Image Based Localization with Simulated Egocentric Navigations

Santi A. Orlando[1,2], Antonino Furnari[1], Sebastiano Battiato[1] and Giovanni M. Farinella[1]
[1]*Department of Mathematics and Computer Science, University of Catania, Catania, Italy*
[2]*DWORD - Xenia Progetti s.r.l., Acicastello, Catania, Italy*

Keywords: Image-Based Localization, Similarity Search, Metric Learning, Simulated Data.

Abstract: Current methods for Image-Based Localization (IBL) require the collection and labelling of large datasets of images or videos for training purpose. Such procedure generally requires a significant effort, involving the use of dedicated sensors or the employment of structure from motion techniques. To overcome the difficulties of acquiring a dataset suitable to train models to study IBL, we propose a tool to generate simulated egocentric data starting from 3D models of real indoor environments. The generated data are automatically associated to the 3D camera pose information to be exploited during training, hence avoiding the need to perform "manual" labelling. To asses the effectiveness of the proposed tool, we generate and release a huge dataset of egocentric images using a 3D model of a real environment belonging to the S3DIS dataset. We also perform a benchmark study for 3 Degrees of Freedom (3DoF) indoor localization by considering an IBL pipeline based on image-retrieval and a triplet network. Results highlight that the generated dataset is useful to study the IBL problem and allows to perform experiments with different settings in a simple way.

1 INTRODUCTION

Estimating the position of a camera in an indoor environment is useful in different application domains related to computer vision and robotics. More in general, any egocentric vision application can exploit the camera pose information as a prior to solve other tasks (e.g. object recognition, augmented reality, etc.). Previous work investigated different approaches to Image-Based Localization for Simultaneous Localization And Mapping (SLAM) (Engel et al., 2015), visual structure from motion (Hartley and Zisserman, 2003) and augmented reality (Schöps et al., 2014). The approaches based on SLAM predict the camera pose directly aligning images with a pose graph. These approaches use the camera motion to map the environment keeping history of key frames assuming an unknown environment. Therefore, localization is memoryless and the environment needs to be mapped in order to be able to localize the camera. Differently, IBL aims to localize the camera position in previously known indoor environments for which labeled images are available. According to the literature, camera localization can be performed with different levels of precision, e.g., to recognize specific environments (Ragusa et al., 2018; Torralba et al., 2003; Pentland et al., 1998) or to retrieve the pose of the camera (Kendall et al., 2015; Arandjelovic et al., 2016). However, in order to train localization systems, a labelled dataset containing a large number of images of the environment is generally needed.

In recent years, the amount of data required for computer vision applications has increased significantly. Often, the raw data have to be labelled in order to train supervised learning algorithms, thus increasing the costs of the system to be deployed. In particular, to tackle indoor localization, we need to collect a large scale dataset as varied as possible. This is usually done by collecting real videos while following different paths inside an environment. To include variability during the acquisition, it is important that the operators look around and observe different points of interest inside the different rooms, walk down a corridor in both directions, as well as enter and leave the rooms of interest.

The aforementioned operational constraints to be considered during the acquisition of the training dataset are the main drawback of IBL systems and limit the size and quality of the current datasets. For instance, the datasets for indoor IBL proposed in (Walch et al., 2017; Glocker et al., 2013) have been acquired with dedicated sensor (e.g. laser scanners, Kinect) and contain only few thousands of labelled images. To address this issue, we propose a tool which can be used to simulate different users navigating a 3D model of a real environment in order to col-

Orlando, S., Furnari, A., Battiato, S. and Farinella, G.
Image Based Localization with Simulated Egocentric Navigations.
DOI: 10.5220/0007356503050312
In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019)*, pages 305-312
ISBN: 978-989-758-354-4

lect videos with an egocentric point of view. This tool can be used to automatically generate datasets of video sequences labelled with the 6 Degrees of Freedom (6DoF) pose of the camera. The tool has been developed as a plug-in for Unity 3D. Using the proposed tool, we collected a huge dataset of labelled images for IBL starting from a 3D model of a real building proposed in (Armeni et al., 2016). Being able to generate simulated data is useful for different aspects:

- It allows to obtain huge amounts of data without increasing the cost of acquisition and labelling;
- It allows to easily introduce a controlled amount of variability in the data, which would be difficult to introduce otherwise;

To demonstrate the effectiveness of the proposed methodology to study the problem of IBL, we propose the benchmark of a localization techniques based on metric learning and image-retrieval on the generated dataset. It should be noted that the photorealism of the dataset collected using the proposed tool depends on the quality of the considered 3D model. Nevertheless, the acquired data can be useful to study the problem of IBL and compare different techniques.

In sum, the contributions of this paper are the followings:

1. we propose and release a tool to create synthetic datasets starting from 3D models of real environments to study egocentric IBL;
2. we collect and publicly release a large dataset for IBL starting from a 3D model of a real building proposed in (Armeni et al., 2016);
3. we perform a benchmark study of IBL based on metric learning and image-retrieval on the generated dataset.

The remainder of the paper is organized as follows: Section 2 presents the related works; Section 3 describes in details the tool to produce simulated data from a 3D model of an indoor environment; Section 4 describes the dataset acquired using the proposed tool, whereas Section 5 reports the proposed IBL benchmark study. Section 6 concludes the paper and gives insights for future works.

2 RELATED WORK

Image-Based Localization consists in determining the location from which a given image has been acquired in a known environment. When addressed with machine learning, this task can be tackled as a classification problem or as a camera pose estimation task.

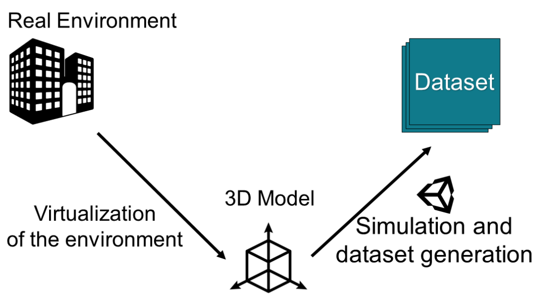
Real Environment
Dataset
Virtualization of the environment
3D Model
Simulation and dataset generation

Figure 1: Pipeline for indoor simulation and generation of the dataset.

In the case of classification, the environment is divided into a fixed number of cells (N) of arbitrary size. Given a set of images I, the classification approach aims to model a function $f : I \to \mathbb{R}^N$ which assigns each input image $x_i \in I$ to one of the N cells. The function f computes a class vector score $s \in \mathbb{R}^N$ which corresponds to a probability distribution over the N possible cells of the environment. The most probable location $\hat{y}$ is then associated to the query image considering the cell with the highest score:

$$\hat{y} = \arg\max_j (s_j). \tag{1}$$

where s_j is the j-th component of the class score vector $s = f(x_i)$. This model can be useful for room-level localization where each room is considered as one of the N cells in order to predict the room in which a query image has been taken. Examples of IBL based on classification are reported in (Pentland et al., 1998; Torralba et al., 2003; Weyand et al., 2016; Santarcangelo et al., 2018; Ragusa et al., 2018).

The second class of approaches aims to model a function f to estimate the pose $p_i = f(x_i)$ of an image $x_i \in I$. The pose p_i is usually related to the 6DoF values determining the location and orientation of the camera in a 3D space. In this case, the labels $y = \{p, q\}$ related to an image represent the position $p \in \mathbb{R}^3$ and the orientation of the camera $q \in \mathbb{R}^4$ (expressed as a quaternion). Given the inferred location label $\hat{y} = \{\hat{p}, \hat{q}\}$ and the ground truth pose y, we can define a loss function for training a localization model as follows (Kendall et al., 2015):

$$J(\theta) = d_p(p, \hat{p}) + \beta d_q(q, \hat{q}) \tag{2}$$

where d_p and d_q denote distance functions with respect to position and orientation, whereas β regulates the trade-off between position and orientation error. Example of this second class of IBL approaches are reported in (Kendall et al., 2015; Arandjelovic et al., 2016).

3 SIMULATED NAVIGATIONS

We propose a general procedure and a tool for Unity 3D which can be used to generate synthetic datasets suitable to study IBL. The overall pipeline of our approach is depicted in Figure 1 and comprises two main steps:

1. Virtualization of the environment;
2. Simulation and generation of the dataset;

3.1 Virtualization of the Environment

The first step requires the virtualization of a real 3D environment to obtain a 3D model suitable for navigation. This procedure can be performed using commercial products such as Matterport 3D[1]. In our experiments, we consider a 3D model of a real building (Area 3) which is part of the S3DIS dataset (Armeni et al., 2016). The considered model has been acquired using the Matterport scanner. This scanner provides a cloud service to process the scans of the environment and returns as output both the 3D model and the point clouds (including the related textures).

3.2 Simulation and Dataset Generation

To simulate navigations inside the 3D model, our tool uses the AIModule of the Unity Engine. This module allows to mark the floor of the 3D mesh as walkable and provides a path finding algorithm to reach 3D points (x,y,z) in the model space. The Unity engine uses a left-handed coordinate system, which implies that the floor of the building lies on the XZ plane.

We define a set of reachable target points $P \in \mathbb{R}^2$ (we consider $y = 1$) arranged in a regular grid on the floor of the 3D model, as shown in Figure 2. The target points are placed at a step of $1m$ both horizontally and vertically. In this way, we obtain a grid of target points to be used during the simulated egocentric navigations.

The tool allows to discard all the points that cannot be reached because they fall outside of the walkable area (see Figure 2). The target points to reach during a navigation are chosen randomly from a uniform distribution in order to create N paths π_j containing T target each. We let each path π_j terminate with the same target as the starting one to obtain closed loops:

$$\begin{gathered} \pi_j = (t_0, ..., t_{T-1}, t_T) : t_0 = t_T \\ for\ j = 0, ..., N \end{gathered} \tag{3}$$

[1]Matterport: https://matterport.com/

Algorithm 1: NavigationProcedure.

```
 1: $N$: number of paths
 2: $T$: number of reachable targets for each path
 3:
 4: currentPath=currentTarget=0;
 5: while currentPath<N do
 6:   if targetReached() then
 7:     currentTarget+=1;
 8:     nextTarget = currentTarget;
 9:     if currentTarget==T then
10:       currentPath+=1;
11:       currentTarget = 0;
12:     end if
13:     setNextDestination(nextTarget);
14:   end if
15: end while
```

Once a path π_j is defined, the AIModule of Unity 3D is used to let a virtual agent navigate the environment, sequentially reaching each target belonging to the path π_j.

In order to simulate the movement of the head of the virtual egocentric agent, we specified 7 different animations for the camera rotation:

1. no rotation: no animation is applied to the camera;
2. clockwise rotation: the camera rotates clockwise;
3. triangle up: the camera first looks at the top left corner, then at the top right corner;
4. triangle down: similar to triangle up, but the camera first looks at the bottom left corner, then at the bottom right corner;
5. yaw: the camera rotates from center to left/right along the Y-axis;
6. pitch: the camera looks up and down along the X-axis;
7. roll: a little tilt is applied to the Z-axis of the camera.

These animations are chosen randomly with a uniform distribution during the simulated navigations. All the aforementioned animations have the same duration of 4 seconds with the exception of the clockwise animation which have been set to 9 seconds. The y coordinates of the camera can be adjusted to simulate agents with different heights. The pseudo code of the navigation procedure is shown in Algorithm 1. The method (SetNextDestination(nextTarget)) sets the next point to be reached using the AIModule in line “14”. During the navigation, random camera animations are applied as discussed above.

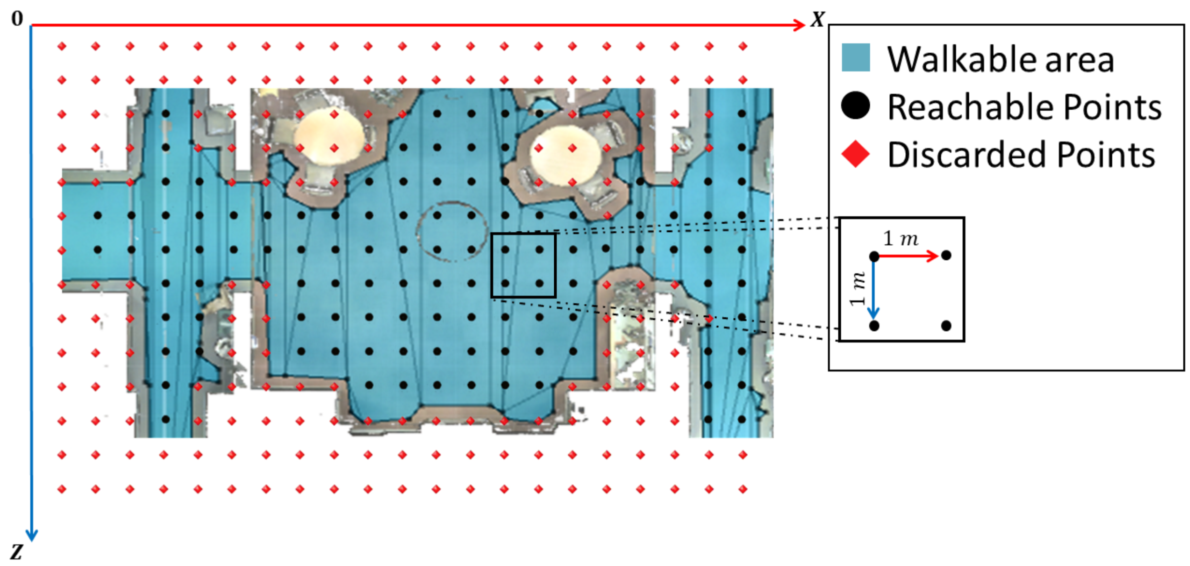

Figure 2: A section of the mesh used to acquire our dataset. The walkable area is marked in blue. Black dots indicate the reachable points, while the red ones are discarded because they fall outside the walkable area.

We attach the FrameCapturer[2] component to the Unity camera. This provides a method to acquire different channels from the Unity camera including: RGB, depth map and normal map. We set the framerate of the FrameCapturer component to $30fps$ (frames per seconds). Our tool allows to save a label with the following information for each frame:

```
- path number
- camera position (x, y, z)
- camera rotation quaternion (w, p, q, r)
- next target to reach
```

where, *path number* is the ID of the current path, *camera position* and *camera orientation* represent respectively the coordinates and the rotation quaternion of the camera, whereas *next target to reach* is the ID of the next target to be reached. The developed tool and the related supplementary material containing details on how to use it are available at the following link: http://iplab.dmi.unict.it/SimulatedEgocentricNavigations/tool.html.

4 DATASET

We employed our tool to collect the Dataset exploited in this paper by using the “Area 3” 3D model proposed in (Armeni et al., 2016). We generated the dataset simulating the navigation of 3 different agents each one with a different height among the following: $[1.50m; 1.60m; 1.70m]$. During the simulation, each agent followed 30 different random paths each, composed by $T = 21$ targets.

[2]FrameCapturer:
https://github.com/unity3d-jp/FrameCapturer

Table 1: Amount of data generated to perform the experiments.

	Dataset generated from “Area 3”			
Agent height	**1.50 m**	**1.60 m**	**1.70 m**	**Total Images**
# of Frames	301,757	296,164	288,902	886,823

For each frame, we stored the RGB buffer and the depth map of the camera. Table 1 reports the amount of images acquired during the generation of the dataset. Please note that the proposed tool allows to easily acquire huge amounts of data which can be useful, for instance, to study how much data we need to tackle the IBL problem. Each frame has a resolution of $1280x720$ pixels. Each path is composed of $9,853$ frames on average. The generation of all the 90 paths required 3 days on a notebook with an Intel i7 processor and an Nvidia GTX960M GPU. Each path has an average duration of 5.47 minutes for a total of about 8.21 hours for the whole dataset.

After the generation, we preprocessed the data as follows:

- each image has been resized to $455x256$ pixels;
- each position label has been converted from 6DoF to 3DoF.

The images have been resized to be subsequently used within a learning method, whereas the pose labels have been converted to perform our benchmark considering 3DoF camera localization following the work in (Spera et al., 2018). In this regard, we localize the camera with respect to the X and Z axes according to the left handed system of Unity. The 3^{rd} DoF is given

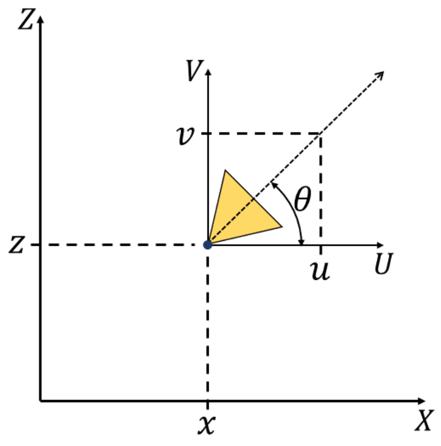

Figure 3: Detail of 3DoF pose representation. We described the camera pose of each frame with the position (x,z) of the camera and the orientation vector (u,v) which represents the angle θ of rotation of the camera about the Y-axis.

by the rotation of the camera around the Y-axis. The camera orientation is converted from quaternions to Euler angles then the rotation along the Y- axis is considered, whereas the rotations about the other axes are discarded. Orientation is represented as a unit vector as shown in Figure 3. The unit vector (u,v) is computed as follows:

$$u = \cos\theta \;\; ; \; v = \sin\theta \tag{4}$$

where θ is the rotation of the camera along the Y-axis. We applied this transformation to each camera position label of our dataset obtaining the new labels for localization in the 3DoF format $[(x,z);(u,v)]$. The dataset is available at the following URL: http://iplab.dmi.unict.it/SimulatedEgocentricNavigations/.

5 BENCHMARK STUDY

In this section, we report a benchmark study performed on the dataset generated using the proposed tool to study the IBL problem. We cast IBL as an image retrieval problem and use a Triplet Network (Hoffer and Ailon, 2015) to learn a suitable representation space for images. We follow the approach in (Spera et al., 2018) and consider a Triplet Network with an Inception V3 backbone (Szegedy et al., 2016). The Triplet Network follows the model proposed in (Hoffer and Ailon, 2015) and uses a Margin Ranking Loss for training. During test, we perform 3DoF localization by considering an image-retrieval approach which assigns a 3DoF label to a new query image by performing a K-Nearest Neighbor search (with $K = 1$) on a set of labelled images mapped to the learned representation space. The network architecture is trained using triplets, each comprising three images: the anchor frame I, an image I^+ similar to the anchor frame and a dissimilar image I^-. The network performs the forward process evaluating the embedded representation $Net(I)$ for each of the three images and returns as output the two L_2 distances defined as follows:

$$\begin{aligned} d_+ &= ||Net(I) - Net(I^+)||_2 \\ d_- &= ||Net(I) - Net(I^-)||_2 \end{aligned} \tag{5}$$

For training, we employ a margin ranking loss which encourages d_- to be grater then d_+. The loss is defined as follows:

$$loss(d_+, d_-) = max(0, d_+ - d_- + m) \tag{6}$$

where m is the margin value of this function (we set $m = 0.2$ in our experiments).

To measure the distance between two images, we consider two distances, one for the position d_{pos}(Euclidean Distance) and one for the orientation d_{or}. The latter is defined as:

$$d_{or} = \arccos(u_0 u_1 + v_0 v_1). \tag{7}$$

Hence, to generate the training triplets $(I,\, I^+,\, I^-)$ for each anchor frame I, we randomly sampled two images to have one similar image I^+ and one dissimilar image I^- using the following rule:

$$\begin{aligned} I^+ &: d_{pos}(I,\, I^+) \leq Th_{xz} \land d_{or}(I,\, I^+) \leq Th_\theta \\ I^- &: d_{pos}(I,\, I^-) > Th_{xz} \lor d_{or}(I,\, I^-) > Th_\theta \end{aligned} \tag{8}$$

where Th_{xz} and Th_θ are respectively thresholds used for position distance (d_{pos}) and orientation distance (d_{or}).

We investigated the use of the values $[60°;\; 45°;\; 30°]$ as thresholds Th_θ for d_{or}, whereas we considered the values $[200cm;\; 100cm;\; 75cm;\; 50cm;\; 25cm]$ for the threshold Th_{xz} related to d_{pos}. Figure 4 shows some examples of triplets sampled from our dataset.

To perform the benchmark study, we split the dataset into three parts: *Training set, Validation set and Test set*. More specifically:

- the *Training set* contains the images of the three agents from $Path_0$ to $Path_{17}$.
- the *Validation set* contains the images of the three agents from $Path_{18}$ to $Path_{23}$.
- the *Test set* contains last six paths of the agents, from $Path_{24}$ to $Path_{29}$.

To study the influence of the training set dimension, we randomly sampled 4 subset of triplets with different size from the *Training set* for the learning procedure of the Triplet Network. We started with a *Training set* of 10,000 images. A fixed number of

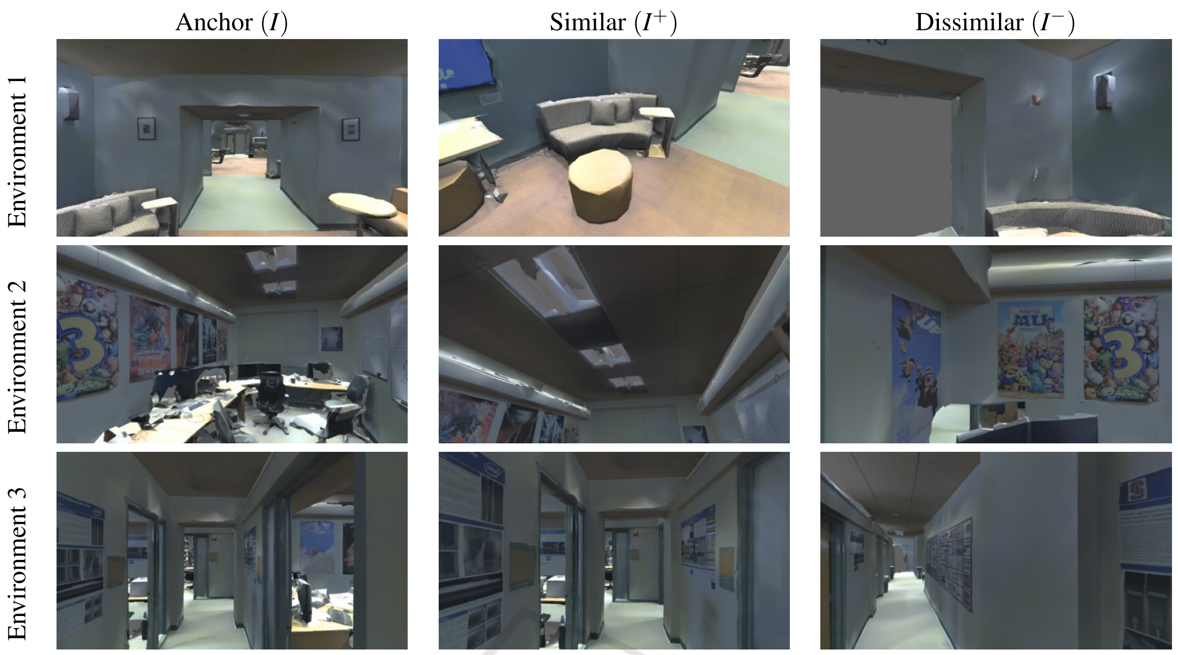

Figure 4: Three examples of triplets sampled from the proposed dataset. The first column (Anchor) shows the pivot image of each Triplet. The second column (Similar) satisfies the first condition of Eq.(8), whereas the third column (Dissimilar) shows an images which satisfies the second condition in Eq.(8).

Table 2: Results related to the selection of the two thresholds $Th_{xz} and\ Th_\theta$ when $10,000$ training samples are considered to build the embedding space with the triplet network. For each considered value of Th_θ, we marked in bold the best result.

Th$_\theta$ / **Th$_{xz}$**	**60°**	**45°**	**30°**
2m	$0.73m \pm 1.79m; 11.93° \pm 22.06°$	$0.71m \pm 1.73m; 12.17° \pm 22.47°$	$0.71m \pm 1.82m; 11.16° \pm 21.72°$
1m	$0.68m \pm 1.81m; 11.52° \pm 20.54°$	$0.72m \pm 1.88m; 11.57° \pm 21.93°$	$\mathbf{0.67m \pm 1.79m; 10.34° \pm 19.88°}$
0.75m	$\mathbf{0.64m \pm 1.68m; 11.42° \pm 20.97°}$	$0.72m \pm 1.94m; 11.59° \pm 22.33°$	$0.67m \pm 1.72m; 10.89° \pm 21.21°$
0.5m	$0.74m \pm 2.09m; 11.59° \pm 21.62°$	$\mathbf{0.69m \pm 1.86m; 11.61° \pm 21.98°}$	$0.71m \pm 1.88m; 11.19° \pm 21.98°$
0.25m	$0.64m \pm 1.71m; 11.41° \pm 21.62°$	$0.71m \pm 1.87m; 11.65° \pm 22.41°$	$0.76m \pm 2,06m; 11.68° \pm 22.65°$

$5,000$ images have been considered for the *Validation set* and $10,000$ images for the *Test set*. The triplets used for learning the representation model are generated at run time at the beginning of each epoch during training, whereas the set of triplets for *Validation* have been generated just once and have been used for all training procedures. The *Test* set is used for testing purposes, so we don't need to generate triplets for this subset. After learning the embedding network "*Net*", the model is used to represent each training and test image. Localization is performed through a K-Nearest Neighbor search ($K = 1$). We train each triplet model for 50 *epochs* with a learning rate of 0.001 using a SGD optimizer with momentum equal to 0.9. We then select the model corresponding to the epoch which achieves the best validation performance.

We performed experiments aimed at assessing the influence of the selection of the thresholds used to generate the triples for training. We then considered the best combinations of thresholds and performed a study on the number of images exploited during training by considering the training sets of dimensions $20,000$, $30,000$ and $40,000$ samples to understand the impact of the training size on performances.

5.1 Results

Table 2 reports the results related to the different position/orientation threshold values involved in the generation of the triplets when $10,000$ samples are considered for the training of the triplet network. The results are reported using the notation $\bar{y} \pm \sigma$, where $\bar{y}$ is the mean error and σ is the error standard deviation. The best results for each Th_θ threshold are marked in bold. As pointed out by the results, we obtained an average position error of $0.64m \pm 1.68m$ and an average orientation error of $11.42° \pm 20.97°$ by setting $Th_{xz} = 0.75m$ and $Th_\theta = 60°$. However, the results in Table 2, seam to be similar for different choices of the considered thresholds, which suggests

Table 3: Results related to different training set dimensionality during training.

Th_θ	Th_{xz}	Training Samples	Position Error	Orientation Error
60°	0.75m	**10000**	$0.64m \pm 1.68m$	$11.42^\circ \pm 20.97^\circ$
		20000	$0.50m \pm 1.37m$	$8.83^\circ \pm 16.52^\circ$
		30000	$0.46m \pm 1.21m$	$8.71^\circ \pm 17.45^\circ$
		40000	$\mathbf{0.42m \pm 1.21m}$	$\mathbf{7.85^\circ \pm 15.35^\circ}$
45°	0.5m	**10000**	$0.69m \pm 1.86m$	$11.61^\circ \pm 21.98^\circ$
		20000	$0.47m \pm 1.27m$	$8.58^\circ \pm 17.07^\circ$
		30000	$0.44m \pm 1.19m$	$8.05^\circ \pm 15.49^\circ$
		40000	$\mathbf{0.40m \pm 1.13m}$	$\mathbf{7.72^\circ \pm 15.65^\circ}$
30°	1m	**10000**	$0.67m \pm 1.79m$	$10.34^\circ \pm 19.88^\circ$
		20000	$0.56m \pm 1.68m$	$8.86^\circ \pm 17.67^\circ$
		30000	$0.48m \pm 1.37m$	$8.01^\circ \pm 16.61^\circ$
		40000	$\mathbf{0.43m \pm 1.25m}$	$\mathbf{7.49^\circ \pm 16.31^\circ}$

Table 4: The table shows the percentage of test images with error below of $0.5m$ and below of 30° for varying the numbers of triplets used during the training of the network. The last row reports the average time spent during the training phase.

	Training Samples			
Thresholds	**10000**	**20000**	**30000**	**40000**
$0.75m$, 60°	72.34%	78.92%	80.22%	82.20%
$0.5m$, 45°	72.26%	79.56%	81.03%	83.26%
$1m$, 30°	71.75%	76.70%	80.34%	82.47%
Avg Training time (in hours)	8.59	16.45	23.98	33.85

that the choice of the pair of thresholds Th_{xz} and Th_θ does not appear critical since the method is not very sensible to them. This might be due to the fact that the thresholds are used to learn an embedding space and not directly for camera pose estimation. Therefore, different thresholds might lead to similar triplet training objectives, and hence similar embedding space.

For each of the optimal threshold pairs highlighted in Table 2, we also performed experiments to assess the influence of the training set dimensionality on performance. Specifically, we trained the triplet network by using: $20,000, 30,000$, and $40,000$ images. The results of these experiments are reported in Table 3. We observe that the localization accuracy increases by increasing the dimensionality of the training set. This encourages the use of the proposed tool to generate a huge amount of labelled data with minimum effort. The best results ($\simeq 0.40m$, $\simeq 7.72^\circ$) are obtained exploiting $40,000$ samples during training. To better highlight the results, Table 4 reports the percentages of images correctly localized with respect to an acceptance error of $0.5m$ for position and 30° for orientation (i.e. the proportion of test images localized with an error below of $0.5m$ for position and below of 30° for orientation). In this case, the two considered thresholds on the error represent a tolerance admittable for practical applications based on localization. Results point out that more than 70% of test samples are correctly classified when the training set containing $10,000$ images, and the percentage increases by increasing the number of training samples. Table 4 also reports the average training times obtained using an Nvidia GeForce GTX 1080 with 12GB of RAM. The time spent for training is approximately linear with respect to the dimension of the training set.

6 CONCLUSIONS

We proposed a tool to create synthetic datasets to study the problem of Image Based Localization generating Simulated Egocentric Navigations. The images generated during the navigations are automatically labelled with 6DoF camera pose information. Using this tool, we considerably reduce the cost of acquisition and labelling time.To assess the usefulness of the proposed tool we performed a study of localization methods based on image retrieval and a Triplet Network to learn a suitable image representation. The best obtained results shown a mean position error of $0.40m$ and a mean orientation error of 7.72°. The results can be useful to understand the amounts of data required to tackle the problem and to fine tune the thresholds used to generate triplets.

Future works can consider techniques which exploit the depth information to improve localization performances. Moreover, we will also investigate methods of domain adaptation to transfer the learned embedding space to the case of real data.

ACKNOWLEDGEMENT

This research is supported by DWORD - Xenia Progetti s.r.l. and Piano della Ricerca 2016-2018 linea di Intervento 2 of DMI, University of Catania.

REFERENCES

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). Netvlad: Cnn architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5297–5307.

Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., and Savarese, S. (2016). 3d semantic parsing of large-scale indoor spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1534–1543.

Engel, J., Stckler, J., and Cremers, D. (2015). Large-scale direct slam with stereo cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942.

Glocker, B., Izadi, S., Shotton, J., and Criminisi, A. (2013). Real-time rgb-d camera relocalization. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 173–179.

Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Hoffer, E. and Ailon, N. (2015). Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92.

Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *IEEE international conference on computer vision (ICCV)*, pages 2938–2946.

Pentland, A., Schiele, B., and Starner, T. (1998). Visual context awareness via wearable computing. In *International Symposium on Wearable Computing*, pages 50–57.

Ragusa, F., Furnari, A., Battiato, S., Signorello, G., and Farinella, G. M. (2018). Egocentric visitors localization in cultural sites. In *Journal on Computing and Cultural Heritage (JOCCH)*.

Santarcangelo, V., Farinella, G. M., Furnari, A., and Battiato, S. (2018). Market basket analysis from egocentric videos. In *Pattern Recognition Letters*, volume 112, pages 83–90.

Schöps, T., Engel, J., and Cremers, D. (2014). Semi-dense visual odometry for ar on a smartphone. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 145–150.

Spera, E., Furnari, A., Battiato, S., and Farinella, G. M. (2018). Egocentric shopping cart localization. In *International Conference on Pattern Recognition (ICPR)*, pages 2277–2282.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Torralba, A., Murphy, K. P., Freeman, W. T., Rubin, M. A., et al. (2003). Context-based vision system for place and object recognition. In *International Conference on Computer Vision (ICCV)*, volume 3, pages 273–280.

Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., and Cremers, D. (2017). Image-based localization using lstms for structured feature correlation. In *International Conference on Computer Vision (ICCV)*, pages 627–637.

Weyand, T., Kostrikov, I., and Philbin, J. (2016). Planet - photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55.