Expert Systems With Applications 213 (2023) 119006

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Three-dimensional reconstruction using SFM for actual pedestrian classification

Francisco Gomez-Donoso [*], Julio Castano-Amoros, Felix Escalona, Miguel Cazorla

Institute for Computer Research, University of Alicante, Alicante, Spain

ARTICLE INFO

Keywords:
Pedestrian recognition
Autonomous vehicles
Perception
Deep learning

ABSTRACT

In recent years, the popularity of intelligent and autonomous vehicles has grown notably. In fact, there already exist commercial models with a high degree of autonomy as regards self-driving capabilities. A key feature for this kind of vehicle is object detection, which is commonly performed in 2D space. This has some inherent issues as an object and the depiction of such an object would be classified as the actual object, which is inadequate since urban environments are full of billboards, printed adverts and posters that would likely make these systems fail. In order to overcome this problem, a 3D sensor could be leveraged, although this would make the platform more expensive, energy inefficient and computationally complex. Thus, we propose the use of structure from motion to reconstruct the three-dimensional information of the scene from a set of images, and merge the 2D and 3D data to differentiate actual objects from depictions. As expected, our approach is able to work with a regular color camera. No 3D sensors whatsoever are required. As the experiments confirm, our approach is able to distinguish between actual pedestrians and depictions of them more than 87% of times in synthetic and real-world tests in the worst scenarios, while the accuracy is of almost 98% in the best case.

1. Introduction

The future of transportation now undeniably lies in intelligent and autonomous vehicles. These kinds of vehicles are packed with a range of sensors that, in the case of intelligent vehicles, gives the driver access to much greater understanding of the road, so safety is increased and the driving task optimized. In the case of autonomous cars, they are able to drive themselves with no interaction from the driver. The sensors of choice commonly include color cameras and LiDAR (Laser Imaging Detection and Ranging). They provide visual and three-dimensional information on the environment at low level. The data provided by these sensors are often processed to extract knowledge. This is the case of segmentation systems that provide pixel-level classification or object detectors.

The methods that work on images already provide high accuracy, but have a significant flaw inherent to the input data, being that all the objects with similar features would be classified as the same object despite being different. For instance, if we feed images of an actual person and the depiction of a person in a billboard to an object detector, it would provide the same label for both, namely, person. This issue is critical when it comes to intelligent and self-driving cars as the urban environment is full of these depictions, in elements such as billboards, posters, bus shelters, and adverts. If we rely just on a 2D object detector, it will likely fail in these cases.

However, LiDAR could be used to classify three-dimensional data, thus resolving the issue. Nonetheless, these kind of classifiers are too computationally complex to be deployed on actual self-driving and intelligent car hardware.

Thus, in order to mix the advantages of three-dimensional data with the accuracy of the image-based object detectors, we propose a pipeline that builds a 3D representation of the environment from images. Concurrently, a 2D object detector is used to segment the persons. The result would include actual persons and depictions of persons. Once both representations are computed, the 3D data for each person in the scene is extracted. Finally, a planarity test is carried out to correctly classify the actual persons and discard the depictions.

The main contributions of our proposal are listed as following:

- We implemented a novel method that fuses 2D and 3D information to handle the problems that 2D object detectors have with object depictions in urban environments.
- We tested our system with both simulated and real data from custom and state of the art datasets that are established in the literature.

* Corresponding author.
E-mail addresses: fgomez@ua.es (F. Gomez-Donoso), julio.ca@ua.es (J. Castano-Amoros), felix.escalona@ua.es (F. Escalona), miguel.cazorla@ua.es (M. Cazorla).

https://doi.org/10.1016/j.eswa.2022.119006
Received 24 September 2021; Received in revised form 28 September 2022; Accepted 9 October 2022
Available online 17 October 2022

- We compared the performance of our system versus a state of the art 3D convolutional neural network, showing that our proposal outperforms this method for our task. Therefore, we prove that the combination of 2D and 3D information achieves better results than single 2D or 3D methods for this kind of task.

The rest of the work is structured as follows. First, we review the most important works in 2D and 3D object detection in Section 2. In Section 3, our proposal is then explained in depth. Section 4 includes information about the data setups we use to carry out the experiments described in Sections 5 and 6. Finally, conclusions, limitations and future works are explained in Section 7.

2. Related works

In this section, we review state-of-the-art methods for detecting objects in images and in three-dimensional data. As far as object detection in images is concerned, we can classify the methods into two families:

- One-stage detectors. These types of detectors perform classification and regression from the characteristics in a straightforward manner.
- Two-stage detectors. These types of detectors perform classification and regression in two phases: they generate the features in one phase, generate the bounding boxes in another phase, and these are then combined for classification.

Furthermore, this classification can be subdivided according to whether the detectors use anchor boxes (bounding boxes calculated offline), that is, whether they are anchor-based or anchor-free.

Among the one-stage detectors, YOLOv4 (You Only Look Once v4) (Bochkovskiy et al., 2020) brings together a number of techniques that have proven to work well independently. The authors' main goal is to obtain a fast detector that maintains competitive accuracy in real time. It consists of three parts. The first part of the detector, known as the backbone, is a convolutional neural network used only to extract features. In the second part, known as the neck, a technique is applied to mix and match these features. Finally, the third part, known as the head, performs the detection from the features.

The SSD (Single Shot Detector) detector (Liu et al., 2016) consists of a VGG16 (Visual Geometry Group, 16 blocks) (Simonyan & Zisserman, 2014) neural network, but replacing the dense layers with convolutional layers. In addition, 4 more convolution layers are added to further reduce the size of the features, thus obtaining features at different scales. Finally, each of the outputs of each convolution layer is combined in the last layer, where the detection is performed.

Finally, RetinaNet (Lin et al., 2017) is a detector that uses a variant of ResNet (He et al., 2016) as the basis of a feature pyramid network to extract features and bounding boxes at different scales. It then uses two subnetworks. The first subnetwork predicts the class of objects. The second subnetwork regresses the location of the objects.

Nonetheless, there are some of prominent methods that do not use predefined bounding boxes.

CornerNet (Law & Deng, 2018) is a method that directly regresses the upper left-hand corner and the lower right-hand corner of the bounding box independently from the features. One network is used for the upper left-hand corner and another network for the lower right-hand corner. Finally, vectors, known as embeddings, are learned to represent these corners. These embeddings are used to associate the corners of the same object.

CenterNet (Duan et al., 2019) is the continuation of CornerNet. This method adds information about the center of the object. CornerNet is used to obtain the corners and the bounding box, and checks whether there is an object inside each bounding box. To do this, the embeddings are used to verify whether there is a point belonging to the same class in the central region of the bounding box.

Among the two-stage anchor-based detectors, Faster R-CNN (Region-Convolutional Neural Network) (Ren et al., 2015) is a well-known method, part of the family of R-CNN (Girshick et al., 2014) and Fast R-CNN (Girshick, 2015), that first generates a feature map from the image. Using this map and with a RPN (Region Proposal Network), it generates candidate bounding boxes. The features, together with the bounding boxes, are combined in a final layer in which feature vectors are obtained for each bounding box. These vectors are used as input to a classifier.

R-FCN (Region-based Fully Convolutional Network) (Dai et al., 2016) is somewhat similar to Faster R-CNN in that it also uses an RPN from the feature map. However, in this method, the bounding boxes and the features are processed in a layer known as Position-Sensitive Score Maps. In this layer, matrices are produced that represent a type of percentage overlap between the features and the bounding boxes. From the matrices, the classes and locations are calculated.

Of the anchor-free methods, RepPoints (Yang et al., 2019) can be highlighted. Firstly, like virtually all detectors, features are obtained from the image by means of convolutional layers. It is made up of two parts: a first part that performs the regression of 9 points that are modified by a refinement process, generating a bounding box, and a second part which performs classification from the bounding boxes and the features.

In general, although 2D image-based methods achieve good results in terms of inference time and/or accuracy, none of them is able to solve the problem addressed in this paper, which consists of determining that a person on a billboard is, in reality, not a person.

Regarding the detection of 3D objects, we will distinguish the methods according to the type of environment in which this detection is applied, indoor and outdoor.

In an indoor environment, the objects to be detected are those that would be found in closed environments, such as tables, chairs, doors and sofas. In this respect, it is worth mentioning works that achieved good results in the challenges of the ModelNet (Wu et al., 2015a) and SUNRGB-D (Song et al., 2015) datasets.

The seminal work by Wu et al. (2015b) introduced the ModelNet dataset and a CDBN (Convolutional Deep Belief Network) to represent and learn 3D shapes as probability distributions of binary variables on volumetric voxel grids. The accuracy of this method was relatively low, but the work paved the way for future research.

Another approach to this problem is presented by Xu and Todorovic (2016), in which an optimal CNN (Convolutional Neural Network) hyperparameter beam search and architecture is proposed. This system models different network configurations as states, which are connected in a directed graph. The system traverses the graph, using a heuristic function that produces the next best state, an improved version of the architecture and hyperparameter set.

Frustum PointNet (Qi et al., 2018) obtains a set of candidate 2D bounding boxes using a 2D convolutional neural network. From each candidate 2D region, a point cloud is obtained, consisting of all the points that fall within the field of view in which the object is located. These point clouds enter a PointNet network that will predict the class of the object. It should be noted that this model has problems when there is more than one object of the same category within the field of view, as the 3D bounding box returned for each object is mixed with the other. In addition, it depends on 2D detection, and so if the 2D detector does not work properly (low light, occlusions, etc.), the 3D detection will have poor results.

3DSS (3-Dimensional Sliding Shapes) (Song & Xiao, 2016) uses the same candidate elicitation technique, but in three dimensions using a RPN. From the bounding boxes, it obtains its projections in 2D and trains a hybrid convolutional neural network so that the bounding boxes serve as input to a 3D Convolutional Neural Network and the projections to a 2D Convolutional Neural Network. RPN has problems with flat objects, such as TVs or monitors, which affects detection because if a good 3D bounding box is not obtained, detection will not work well. It also fails with objects whose category has great variance, e.g. the box category.

An important step forward was made by Sedaghat et al. (2016) who introduced object orientation prediction, in addition to the class label itself, to increase classification accuracy. Their ORION network is a 3DCNN (3-Dimensional Convolutional Neural Network) that produces class labels and orientations as outputs and uses both to contribute to training. By adding orientation estimation as an auxiliary task during training, they were able to learn orientation invariance and increase accuracy.

VRN (Voxception-ResNet) Ensemble (Brock et al., 2016) comprises an encoder network, the latent layer and a decoder network. The encoder network consists of 4 convolutional layers and a fully connected layer, followed by a linear projection from the fully connected layer to the latent layer. The decoder network has an identical architecture, but inverted, and its weights are not linked to those of the encoder. The output of each element in the final layer can be interpreted as the prediction that a voxel is present at a given location.

Finally, in order to approximate the results to real life scenarios, *Par3DNet* (Gomez-Donoso et al., 2020) used a 3DCNN to perform object recognition over tridimensional partial views of the objects, making an in-depth analysis of the easiest and hardest views to classify an object.

Most of these approaches work with a voxelised representation of the point clouds and predict the categories from the voxels by means of a 3D deep convolutional neural network or similar. Although, for example, VRN Ensemble performs best in the challenges, the voxelisation operations it uses are too computationally complex for real-time execution.

For the outdoor environment, the KITTI dataset (Geiger et al., 2013) is used as a reference. KITTI is a state-of-the-art dataset of urban objects. The main classes typically used to test the models are pedestrians, cyclists and cars.

As this dataset is the most widely used for urban object detection, we can find many proposals, such as MV3D (Multi-View 3D object detection network) (Chen et al., 2017) and Vote3Deep (Engelcke et al., 2017).

MV3D is a pipeline, the input of which is 3D data from different perspectives (bird's eye view and frontal view) and the RGB image. It is composed of two phases, a first phase consisting of a 3D Proposal Network that will use the bird's eye view to generate candidate 3D bounding boxes. The second phase is a Region-based Fusion Network. The bounding boxes are projected in both views and in the image, with the regions corresponding to the bounding boxes being obtained. Finally, features are extracted from the three views, fused, and sent as input to a classifier. Although it appears to be highly accurate in generating the bounding boxes, it has problems with small objects, such as cyclists and pedestrians, and also has difficulties with scenes where there are many objects moving vertically.

Vote3Deep, on the other hand, only works with point clouds represented as a 3D grid. This model uses a 3DCNN together with a voting algorithm and L1 regularization. The voting algorithm consists of performing the convolutions, but only on the elements that are non-zero and obtaining the same result as a normal convolution, but with greater efficiency. It is fast, although the prediction speed could be increased by developing a GPU (Graphics Processing Unit) implementation of the voting algorithm. Moreover, the authors present two models, one for car detection and one for pedestrian and cyclist detection, because they need to change the kernel size in the last layer due to the size of cars compared to pedestrians or cyclists. Therefore, if they were to present a single model for all 3 classes, the results would be affected.

In summary, three-dimensional object detectors could solve the proposed problem because they learn from 3D information. However, such methods do not yet have sufficiently high success rates. In addition, a pure three-dimensional approach requires too much computational power to be deployed in intelligent vehicles.

Regarding methods intended exclusively for pedestrian recognition, many use only color images. In Benenson et al. (2014), Dollár et al. (2009), Dollar et al. (2011), we can find a review with the most significant approaches in the early years using traditional classifiers over images. Most of these are based on handcrafted features and the accuracy is too poor to be used in real life scenarios.

Subsequently, with the advent of deep learning, researchers moved into CNN architectures to perform this classification. In Brunetti et al. (2018), the authors present an exhaustive review of computer vision and deep learning techniques for pedestrian detection and tracking. The most common approach is presented in works such as Lan et al. (2018), where a single stage detector (YOLO) is used to predict a bounding box and its category confidence. The authors present a modification with pass-through layers and additional connections to improve the quality of the features and their accuracy. In the case of Liu et al. (2019a), the authors propose a feature detector (Center and Scale Prediction) that is able to search for central points where there are pedestrians, and obtain the scale of the detection with a Fully Convolution Network (FCN). By performing the search on the center of the pedestrians instead of on boxes in the image, it is more robust to occlusions. However, systems that work with color images present the problem we aim to solve in this paper.

Finally, some works use a LiDAR to perform classification of three-dimensional data. In Liu et al. (2019b), the authors present a pedestrian detection system using template matching. As a first step, they segment the ground and filter the points in grid cells whose height difference does not correspond to humans. They then perform a Kernel Density Estimation (KDE) clustering to extract candidate pedestrians. Finally, they project the candidates in a range image, extract features from the contour and compare the cosine similarity with the template of an actual pedestrian. This kind of approach suffers when point clouds are sparse or occluded. Furthermore, in Wu et al. (2021), the authors propose a multi-LiDAR system where they fuse the classification scores of the left-hand and right-hand LiDARs to improve the accuracy. This system is composed of three stages: object proposal, fine classification and score fusion. The object proposal generates object candidates, using ground segmentation and point clustering. These are therefore passed to an object classifier, a Support Vector Machine (SVM). In the fine classification step, a set of features (153 dimensions) are generated, representing the number of points, covariance matrix and rotational projection statistics. The system provides the pedestrian likelihood estimation for every cluster. Finally, the results are fused in the score fusion stage. However, these systems are computationally expensive and do not provide the speed of execution required by a real-time system.

3. Reconstruction and classification pipeline

In this section, we provide an in-depth description of our proposed reconstruction and classification pipeline. We first provide a general description, and each subsection focuses then on one stage of the pipeline. As mentioned, this work presents a new method that mixes 2D and 3D data in order to discriminate actual persons from elements that look like persons, but are not. Our method draws on the idea that actual persons are not flat, but have three-dimensional volume, while elements that show representations of personas, such as billboards, posters, adverts, bus shelters and so on, are flat. The input of our method is a sequence of images and the output is the classification in either person or background. To achieve this goal, we propose the pipeline displayed in Fig. 1. It is worth mentioning that this method is intended to be applied to intelligent vehicles in urban environments.

First, a sequence of images is captured as the vehicle moves with its built-in regular color camera. The images are then fed to two different methods running simultaneously. On one hand, the sequence is processed by a Structure from Motion (SfM) algorithm that provides

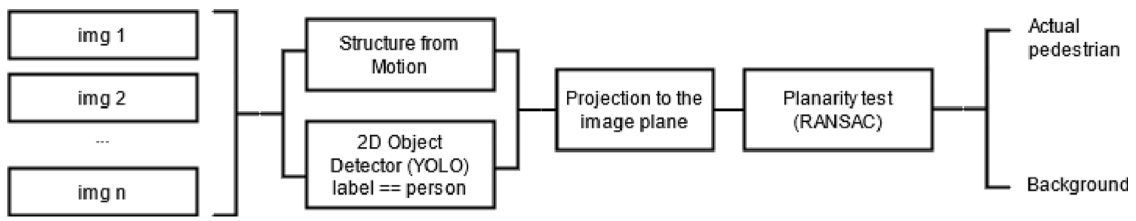


Fig. 1. Our proposal takes a number of images to perform a three-dimensional reconstruction of the scene using SfM, thus retrieving a point cloud. Concurrently, the persons are detected using a 2D object detector. The 3D points are projected into the image plane, and the corresponding points of the persons are extracted. A planarity test is carried out to finally classify each detected person as an actual person or background.

a three-dimensional representation of the environment. This representation is actually a point cloud. On the other hand, the last image of the sequence is fed to a 2D object detector in order to detect the persons that are present in the scene. As a result of this step, a collection of bounding boxes are computed, each corresponding to an object that looks like a person. Once we obtain the point cloud of the scene and the position of all the elements in the image that look like persons, the next stage consists of projecting the 3D points to the last image of the sequence. This can be done since the SfM algorithm also generates the poses of the camera for each image in the sequence. We extract the points inside the bounding boxes of each element retrieved by the 2D object detector. As a result, a collection of point clouds is obtained. Finally, for each point cloud a planarity test is carried out. If the object is flat, it is classified as background. Otherwise, it is classified as a person.

The following subsections further delve into each part of the proposed pipeline.

3.1. Three-dimensional reconstruction

The first stage of the proposed pipeline comprises a three-dimensional reconstruction, namely, we obtain a point cloud of the scene. To do so, we leverage an SfM method (Özyesil et al., 2017). Specifically, we use the OpenSfM (Mapillary, 2021) implementation. The SfM method takes a sequence of images as input. A feature detection, matching and filtering process is then run. Finally, a bundle adjustment algorithm is also executed, in order for the best alignment to be obtained. As a result of this method, a point cloud and the estimated poses of the camera are computed, as shown in Fig. 2.

In this case, the input sequence is composed of 8 images and the feature detection of choice is HAHOG. The matching is carried out by a K-Means (Jin & Han, 2010) algorithm.

3.2. 2D object detection

A 2D object detector is also run in the first stage of our proposal. We use YOLOv3 (Redmon & Farhadi, 2018) to perform this task. This method is a deep-learning-based architecture that takes an image as an input and provides the location of every detected object in the image, that is, it computes their bounding boxes.

We chose this approach because it is the best performer in terms of accuracy whilst keeping the computation cost at bay. We adopted the model as provided by the original authors. Finally, it is worth noting that this architecture is able to detect a range of different objects, but we only consider those labeled as persons. As this algorithm leverages visual features, every element in the scene that looks like a person would be classified as one, including objects that are not actually persons, as seen in Fig. 3.

3.3. Point cloud instance segmentation

Once the point cloud of the scene and the bounding boxes of the elements labeled as persons are obtained, the next step is to segment the 3D points that belong to each of them. To do so, we project the point cloud to the current image. This process is straightforward as we already computed the camera pose $[R \mid t]$ in the three-dimensional reconstruction step explained in Section 3.1, and the intrinsic parameters are also known. Thus, we can use Eq. (1) to project each 3D point $[X \;\; Y \;\; Z]$ in the point cloud to the image plane.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} max(w,h) & 0 & \frac{w-1}{2} \\ 0 & max(w,h) & \frac{h-1}{2} \\ 0 & 0 & 1 \end{bmatrix} [R \mid t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

The points $[u \;\; v]$ that are inside of each bounding box are then collected so that the corresponding $[X \;\; Y \;\; Z]$ of each element labeled as object is obtained, as shown in Fig. 4. As a result of this step, a point cloud is returned for each object labeled as a person by the 2D object detector.

As the bounding boxes are not fitted to the shape of the persons, but, rather, they are rectangular, some background and traces of the floor are also included. Thus, the 3D points of these undesired artifacts must be filtered so that the point clouds only depict the elements of interest. Our filtering process is thus as follows. First, we compute the mean of the whole point cloud and a Z filtering is performed, removing the points that are beyond the mean for a given value f_1. This is done to remove the background points. The mean of the remaining points is then computed and another Z-filtering of the mean $\pm$ a new fixed value f_2 is performed, so only the points corresponding to the subject of interest are kept. f_1 and f_2 are set to 0.5 and 1.5 in the experiments.

3.4. Planarity test

As mentioned, in order to tell actual persons apart from persons depicted in bus shelters, billboards, adverts posters or other elements that look like persons but are actually not, we can take advantage of their three-dimensional data. The former have volume in the three dimensions whilst the latter are completely flat. Thus, in order to detect whether the point cloud extracted of an object is flat or not, we implemented a planarity test.

The planarity test taps Random Sample Consensus (RANSAC) (Fischler & Bolles, 1981). This algorithm, applied to three-dimensional data, is typically used for detecting geometric primitives in point clouds. The method samples some points from the input data and fits the primitive, which in this case is a planar surface. It then checks the congruency level of the whole point cloud within a threshold and provides the number of inliers. This process is repeated a fixed number of times it and the larger number of inliers is reported. In our case, we run RANSAC for $it = 1000$, in order to assure the predominant plane will likely be fitted. The number of points to sample from the point cloud in each iteration is 3, which is the minimum required to define a plane. The threshold to consider that a point is congruent with the

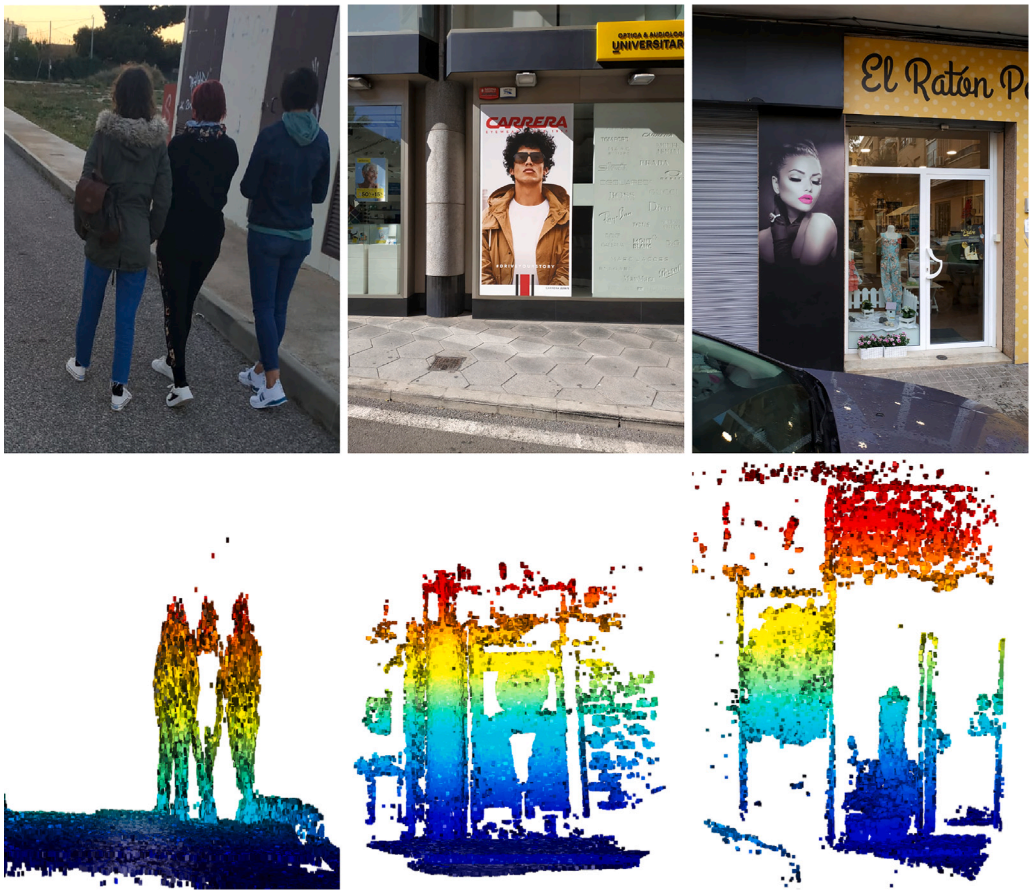


Fig. 2. The top row depicts images from three random image sequences obtained by the camera that are used to perform the scene reconstruction. The bottom row shows SfM reconstructions from the images sequences that contain the top row image samples.

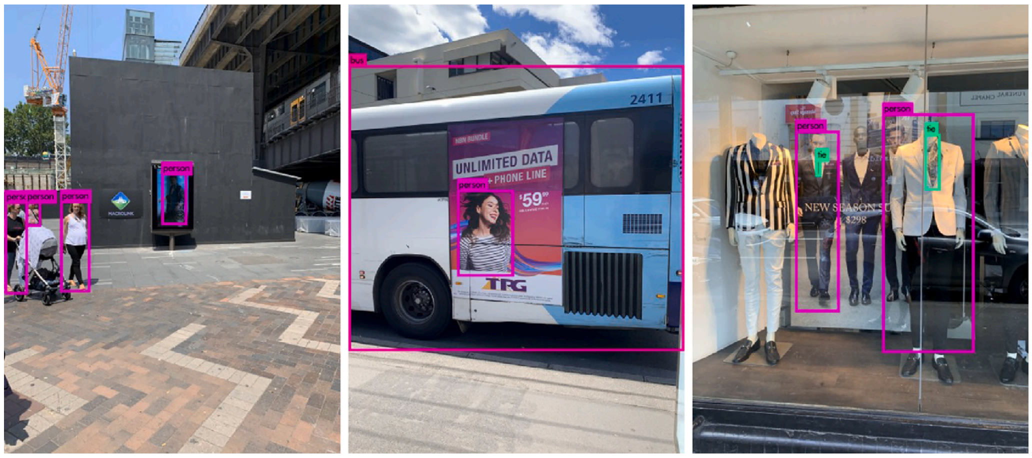
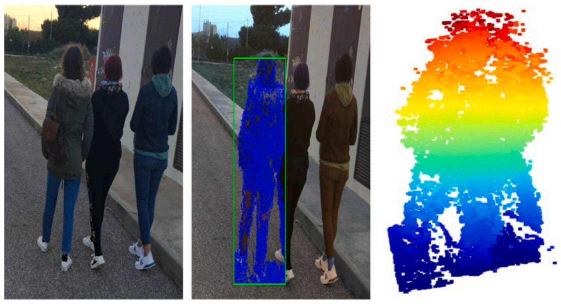


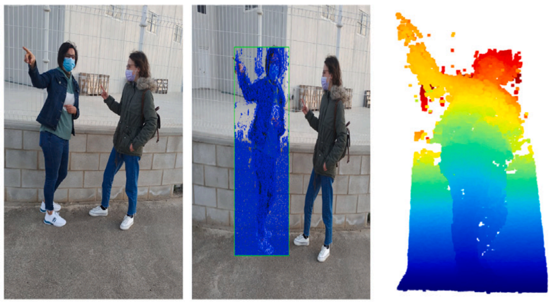
Fig. 3. The object detector of choice is YOLOv3. Note that this architecture labels as a person anything that is visually similar to a person, despite not actually being one. This kind of error is highly dangerous and problematic, for example, in autonomous driving tasks.

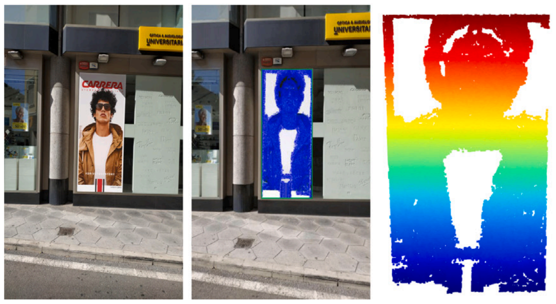
estimated plane is the mean distance between the neighboring points of the input point cloud. Thus, this distance is a function of the density of the point cloud. The final threshold is this mean distance plus a fixed value t_1. We do this because the point cloud yields no real scale as it is computed by an SfM method. If we make t_1 larger, more points will be congruent to the fitted plane and the surface described by these points would be less planar. In contrast, if t_1 is more restrictive, fewer points would be attached to a fitted plane so the best plane returned is likely to be a local one instead of the predominant one.

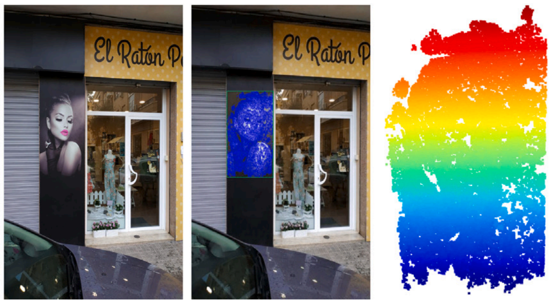
Finally, as the number of inliers is returned by RANSAC, we compute the relative number of points (RNP) that are inliers by dividing the returned inliers by the total number of points in the input point cloud. This makes the score robust to clouds with different sizes.

The normal vector of the plane is then used to state its orientation. First, the normal vector is computed using Eq. (2) from the returned plane $ax + by + cz = d$, and its angle in the Z direction regarding the global coordinate frame is then also calculated. Finally, if the angle in the Z axis is within the 25–155 degree range, it means the plane is


(a) Real person sample 1.


(b) Real person sample 2.


(c) Fake person sample 1.



(d) Fake person sample 1.

Fig. 4. The 3D data of each object labeled as person is computed by extracting the 3D points from the scene point cloud that are inside the bounding box detected in the image.

Table 1
Experimental results (accuracy) on simulated data modifying $t1$ values from 0.001 to 0.05 and RNP value equal to 80%. The background class is divided into two subclasses: samples that were captured parallel to the camera (A) and 45 degrees to the camera (B). Total samples used in this experimentation: 214 for background(B), 140 for actual pedestrian and 229 for background(A).

$t1$ value	RNP value	Actual ped.	Background(A)	Background(B)
0.001	80%	100%	80.35%	0.009%
0.01	80%	99.29%	100%	80.84%
0.02	80%	99.29%	100%	98.13%
0.03	80%	86.43%	100%	98.6%
0.04	80%	81.4%	100%	99.5%
0.05	80%	76.43%	100%	100%

Table 2
Experimental results (accuracy) on simulated data modifying $t1$ values from 0.001 to 0.05 and RNP value equal to 60%. The background class is divided into two subclasses: samples that were captured parallel to the camera (A) and 45 degrees to the camera (B). Total samples used in this experimentation: 214 for background(B), 140 for actual pedestrian and 229 for background(A).

$t1$ value	RNP value	Actual ped.	Background(A)	Background(B)
0.001	60%	100%	98.25%	27.57%
0.01	60%	97.8%	100%	97.2%
0.02	60%	82.86%	100%	100%
0.03	60%	72.86%	100%	100%
0.04	60%	64.93%	100%	100%
0.05	60%	57.9%	100%	100%

perpendicular to the ground plane and the algorithm continues. This is done to ensure that the computed plane does not depict undesired artifacts, like the floor or ground, but those belonging to the object of interest.

$$n = \begin{pmatrix} \frac{a}{\sqrt{a^2+b^2+c^2}} \\ \frac{b}{\sqrt{a^2+b^2+c^2}} \\ \frac{c}{\sqrt{a^2+b^2+c^2}} \end{pmatrix} \tag{2}$$

If the plane is perpendicular to the floor and the final normalized score is above a fixed value T, the corresponding object is considered as background (flat surface). Otherwise, it is considered as an actual person (not a flat surface). Experiments conducted in order to set the best threshold T are described in Section 5.

At this point, it is worth noting that actual pedestrians in an urban environment are non-static. This impacts on the SfM pipeline as it is likely not to provide a completely faithful reconstruction of the person but a very noisy one. Our approach takes advantage of this matter, as we are searching for planar surfaces and these are unlike those shown in Fig. 5.

Table 3
Experimental results (accuracy) on simulated data modifying $t1$ values from 0.001 to 0.05 and RNP value equal to 40%. The background class is divided into two subclasses: samples that were captured parallel to the camera (A) and 45 degrees to the camera (B). Total samples used in this experimentation: 214 for background(B), 140 for actual pedestrian and 229 for background(A).

$t1$ value	RNP value	Actual ped.	Background(A)	Background(B)
0.001	40%	100%	100%	88.78%
0.01	40%	82.14%	100%	100%
0.02	40%	64.28%	100%	100%
0.03	40%	52.86%	100%	100%
0.04	40%	40.71%	100%	100%
0.05	40%	31.43%	100%	100%

4. Datasets and data acquisition

To validate our approach, we used several synthetic and real-world datasets, which are explained in this section. To perform the experiments, three different datasets were used. First, a synthetic dataset that includes actual persons and elements that look like persons was compiled. Then, a dataset captured in a real urban environment was involved in the experiments as well. Some experiments also included a real-world custom dataset of pedestrians and billboards, posters, adverts and similar elements that were captured for qualitative evaluation.

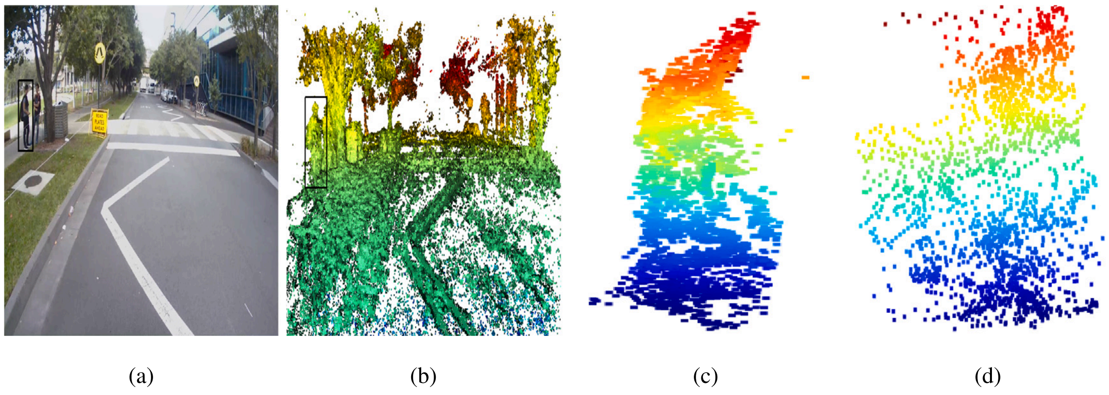


Fig. 5. (a) Image from a random sequence from The University of Sydney Campus Dataset. (b) 3D noisy reconstruction done with OpenSfM from the image sequences. (c) Front-view of the 3D noisy reconstruction of the person with the black bounding box. (d) Side-view of the 3D noisy reconstruction of the person with the black bounding box. Both (c) and (d) are from the same person.

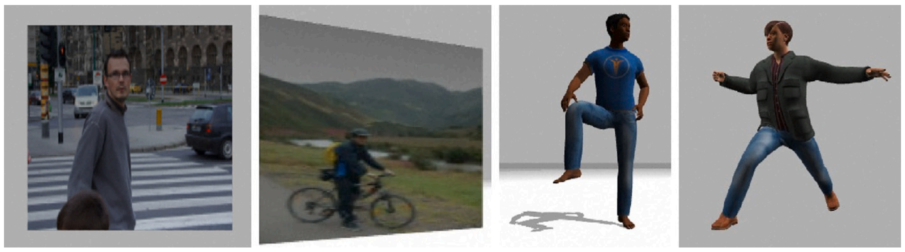


Fig. 6. Random samples of the synthetic dataset featuring planar surfaces and three-dimensional models of persons.

4.1. *Custom synthetic dataset*

First, we collected a synthetic dataset comprising planar surfaces covered by images of persons in urban environments and three-dimensional models of persons in a range of poses. This synthetic dataset also allows us to isolate the subject of interest from neighboring and background elements that would affect the performance of the methods that are tested with it.

The split of the planar surface dataset aims to simulate those urban objects that are visually similar to, but are actually not, persons, for instance, billboards, adverts or posters. Several blocks were placed in a simulated environment. The planar surfaces, which are covered with images of persons extracted from the Pascal VOC (Everingham et al., 2015) dataset, are in parallel (A on Table 1, 2 and 3) and at 45 degrees (B on Table 1, 2 and 3) on the pavement with respect to the simulated car. The car took pictures as it moved around the environment. In total, 140 images of persons were placed on the flat surfaces and 443 images were captured of this setup.

The split of real persons consists of images of three-dimensional models of persons in a variety of poses and appearance. These models were also placed in the simulated world as the virtual car moved around it whilst capturing images. Finally, 20 models of persons in 17 poses were used. This setup allowed us to capture 1768 images. Random samples of this dataset are shown in Fig. 6.

4.2. *Custom real dataset*

This dataset was created with the aim of testing our system with real data in different scenarios. We captured 46 examples of actual pedestrians and 30 examples of objects that are visually similar to, but are actually not, pedestrians. The dataset size is smaller compared to 4.1 and 4.3 because image sequences were captured manually using a smartphone camera. However, every image was captured from the perspective of a vehicle to make data more valuable. Random samples of this dataset are shown in Fig. 7.

4.3. *The university of Sydney campus dataset*

The University of Sydney Campus dataset (Zhou et al., 2019) is composed of different data streams produced by a range of sensors built into an intelligent vehicle. The streams include two color cameras, a LiDAR and GPS positioning, among others. The dataset depicts the same route around the University of Sydney campus for about 60 consecutive weeks, such that a range of different situations, weather and lightning conditions, and changes in the urban environment are captured. As our approach only requires images to work, the left-hand color camera of the vehicle is used for the experiments, while the remaining streams are discarded.

5. Experiments and results

In this section, we present the experimentation carried out to validate our approach. It is worth noting that we used synthetic data, real data and a well-known state of the art dataset. In addition, we also tested the different thresholds and parameters that are used in our proposal.

5.1. *Experiments with custom synthetic dataset*

As mentioned in Section 4.1, a custom synthetic dataset was created to determine whether our system is capable of differentiating these samples of pedestrians. We first tested the system with this dataset because we can control the scene, which cannot be done with real data.

Before implementing the experimentation and obtaining the final results in simulation, two variables need to be set up: the fixed value for RANSAC ($t1$) and the fixed planarity value (RNP) previously described. To do so, an extensive and rigorous experimentation is performed. This experimentation consists of studying the results depending on the value of the variables. All the results are expressed in terms of % accuracy. First, fixing the RNP value to 80%, we perform the real or fake classification task changing $t1$ values from 0.001 to 0.05. In the

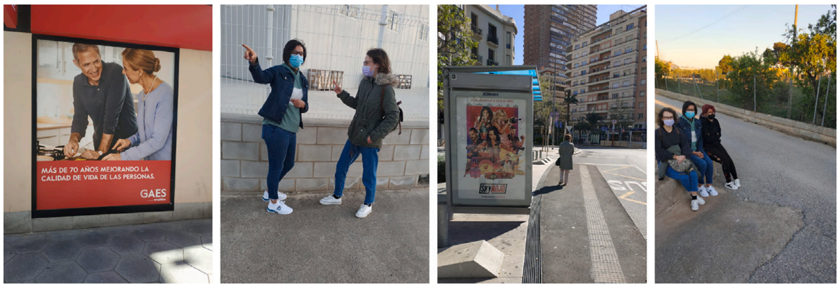
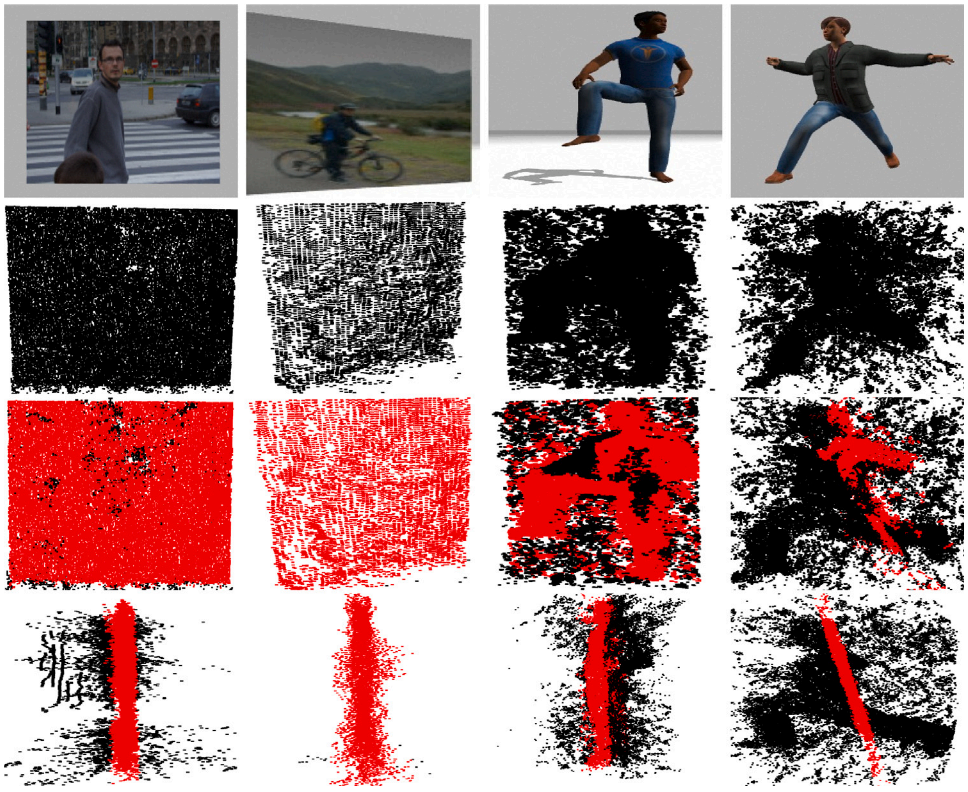

MÁS DE 70 AÑOS MEJORANDO LA CALIDAD DE VIDA DE LAS PERSONAS.
GAES

Fig. 7. Random samples of the real custom dataset with actual and fake pedestrians.



(a) $RAP = 86.2\%$. (b) $RAP = 99.77\%$. (c) $RAP = 57.07\%$. (d) $RAP = 18.47\%$.

Fig. 8. (First row) Images from four random image sequences obtained by the simulated camera that will be used to perform the scene reconstruction. (Second row) SfM reconstructions from the image sequences that contain the top row image samples. (Third row) Front-view of RANSAC plane calculations colored in red over the SfM 3D reconstruction in black. (Fourth row) Side-view of RANSAC plane calculations colored in red over the SfM 3D reconstruction in black.

second place, RNP is decreased to 60% for the same $t1$ values. Finally, we use 40% as the RNP value again with $t1$ values between 0.001 and 0.05. No higher or smaller $t1$ or RNP values are needed because these results already prove the theoretical concept of these variables.

Tables 1–3 show the aforementioned behavior that these variables generate in the results. The best results are obtained in the first experiment, with a $t1$ value equal to 0.02 and an RNP value equal to 80%. Additionally, Fig. 8 shows several results from this experimentation. Although these results are obtained with simulation data, it is a good starting point to demonstrate that our system is able to classify and distinguish the desired samples correctly. Therefore, the experimentation in real environments starts from these values for the variables.

5.2. Experiments with custom real dataset

Starting with the parameter values obtained in the previous experimentation, we test our system with the Custom Real Dataset from Section 4.2. To recall, this dataset contains 37 samples of actual static people and 30 samples of fake people.

In Table 4, we can observe that the RNP and t1 values from the simulation results are sufficiently precise to produce decent results, especially with fake pedestrians. This issue arises because 3D reconstructions of fake people in real scenarios contain more noise than in simulation. Therefore, decreasing RNP value and applying filtering operations are the optimal steps to handle this noise, as can be seen in Table 4.

Once our system is able to obtain decent results with static pedestrians, we add dynamic pedestrian detection by calculating the angle of the plane normal vector with the Z axis, as is explained in Section 3.4. Hence, we add 9 dynamic samples to the real dataset, and we test our system with an RNP value of 50%, while modifying the t1 value from 0.02 to 0.06. The results, shown in Table 5, demonstrates that a t1 value of 0.06 yields the best results when adding dynamic pedestrians,

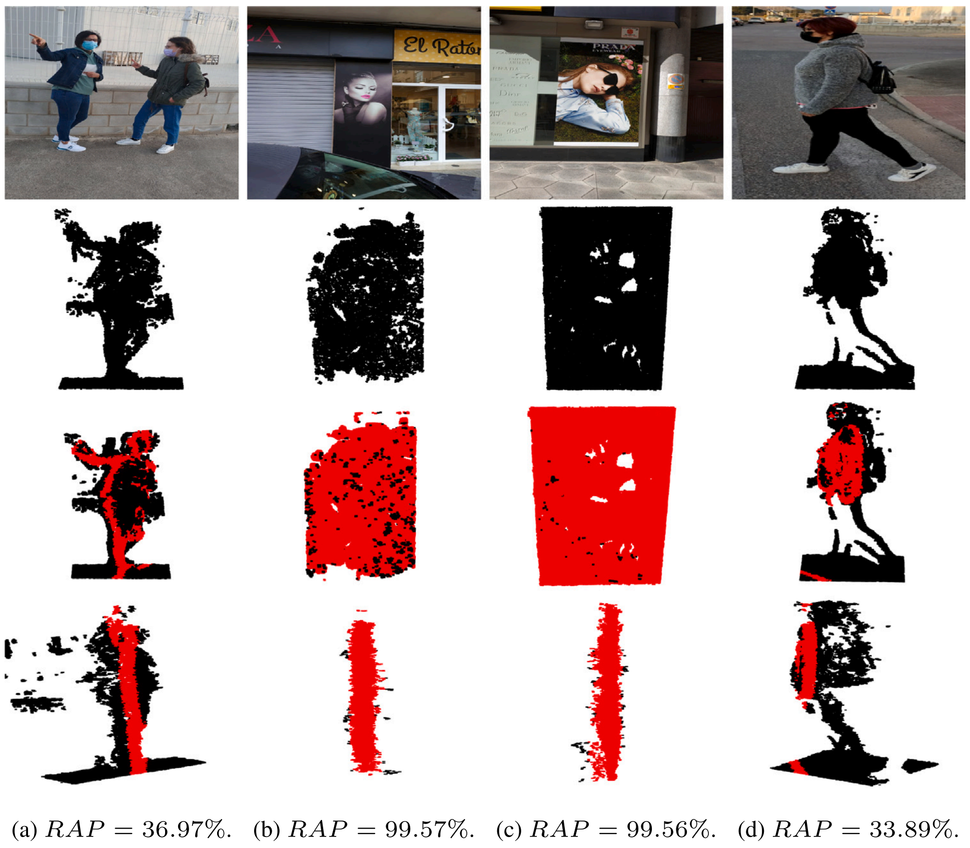


Fig. 9. (First row) Images from four random image sequences obtained by the real camera from the Custom Real Dataset 4.2. (Second row) SfM reconstructions from the image sequences that contain the top row image samples. (Third row) Front-view of RANSAC plane calculations colored in red over the SfM 3D reconstruction in black. (Fourth row) Side-view of RANSAC plane calculations colored in red over the SfM 3D reconstruction in black.

Table 4
Experimental results (accuracy) on real data from Section 4.2 modifying RNP values from 80% to 50% and $t1$ value equal to 0.02. Background class contains all the samples about fake people and pedestrians. Actual pedestrian class contains only static pedestrians. Total samples used in this experimentation: 30 for background, 37 for actual pedestrian.

$t1$ value	RNP value	Actual ped.	Background
0.02	80%	94.6%	75.86%
0.02	70%	91.9%	75.86%
0.02	60%	89.2%	82.76%
0.02	50%	89.2%	93.1%

Table 5
Experimental results (accuracy) on real data from Section 4.3 modifying $t1$ values from 0.02 to 0.06 and RNP value equal to 50%. Background class contains all the samples about fake people and pedestrians. Actual pedestrian class contains both static and dynamic pedestrians. Total samples used in this experimentation: 30 for background, 37 for actual static pedestrian and 9 for actual dynamic pedestrians.

$t1$ value	RNP value	Actual ped.	Background
0.02	50%	100%	73.33%
0.03	50%	97.83%	76.67%
0.04	50%	97.83%	83.33%
0.05	50%	97.83%	80%
0.06	50%	97.83%	86.67%

detecting all of them correctly. Nonetheless, fake people or background class drops from 93% to 87%. Although those results show that our system performs with high accuracy, the number of samples of this dataset is small, and so a final experimentation with a state of the art dataset is needed to verify our results (see Fig. 9).

5.3. Experiments with the university of Sydney campus dataset

Last but not least, we test our system with a state-of-the-art dataset in intelligent vehicles. We ran the entire pipeline over almost 100 reconstructions extracted from this dataset, including both static and dynamic samples. As there are not examples of fake pedestrians in this dataset, we can only test it with actual pedestrians. From these 100 reconstructions, 185 pedestrians were extracted, of which 182 were detected as actual pedestrians and 3 as background. These results show that our system is able to detect actual pedestrians and background in different situations and scenarios, detecting both static and non-static samples as actual pedestrians. Fig. 10 shows examples of results from this last experimentation. As can be seen in Fig. 10(a), when our system classifies a person as a plane (planarity value greater than RNP value), the angle calculation between the normal vector of the detected plane and the Z axis allows us to determine whether this person is indeed a plane or not. In this case, the calculated plane is a horizontal plane, so cannot be a depiction of a person, and is thus a moving pedestrian.

6. Comparison with a state of the art method

In this section, the comparison between our method and a state of the art 3D convolutional neural network, named PointNet++ (Qi et al., 2017), is made. First, PointNet++ is included in our method as the final classifier, replacing RANSAC and RNP threshold. Therefore, after the YOLOv3's detection, when the 3D points from the real or fake person are obtained, PointNet++ is executed to classify this object in real or depicted person, using pre-trained weights from ModelNet dataset. Second, we carry out the comparison only with real datasets, from Sections 4.2 and 4.3, because the final goal of these methods is to run them with real data.

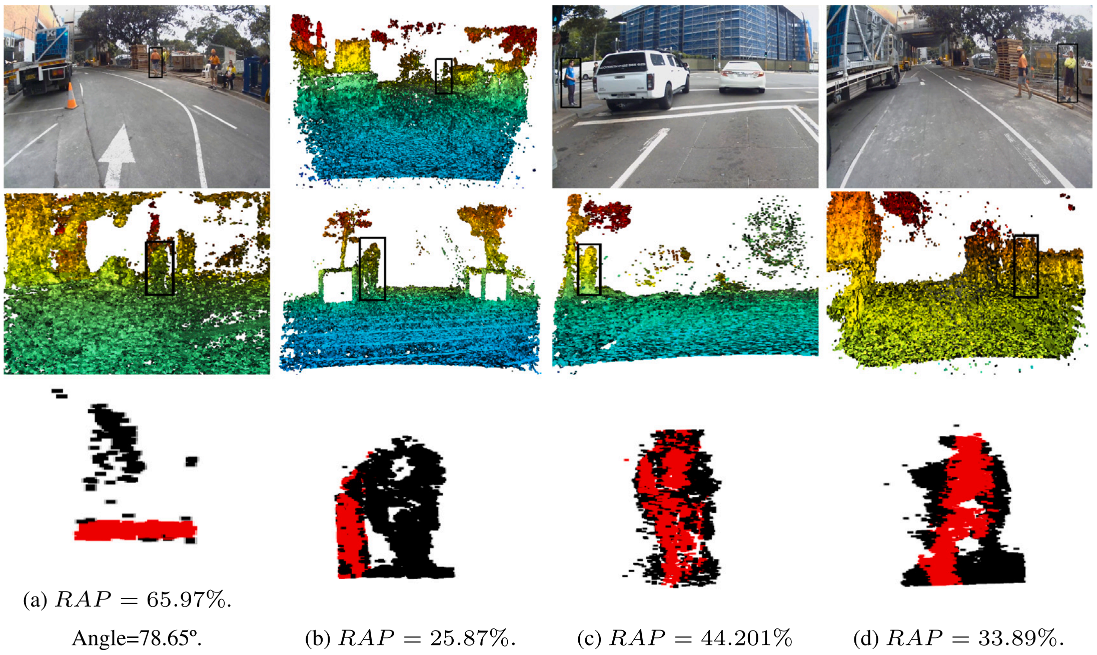


Fig. 10. (First row) Images from four random image sequences obtained by the real camera from the University of Sydney Campus dataset 4.3. (Second row) SfM reconstructions from the image sequences that contain the top row image samples. (Third row) Front-view of RANSAC plane calculations colored in red over the SfM 3D reconstruction in black.

Table 6
Comparison between our method and the state of the art 3D convolutional neural network PointNet++. The results are obtained using the real data from the Custom Real Dataset from Section 4.2.

	Actual ped.	Background
Ours	97.83%	86.67%
PointNet++	54.5%	98%

With respect to our custom dataset, Table 6 shows that PointNet++ barely classifies real people correctly. One limitation of this method is that depends on the structure and density of the point clouds that were used to train the model. In contrast, our method does not depend on any specific data or sensor, instead it would work with any type of point cloud. On the other hand, PointNet++ is able to classify fake people with high accuracy, but our method is also able to achieve promising and reliable results.

Finally, with respect to the state of the art dataset from Section 4.3, our method outperforms PointNet++ when detecting both static and dynamic objects achieving 98.38% of accuracy compared to the 22.22% of PointNet++. Overall, our system performs better than the state of the art method PointNet++, when detecting and classifying between actual or fake pedestrians.

7. Conclusions and future work

In this work, a method is proposed for distinguishing actual objects from depictions of objects for intelligent and autonomous vehicles. The approach merges the output of a 2D object detector with a three-dimensional reconstruction of the scene to rely on structural data to finally classify the objects into actual pedestrians or depictions, such as billboards, posters or printed adverts. The method works with notable success in both synthetic and real-world environments, reaching 97% accuracy in correctly classifying actual pedestrians and 87% accuracy in classifying fake pedestrians (depictions of persons in billboards, adverts or on any surface) in real environments. It is worth noting that a pure 2D classifier, such as YoLo (Bochkovskiy et al., 2020) or SSD (Liu et al., 2016) would miss all the fake samples.

The main advantage of our proposal is that it requires no 3D sensor, such as LiDAR, or 3D cameras to provide accurate pedestrian classification in urban environments.

Nonetheless, it has some limitations. We noticed that the accuracy of the method decreases in the case of reflections, in shops windows, for example. In these cases, our approach is prone to failure. The experimentation also revealed that the Point Cloud Instance Segmentation step sometimes returns a high number of background points, which causes the following step to detect the dominant plane in the floor. As a result, our approach would always find that the category is background, regardless the actual category. The same effect could sometimes be seen if the pedestrian was very close to a wall. In this case, all the background points that lay in the wall would not be filtered and the dominant plane would be detected in them, also being classified as background instead of pedestrian.

As future work, we plan to further delve into the reflection issue to tackle it properly. We also plan to enhance the filtering process so that the floor points and the rest of points that do not belong to the pedestrian are removed before running the planarity test. A promising way to do this would be to involve pixel-wise classification methods, such as Mask-RCNN (He et al., 2017). This would enable us to accurately detect the actual pixels of the subject, but would be more computationally expensive than the approach we currently use to detect pedestrians. In addition, we plan to deploy our method in an actual intelligent vehicle, so that we can test the approach in a fully real-world environment.

CRediT authorship contribution statement

Francisco Gomez-Donoso: Conceptualization, Methodology, Writing – original draft. **Julio Castano-Amoros:** Methodology, Software, Investigation, Writing – original draft. **Felix Escalona:** Conceptualization, Methodology, Writing – original draft. **Miguel Cazorla:** Conceptualization, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All the data used in this work are from publicly available datasets of the state of the art

Acknowledgments

This work was funded by a Spanish Government PID2019-104818RB-I00 grant, supported by Feder funds. It was also supported by Spanish grants for Ph.D. FPU16/00887. Experiments were made possible by a generous hardware donation from NVIDIA.

References

Benenson, R., Omran, M., Hosang, J., & Schiele, B. (2014). Ten years of pedestrian detection, what have we learned? In *European conference on computer vision* (pp. 613–627). Springer.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

Brock, A., Lim, T., Ritchie, J. M., & Weston, N. (2016). Generative and discriminative voxel modeling with convolutional neural networks. arXiv preprint arXiv:1608.04236.

Brunetti, A., Buongiorno, D., Trotta, G. F., & Bevilacqua, V. (2018). Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, *300*, 17–33.

Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1907–1915).

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. arXiv preprint arXiv:1605.06409.

Dollár, P., Wojek, C., Schiele, B., & Perona, P. (2009). Pedestrian detection: A benchmark. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 304–311). IEEE.

Dollar, P., Wojek, C., Schiele, B., & Perona, P. (2011). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(4), 743–761.

Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., & Tian, Q. (2019). Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6569–6578).

Engelcke, M., Rao, D., Wang, D. Z., Tong, C. H., & Posner, I. (2017). Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE international conference on robotics and automation* (pp. 1355–1361). IEEE.

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, *111*(1), 98–136.

Fischler, M., & Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395, URL/brokenurl#http://publication.wilsonwong.me/load.php?id=233282275.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research*, *32*(11), 1231–1237.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440–1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).

Gomez-Donoso, F., Escalona, F., & Cazorla, M. (2020). Par3dnet: Using 3dcnns for object recognition on tridimensional partial views. *Applied Sciences*, *10*(10), 3409.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *2017 IEEE international conference on computer vision* (pp. 2980–2988). http://dx.doi.org/10.1109/ICCV.2017.322.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Jin, X., & Han, J. (2010). K-means clustering. In *Encyclopedia of machine learning* (pp. 563–564). Boston, MA: Springer US, http://dx.doi.org/10.1007/978-0-387-30164-8_425.

Lan, W., Dang, J., Wang, Y., & Wang, S. (2018). Pedestrian detection based on YOLO network model. In *2018 IEEE international conference on mechatronics and automation* (pp. 1547–1551). IEEE.

Law, H., & Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In *Proceedings of the european conference on computer vision* (pp. 734–750).

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980–2988).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21–37). Springer.

Liu, W., Liao, S., Ren, W., Hu, W., & Yu, Y. (2019). High-level semantic feature detection: A new perspective for pedestrian detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5187–5196).

Liu, K., Wang, W., & Wang, J. (2019). Pedestrian detection with LiDAR point clouds based on single template matching. *Electronics*, *8*(7), 780.

Mapillary (2021). Opensfm. https://github.com/mapillary/OpenSfM, Accessed: 2021-05-24.

Özyesil, O., Voroninski, V., Basri, R., & Singer, A. (2017). A survey on structure from motion. CoRR arXiv:1701.08493.

Qi, C. R., Liu, W., Wu, C., Su, H., & Guibas, L. J. (2018). Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 918–927).

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, *30*.

Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. CoRR arXiv:1804.02767.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497.

Sedaghat, N., Zolfaghari, M., & Brox, T. (2016). Orientation-boosted voxel nets for 3D object recognition. arXiv preprint arXiv:1604.03351.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Song, S., Lichtenberg, S. P., & Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 567–576).

Song, S., & Xiao, J. (2016). Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 808–816).

Wu, T., Hu, J., Ye, L., & Ding, K. (2021). A pedestrian detection algorithm based on score fusion for multi-lidar systems. *Sensors*, *21*(4), 1159.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1912–1920).

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1912–1920).

Xu, X., & Todorovic, S. (2016). Beam search for learning a deep convolutional neural network of 3D shapes. URL https://arxiv.org/abs/1612.04774.

Yang, Z., Liu, S., Hu, H., Wang, L., & Lin, S. (2019). Reppoints: Point set representation for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9657–9666).

Zhou, W., Perez, J. S. B., Alvis, C. D., Shan, M., Worrall, S., Ward, J., & Nebot, E. (2019). The USyd campus dataset. *IEEE Dataport*, http://dx.doi.org/10.21227/sk74-7419.