, $X'$);
11         $Errors \leftarrow Errors \cup \epsilon$;
12         Associate $\epsilon$ with the corresponding $\hat{X}$ that produced $\hat{X'}$;
13     $\bar{\epsilon}_{current}, std_{\epsilon} \leftarrow$ get_mean_std(Errors);
14     sort WorldPtSets according to their associated $\epsilon$;
15     keep first $K$ world point sets $\hat{X}$ in WorldPtSets;
16     $\sigma, \mu \leftarrow getMeanVariancePtSets(WorldPtSets)$;
17     $WorldPtSets \leftarrow WorldPtSets \cup genPopFromStats(\sigma, \mu, N)$;
18 $\hat{X}_{out} \leftarrow$ first set of $WorldPtSets$;
19 $\hat{H}_{out} \leftarrow DLT\_homography(\hat{X}_{out}, X')$;
```