- - - - -

A Synthetic Human-Centric Dataset Generation Pipeline for Active Robotic Vision

Charalampos Georgiadis[a], Nikolaos Passalis[a], Nikos Nikolaidis[a,*]

[a]*Department of Informatics, Faculty of Sciences, Aristotle University of Thessaloniki, Thessaloniki, Greece*

Abstract

Active vision aims to equip computer vision methods with the ability to dynamically adjust the capturing sensor's viewpoint, position, or parameters in real time. This dynamic capability allows for improving the accuracy of the perception process. However, training and evaluating an active vision model often requires a large number of annotated images captured under different sensor and environmental settings, in order to emulate actions like moving around, approaching, or moving away from a person and thus effectively model the active perception dynamics. Obviously, collecting and annotating such datasets is a challenging and expensive task. To overcome these limitations, this paper introduces a synthetic image generation pipeline specifically designed to support active vision tasks. The pipeline is developed using a highly realistic simulation framework based on Unity and allows for the generation of images depicting humans, captured at varying view angles, distances, illumination conditions, and backgrounds, supporting a wide range of different tasks. Two annotated datasets, namely ActiveHuman and ActiveFace, are generated using the pipeline and the effectiveness of the proposed approach is demonstrated by a solid use case that involves training and evaluating an embedding-based active face recognizer. Furthermore, we demonstrate how the proposed generation approach enables expanding existing active face recognition methods by training models that control both the left/right movements, as well as the distance to a subject, leveraging the additional information provided by ActiveFace dataset. To facilitate replication and encourage the use of the generated datasets for training and evaluating other active vision approaches, the associated assets and the developed dataset generation pipeline is to become publicly available.

1. Introduction

Active vision is a subfield of computer vision that draws inspiration from our ability to appropriately navigate our environment to gain a better understanding of our surroundings. Its objective is to enhance the efficiency of traditional computer vision methods by enabling the capturing sensor(s), positioned on an autonomous system,

*Corresponding author.

Email addresses: georgicd@csd.auth.gr (Charalampos Georgiadis), passalis@csd.auth.gr (Nikolaos Passalis), nnik@csd.auth.gr (Nikos Nikolaidis)

such as a robot, to dynamically adjust their viewing position, direction, or parameters in real-time. This dynamic adaptation allows models to make more informed decisions regarding the subject of interest. Active vision models find applications in various computer and robotic vision tasks, including face and object recognition/detection (Mahaur and Mishra, 2023; Luo et al., 2020), human pose estimation (Kumarapu and Mukherjee, 2021), and have demonstrated advantages in terms of speed (Passalis and Tefas, 2020), and size of models (Pan et al., 2021), as well as improved accuracy (Passalis and Tefas, 2020; Pan et al., 2021; Kakaletsis and Nikolaidis, 2023; Murali et al., 2022) compared to models employing a static approach.

Although there is a wealth of datasets containing a large number of annotated images for various computer and robotic vision tasks (Geiger et al., 2013; Sikder and Nahid, 2021), datasets specifically designed for active vision problems are relatively scarce. Training and evaluating an active vision model often necessitates the use of a large number of annotated images captured under diverse sensor and environmental configurations to grasp the dynamics underlying the active perception process. However, collecting and annotating such datasets is a challenging and expensive task. It involves not only providing carefully crafted ground truth annotations but also accurately modeling the effect of various actions, e.g., the movement towards or around a person. Two datasets that support active vision tasks are the ModelNet dataset (Wu et al., 2015) and the Active Vision dataset (Ammirato et al., 2017), both suitable for active object detection and recognition. The ModelNet dataset encompasses over 150,000 3D CAD model images from 161 object categories, captured at various angles. The Active Vision Dataset comprises more than 30,000 RGBD real-world images representing 15 different scenes, accompanied by over 70,000 2D bounding box annotations. Regarding face detection/recognition tasks, there have been instances where active vision models were trained using smaller datasets, such as the HPID dataset (Gourier et al., 2004) (Head Pose Image Database). Nevertheless, existing datasets are relatively limited in size and they often do not provide adequate data for supporting training active vision pipelines for different tasks. For instance, the HPID dataset contains only 2,790 facial images, while it does not support training models that can control the distance between the robot and a subject to be recognized, significantly constraining the practical use of such models.

Recently, there have been attempts to create frameworks for generating synthetic annotated datasets in computer and robotic vision tasks, including active vision. Two examples are BlenderProc (Denninger et al., 2019), an open-source extension of Blender (Blender Foundation, 2018), and Nvisii (Morrical et al., 2021). BlenderProc provides a flexible pipeline that can generate realistic synthetic images with annotations. Nvisii, on the other hand, allows the generation of realistic synthetic images with additional information like bounding boxes, segmentation masks, and optical flow vectors. However, it's worth noting that these frameworks, despite offering visual realism, may have specific limitations in terms of certain aspects of computer vision or robotics-related simulation functionalities and realism, as well as physics, when compared to more advanced frameworks, such as Unity's Perception package (Unity Technologies, 2020). For example, both BlenderProc and Nvisii do not support domain randomization (which is critical for training and testing computer vision or robotics-related methods), whereas BlenderProc does not generate keypoint annotations. Another effective

approach for training and testing active vision methods is the use of photorealistic simulators designed for autonomous systems, robots, or embodied AI applications, Habitat-Sim[1] being a notable example. For instance, a real-time active vision humanoid soccer robot was trained and evaluated in a simulation environment by Khatibi et al. (Khatibi et al., 2021) using deep reinforcement learning, demonstrating how using a simulation environment can indeed be very effective in active vision tasks. However, employing image/video datasets within simulation environments can offer advantages during the initial stages of algorithm development, as it simplifies the handling of robot motion and provides a convenient platform for training and testing such algorithms.

The main research question examined in this paper is whether it is possible to develop realistic synthetic human-centric image generation pipelines specifically designed to support active vision tasks, enabling capabilities that go beyond existing datasets with minimal data collection and annotation effort. To this end, we introduce a realistic synthetic human-centric image generation pipeline, which is built using a modified version of Unity's Perception package (Unity Technologies, 2020), integrated into a URP project. This allows the generation of images captured around humans across a wide range of view angles, distances, illumination conditions, and backgrounds (Fig. 1), enabling training active perception models for different tasks. To validate the effectiveness of the proposed pipeline we also created two annotated datasets, ActiveHuman and ActiveFace. Furthermore, we also employed ActiveFace dataset to apply and considerably enhance an embedding-based active face recognizer (Passalis and Tefas, 2020), providing a realistic use case that showcases the value of the generated datasets. Through the proposed extension, we can perform active perception by controlling two axes, i.e., controlling both the left/right movements and the distance from a subject, surpassing the capabilities of the initial method and attaining superior results. The datasets, along with the associated assets and dataset generation pipeline, will be made publicly available[2] in order to allow anyone to seamlessly replicate them, as well as use the generated datasets for training and evaluating other active vision approaches. Therefore, the main contributions of this paper are as follows:

(i) a realistic synthetic human-centric image generation pipeline that enables training active perception models,
(ii) two human-centric annotated datasets generated using the proposed approach covering a variety of different scenarios and setups, as well as
(iii) a realistic use case that showcases the value of the generated dataset, i.e., employing the ActiveFace dataset to considerably enhance an embedding-based active face recognizer approach.

The rest of the paper is structured as follows. First, the datasets are introduced in Section 2 along with a detailed description of their generation. Subsequently, the active vision method used to evaluate the facial image dataset is presented in Section 3, followed by an extensive experimental evaluation provided in Section 4. Finally, conclusions are drawn in Section 5.

[1] `https://github.com/facebookresearch/habitat-sim`
[2] Public release under preparation, will be available by the time the review process is completed.

2. Datasets Description

The process of generating the first dataset, namely the *ActiveHuman* dataset, can be outlined using a nested for-loop structure, illustrated in Algorithm 1. This algorithm iterates through all possible combinations of environments ($\mathcal{E}$), human models ($\mathcal{H}$), and lighting conditions ($\mathcal{L}$). By varying the camera angle and distance from the human subject, the algorithm captures different views. To construct the dataset, we utilized freely available environmental assets and human models from sources like the Unity Asset Store, Maximo, Turbosquid, as well as human models created using MakeHuman. The setup of the Unity Perception package project enables easy addition of new environments or human models without requiring modifications to the scripts responsible for altering the environmental or sensor configurations of each captured image (randomizer).

Algorithm 1 : Dataset generation procedure.

```
for each environment E in 𝓔 do
  for each human H in 𝓗 do
    for each lighting condition L in 𝓛 do
      for each camera position P from [1m − 4m] in increments of 0.5m do
        for each camera angle Θ from [0 − 360] in increments of 10 degrees do
          Capture and output images and metadata
        end for
      end for
    end for
  end for
end for
```

We captured $175,428$ images with dimensions 1600×900 for every valid combination of 8 environments, 33 human models, 4 lighting conditions, 7 camera distances $1m - 4m$ from the subject in increments of $0.5m$, and 36 camera angles around the subject in 10 degrees increments. In each captured image, the human subject is positioned at the center, as depicted in Fig. 1. The objective was to imitate robot motion in all feasible and permissible locations surrounding a human, considering various lighting conditions that simulate different times of the day within realistic environments. These environments were designed to resemble typical rooms such as living rooms, bedrooms, and kitchens, furnished with items like tables, chairs, and beds. However, due to the presence of furniture, certain locations within a room were inaccessible to the camera-equipped robot whose motion the dataset imitates. Consequently, the dataset does not include images from such inaccessible locations that are occupied by furniture. In more detail, if the camera collided/coincided with an object in the given environment for a certain position P and angle Θ all combinations that involved E, P and Θ were deemed invalid. Regarding humans, out of the 33 used in the generation process, 17 are females (1 infant and 16 adults), while the remaining 16 are males (1 infant and 15 adults).

The generated dataset, apart from the captured RGB images, also contains their

semantic instance segmentation masks, as shown in Fig. 2, as well as, annotation files which are comprised of camera parameters, 2D bounding box, 3D bounding box annotations of humans and selected objects/entities (chairs, tables, lamps, floor, ceiling, windows etc) for each captured image. Key points annotations are also included for humans (Fig. 3). Each object and human in the scene is labeled accordingly with its ID in order to have access to the 2D and 3D coordinates of every visible entity of each captured image. The key points annotations describe 18 body parts and their respective skeleton connections and are generated in the COCO format. In addition, a file consisting of definitions and color codes (when applicable) for each annotation is also included. In this way, the generated dataset can support a wide variety of different (active) perception tasks, ranging from semantic scene segmentation to human pose estimation. We have also generated a face recognition-oriented version of this dataset, cropping only the facial images from the generated dataset, aiming to support specifically active face recognition tasks. Some examples of the synthetic images contained in the facial dataset are shown in Fig. 4. We call this version of the dataset *ActiveFace* to distinguish it from the full high resolution *ActiveHuman* dataset.

3. Active Face Recognition

In this Section we provide an active perception use case using the proposed*ActiveFace* dataset, building upon the embedding-based active face recognition method presented in (Passalis and Tefas, 2020). This method was shown to yield much better recognition results than the ones achieved when using a static perception approach, since it takes advantage of a robot's ability to interact with its environment in order to get a more informative view of the person's face. We demonstrate how the rich annotation and variety of data provided by the proposed dataset, enables us to further extend this active perception approach, further improving its performance, e.g., by allowing to perform control in additional axes. This is achieved with the use of a trainable controller which, when given an image $\mathbf{x}^{(t)}$ at a time t, dictates the robot to move towards a certain direction in order to acquire a new image which offers a better frontal view of the person. The new image is given by:

$$\mathbf{x}^{(t+1)} = v(a_t, t), \tag{1}$$

where $v(\cdot)$ denotes the current environment. The trainable controller is represented as:

$$a_t = g_{\theta_c}(\mathbf{x}^{(t)}), \tag{2}$$

where θ_c denotes a set of trainable action parameters.

The model is comprised of two modules, the feature extractor model $f_{\theta_r}(\cdot)$, which learns discriminative embeddings of a given face image, thus being able to separate the representations extracted from images that belong to different persons, and the controller model $g_{\theta_c}(\cdot)$ which is responsible for learning the best possible action that the robotic system should take next in order to acquire a better view of a person's face.

When an unseen image is given as input during the evaluation process and the controller has given the appropriate control commands to the robotic system, the id of the person is obtained using the 1-nearest neighbor approach on a database that contains frontal and nearly frontal facial images for every person.

Instead of using reinforcement learning when training the controller, the model executes all possible control actions at the same time and calculates the recognition accuracy of each of the obtained images, improving learning efficiency (Passalis and Tefas, 2020). The action that led to the lowest distance between the representation of the current face and the correct face is retained and used to train the controller. The optimal action when given an image x_i and a correct image x_p is given by:

$$d_i^{(a)} = \underset{k \in 0,1,2,\ldots,n}{\arg\min} \|f(\mathbf{x}_{ik}) - f(\mathbf{x}_p)\|_2, \tag{3}$$

where n is the total number of possible actions that the controller can choose.

The loss function that the controller aims to minimize is given by:

$$L_g = \sum_{i=1}^{N} L_x(g_{\theta_c}(\mathbf{x}_i), d_i^{(a)}), \tag{4}$$

where L_x represents the cross-entropy loss function. The feature extractor, on the other hand, aims to minimize the following loss function:

$$L_f = \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} L_e(f_{\theta_r}(\mathbf{x}_i), f_{\theta_r}(\mathbf{x}_j), d_{ij}), \tag{5}$$

where the the binary variable $d_{ij} \in \{0, 1\}$ denotes whether the i-th face image belongs to the same person as the one depicted in the j-th face image and L_e is a loss that encourages the separability of different face embeddings. In this work we use the contrastive loss, as suggested in (Passalis and Tefas, 2020), which is minimized when embeddings that belong to the same identity are as close as possible, while the representations of face images that do not belong to the same person maintain at least a distance of $\sqrt{m}$:

$$L_e(\mathbf{y}_i, \mathbf{y}_j, d_{ij}) = d_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|_2^2 + (1 - d_{ij}) \max(0, m - \|\mathbf{y}_i - \mathbf{y}_j\|_2^2), \tag{6}$$

where $\mathbf{y}_i = f_{\theta_r}(\mathbf{x}_i)$ is the representation extracted from the face recognition model and $\|\cdot\|_2$ refers to the l^2 norm of a vector. The final loss of the model is given by the sum of (4) and (5):

$$L = L_g + L_f \tag{7}$$

The model uses the Adam optimization algorithm with initial learning rates $\eta_r = \eta_c = 10^{-3}$ for the feature extractor and controller, respectively.

We also appropriately modified the aforementioned approach to allow for an extra `Front` (i.e., towards the subject) movement/action of $0.5m$ per move in order to take advantage of the range of camera-subject distances provided by the dataset generated in this work. The `Left` and `Right` actions dictate the controller to move by 10 degrees either to the left or to the right, respectively, on a circle centered at the human subject. Since the classes involved in (4) are not balanced, different weights were used for different classes. More specifically, the `Stay` action was given a action weight of 0.01, while both the `Left` and `Right` ones were given a weight of 1 and the `Front` was weighted by 1.2.

Since the dataset does not contain, due to the existence of furniture, images from every camera/robot position, it was observed that the model was not always able to find an existing image for every available action. As each environment had missing images at different camera distances and angles (i.e., for the locations occupied by the furniture) and the model could learn to avoid collisions for environments that do not require such actions, it was decided to not train the model for any image where any of the left, right or front images are missing. During inference, the controller chooses the best action that leads to an image that exists. If for a given image there are no left, right or front images the controller dictates the robotic system to stay in place. In a real-world scenario, the controller would output different actions, from most optimal to less optimal, until the robotic system could move towards the best available spot.

4. Experimental Evaluation

Details for the experimental evaluation are provided in this Section. First, the experimental setup that was used for training each model upon the *ActiveFace* dataset is presented. Then, both the static (i.e. the approach that decides on the person's identity using the initial image) and the extended active vision methods are evaluated using various configurations and the experimental results are discussed.

4.1. Experimental Setup

The active vision model was evaluated on both the entire *ActiveFace* face image dataset (Set 1) and on a subset (Set 2) of the dataset containing only facial images with a pan range of -90° to 90° (0° corresponds to frontal view). In both cases, the training set consisted of 22 subjects, while the remaining 11 were used to evaluate the trained model. For those 11 subjects, all the frontal and nearly frontal (-10° to 10°) images at $1m$ distance away from the human, for every environment and lighting condition, were added to the recogniser database, while the remaining ones were used for testing the trained model, i.e., they were used as images captured at the starting location of the robot. All images were resized to 96×96 and all experiments were conducted 5 times using different random seeds and the mean and standard deviation of their accuracy scores was recorded. For each of Sets 1 and 2 both a static and an active vision model were trained in order to evaluate the increase in accuracy when using the latter method. It is expected that the network will perform worse on the entire dataset (Set 1) compared to its accuracy score on the -90° to 90° subset (Set 2), since the model may not even detect a face for extreme pan values and large distances. The active model was first pretrained without the control branch and then trained simultaneously on both the feature extractor and the control branch.

The static vision model was trained for 5, 10, 20 and 30 epochs for both Set 1 and Set 2. The active model was trained for 10 (5 for the feature extractor and 5 for both branches), 20 (10 for the feature extractor and 10 for both branches) and 30 (15 for the feature extractor and 15 for both branches) epochs for both subsets. Moreover, the active vision model was evaluated for 30 control steps, which would essentially allow the robotic system to move to any location in an environment. This way, the recognition accuracy ceiling of the model for both Sets 1 and 2 will be reached.

Finally, the active model was also trained and evaluated without the addition of the extra `Front` action in order to demonstrate how allowing the robotic system to move towards the subject can result in an increase in inference performance. Note that we use the DL networks and follow the hyper-parameters proposed in (Passalis and Tefas, 2020), unless otherwise stated.

4.2. Evaluation Results

In this subsection, we provide the empirical evaluation of the proposed method. First, we provide results for training both static and active perception models. We also report results on different points of the training curve (10, 20, and 30 epochs) in order to provide a more complete evaluation. Then, we provide experimental results where we evaluate the impact of the number of active perception control steps on the accuracy of the models. Finally, we conclude this section by evaluating the impact of the size of the training set on the accuracy of the models.

First, we compare the proposed method to both a static model with the same architecture ("Static"), as well as to a more powerful model (Inception-ResNet (v1) (Szegedy et al., 2017)) (abbreviated as "Static (Inception-ResNet)"). As a reminder, **Set 1** represents the full ActiveFace dataset, while **Set 2** denotes the dataset with the reduced pan range. The experimental results are reported in Table 1. Using the more powerful static model leads to better results than the baseline architecture. However, active perception leads to improved accuracy in both setups. The proposed active perception agent ("Active (Proposed)"), which can control the distance between the subject and the camera, leads to overall best results, outperforming both the static perception approaches as well as the simpler method proposed in the literature (Passalis and Tefas, 2020) ("Active (1 axis)"). .

Table 1: Comparison between static perception and active perception models. Face recognition accuracy mean and standard deviation are reported in both Set 1 and Set 2. All models were trained for 20 epochs.

Model	**Set 1**	**Set 2**
Static	44.9 ± 2.4%	57.9 ± 2.9%
Static (Inception-ResNet)	46.4 ± 3.4%	65.1 ± 2.9%
Active (1 axis)	55.5 ± 1.9%	66.6 ± 6.8%
Active (Proposed)	**69.2 ± 7.6%**	**79.1 ± 1.7%**

Table 2: Static vision model evaluation at various points of the training process: accuracy mean and standard deviation.

Model	**Set 1**	**Set 2**
Static (5 epochs)	51.1 ± 4.2%	60.3 ± 1.5%
Static (10 epochs)	47.9 ± 4.2%	58.1 ± 3.5%
Static (20 epochs)	44.9 ± 2.4%	57.9 ± 2.9%
Static (30 epochs)	44.3 ± 2.2%	58.5 ± 2.8%

Table 3: Active vision model evaluation **with** the additional `Front` movement/action at various points of the training process: accuracy mean and standard deviation.

Model	**Set 1**	**Set 2**
Active (10 epochs)	67.9 ± 6.8%	76.9 ± 6.5%
Active (20 epochs)	69.2 ± 7.6%	79.1 ± 1.7%
Active (30 epochs)	67.4 ± 8.6%	78.3 ± 6.8%

Table 4: Active vision model evaluation **without** the additional `Front` movement/action at various points of the training process: accuracy mean and standard deviation.

Model	**Set 1**	**Set 2**
Active (10 epochs)	60.3 ± 6.4%	66.4 ± 7.3%
Active (20 epochs)	55.5 ± 1.9%	66.6 ± 6.8%
Active (30 epochs)	59.1 ± 4.4%	72.1 ± 3.2%

The evaluation results for different number of epochs are shown in Tables 2, 3 and 4. Evaluation results for the static model are presented in Table 2. Clearly, the models perform best when trained for 5 epochs, reaching accuracy scores of ∼51.1% and ∼60.3% for **Set 1** and **Set 2**, respectively. Increasing the number of epochs seems to cause an overfit of the model on the training data.

Again, once the active approach is employed, a substantial increase in prediction accuracy can be observed for both datasets by a maximum of ∼18.1% and ∼18.8%, respectively, as seen in Table 3. Since we introduce more parameters, the models can be trained for more epochs and seem to overfit when the number of epochs is set to 30 (15 for the feature extractor and 15 for both branches). Evidently, the ability to train the robotic system to move within its environment in order to get a more informative view of the subject, namely a view which is closer to the frontal or nearly frontal views that the system has learned to recognize, yields much better face recognition results. Furthermore, once the controller's `Front` movement is removed (Table 4) the model is ∼8.9% and ∼7% less accurate than the one with the additional `Front` action, when comparing the highest recorded prediction accuracy scores of each respective conducted experiment.

This clearly demonstrates that allowing the model to move in more directions, i.e., not only around but also towards the subject, can further increase its ability to recognize faces.

Figure 5 depicts an example of how the control branch has learned to change its viewpoint in order to get a better (more closer and towards a frontal position) view of the person depicted in the original image, that is, the one obtained from the initial robot location. The original image is obtained from point **a** (starting position for the robot) and then the robot moves along the depicted path until it reaches a frontal view of the subject's face at a distance of $1m$ (point **h**). We can observe that, generally, at larger distances the controller prefers to make a `Front` movement in order to move closer to the subject and, thus, increase the captured facial image resolution. Once the image is clear enough, it then makes either `Left` or `Right` movements in order to move in front

of the subject.

Furthermore, we evaluated the prediction accuracy of each trained model as the number of allowed steps increased from 1 to 20 steps. Our hypothesis was that a well-trained model, neither underfitted nor overfitted, would reach a point where its performance stagnates. This suggests that the model does not need to take additional steps to obtain a better view of the subject's face. Fig 6a illustrates the average prediction accuracy of the model trained on Set 2 for 20 epochs, incorporating the `Front` movement command per maximum allowed number of steps. We can observe that the accuracy increases as the number of steps increases until reaching a plateau at $n = 12$. This indicates that the active perception process converges and consistently produces better results with a higher number of steps up to a point, where the best view has been obtained.

Additionally, we recorded the average number of images that required a certain number of steps ($n = 1, 2, ..., 20$) before the active perception process stopped. Fig. 6b represents the percentage of images that needed a specific number of steps to make a prediction for the same model. Most images required 5 steps, but the proportion of images requiring additional steps decreased gradually. Notably, at 20 steps, the recorded percentage appears to increase. We identified that this occurs in some cases where the controller reaches a frontal view of the subject's face but continues moving towards the left or right without stopping. This suggests that the agent may not be robust enough to consistently choose the `Stay` command when the robotic system achieves a frontal view of the subject.

To highlight the importance of generating additional data for active perception algorithms, we conducted additional experiments to evaluate the impact of the volume of data used for training on the accuracy of the final model. To this end, we performed experiments using 5 persons (about 23% of the original training set), 10 persons (about 45% of the original training set), 15 persons (about 68% of the original training set) and 20 persons(about 91% of the original training set) in the training set using both a static perception setup, as well as the proposed active perception setup. For all the conducted experiments we used the second set, in order to better demonstrate the impact of the additional data included in the training set.

The experimental results are shown in Table 5. Several interesting results can be drawn from these results. First, using more training data increases face recognition accuracy in both the static perception and active perception setups. Furthermore, these results also demonstrate that using smaller training sets can have a more profound negative effect on active perception models since the initial accuracy for active perception is smaller (about 36% for static perception and about 34% for active perception when using 23% of the training dataset). However, larger increases are observed in the accuracy of the active perception model when additional training data are included, also validating the value of the generated data when training active perception models.

5. Conclusions

This paper introduced a publicly available synthetic and realistic data generation pipeline using Unity's Perception package for training and evaluating active vision methods. Two public datasets were generated using this pipeline, comprising of high

Table 5: Evaluating the impact of the size of the training dataset to the accuracy of the resulting models. "Static Perception" refers to the regular static face recognition setup, while "Active Perception" refers to the proposed active perception approach that supports front movements.

% training data	Static Perception	Active Perception
23% (5 persons)	36.4 ± 2.8%	34.4 ± 1.2%
45% (10 persons)	43.5 ± 5.2%	47.5 ± 1.5%
68% (15 persons)	56.9 ± 1.8%	75.1 ± 3.1%
91% (20 persons)	57.7 ± 2.9%	78.5 ± 6.2%
100% (all persons)	57.9 ± 2.9%	79.1 ± 1.7%

resolution annotated images (instance segmentation masks, 2D and 3D bounding boxes, human body keypoints) depicting humans in various environments, as well as cropped facial images extracted from the originally captured 1600×900 images. These facial images were then utilized to train and evaluate both static and active vision embedding-based face recognizers, showcasing the significant improvement in prediction accuracy achieved through the active approach compared to the static model. Experimental results demonstrated a substantial increase in recognition performance, with a maximum improvement of approximately 18.8%. Moreover, we illustrated that enabling the robotic system to move towards the subject, rather than being limited to left or right movements on a circle as in (Passalis and Tefas, 2020), led to better recognition accuracy, with a maximum observed difference of around 8.9%.

Acknowledgment

This work was supported by the European Union's Horizon 2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

Ammirato, P., Poirson, P., Park, E., Kosecka, J., Berg, A.C., 2017. A dataset for developing and benchmarking active vision, in: Proceedings of the IEEE International Conference on Robotics and Automation.

Blender Foundation, 2018. Blender - a 3D modelling and rendering package. Technical Report. Stichting Blender Foundation. Amsterdam.

Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., Katam, H., Teja, H., 2019. Blenderproc. arXiv preprint arXiv:1911.01911 .

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

Gourier, N., Hall, D., Crowley, J.L., 2004. Estimating face orientation from robust detection of salient facial features, in: Proceedings of Pointing, ICPR, International Workshop on Visual Observation of Deictic Gestures, Cambridge, UK.

Kakaletsis, E., Nikolaidis, N., 2023. Using synthesized facial views for active face recognition. Machine Vision and Applications Accepted.

Khatibi, S., Teimouri, M., Rezaei, M., 2021. Real-time active vision for a humanoid soccer robot using deep reinforcement learning, in: Proceedings of the International Conference on Agents and Artificial Intelligence.

Kumarapu, L., Mukherjee, P., 2021. Animepose: Multi-person 3d pose estimation and animation. Pattern Recognition Letters 147, 16–24.

Luo, J., Liu, J., Lin, J., Wang, Z., 2020. A lightweight face detector by integrating the convolutional neural network with the image pyramid. Pattern Recognition Letters 133, 180–187.

Mahaur, B., Mishra, K., 2023. Small-object detection based on yolov5 in autonomous driving systems. Pattern Recognition Letters 168, 115–122.

Morrical, N., Tremblay, J., Lin, Y., Tyree, S., Birchfield, S., Pascucci, V., Wald, I., 2021. Nvisii: A scriptable tool for photorealistic image generation. arXiv preprint arXiv:2105.13962 .

Murali, P.K., Dutta, A., Gentner, M., Burdet, E., Dahiya, R., Kaboli, M., 2022. Active visuo-tactile interactive robotic perception for accurate object pose estimation in dense clutter. IEEE Robotics and Automation Letters 7, 4686–4693. doi:10.1109/LRA.2022.3150045.

Pan, N., Zhang, R., Yang, T., Cui, C., Xu, C., Gao, F., 2021. Fast-tracker 2.0: Improving autonomy of aerial tracking with active vision and human location regression. IET Cyber-Systems and Robotics 3, 292–301.

Passalis, N., Tefas, A., 2020. Leveraging active perception for improving embedding-based deep face recognition, in: Proceedings of the IEEE International Workshop on Multimedia Signal Processing, pp. 1–6.

Sikder, N., Nahid, A.A., 2021. Ku-har: An open dataset for heterogeneous human activity recognition. Pattern Recognition Letters 146, 46–54.

Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A., 2017. Inception-v4, inception-resnet and the impact of residual connections on learning, in: Proceedings of the AAAI Conference on Artificial Intelligence.

Unity Technologies, 2020. Unity perception package. https://github.com/Unity-Technologies/com.unity.perception.

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.

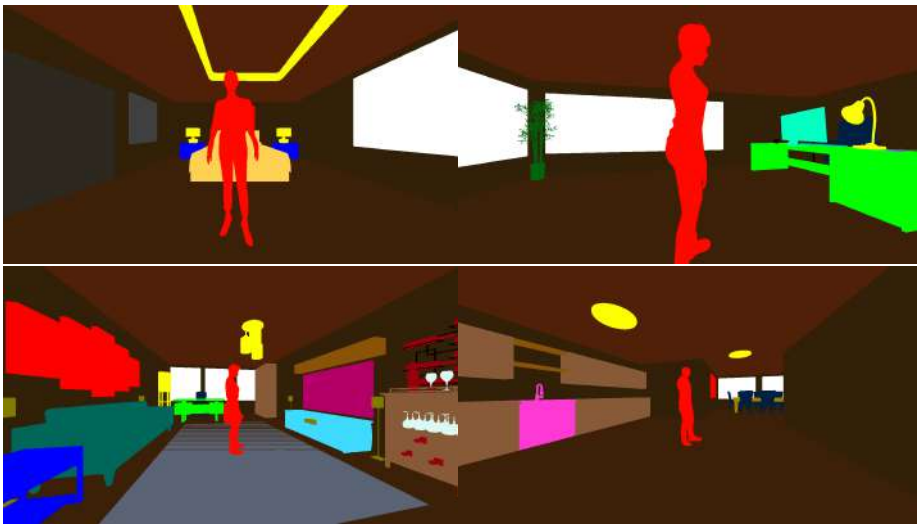


Figure 2: Examples of segmentation masks generated along with the dataset.

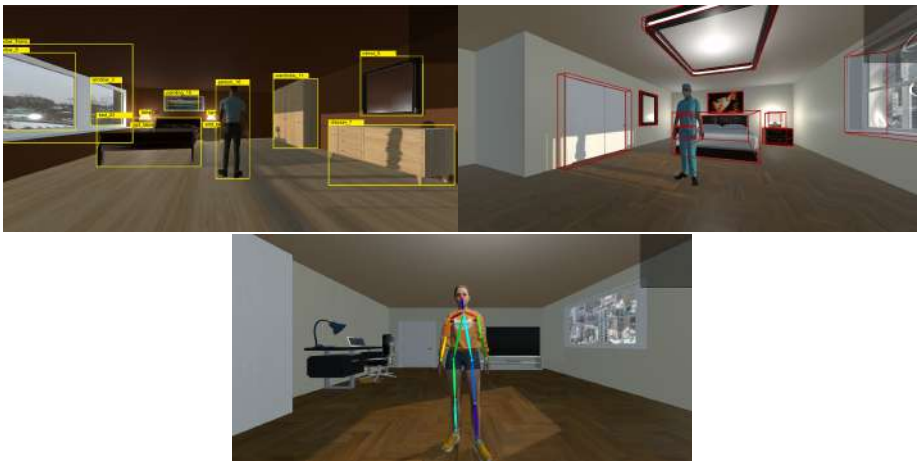


Figure 3: Examples of 2D and 3D bounding boxes (top) as well as human body key points (bottom).

Figure 4: Examples of images included in the *ActiveFace* datasset. Note that lower resolution images correspond to larger distances between the person and the camera.

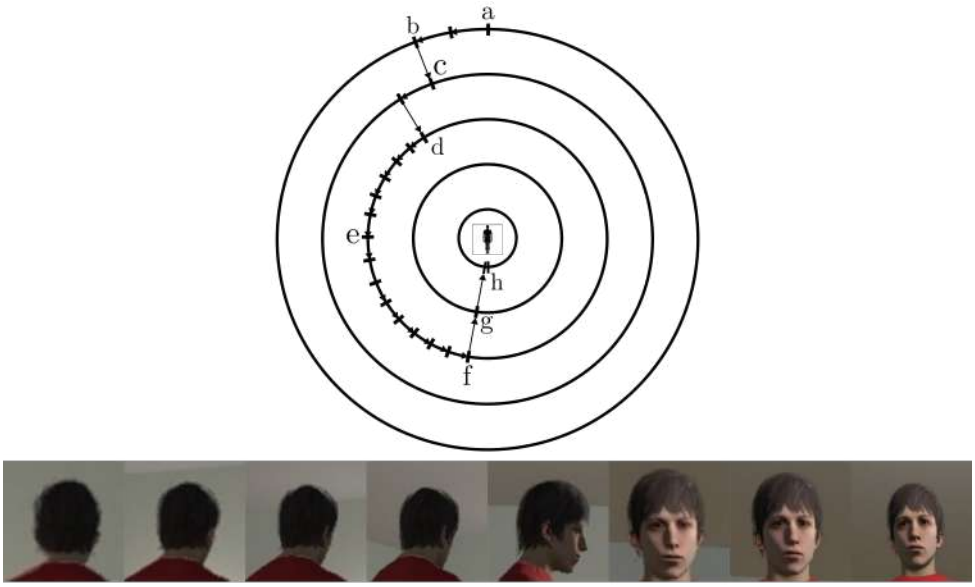
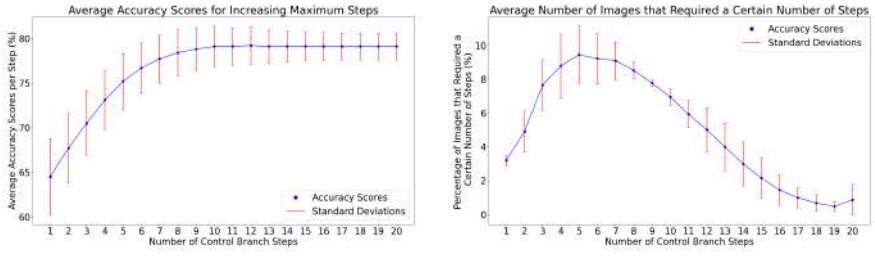


Figure 5: Example of control branch movements. Images from left to right correspond to the robot locations depicted on the diagram: (a) distance 3m, angle 180°; (b) distance 3m, angle 160°; (c) distance 2.5m, angle 160°; (d) distance 2m, angle 140°; (e) distance 2m, angle 90°; (f) distance 2m, angle 10°; (g) distance 1.5m, angle 10°; (h) distance 1m, angle 0°.



(a) Average accuracy score when increasing maximum allowed control branch steps

(b) Average number of images that required n steps, where $n = 1, 2, ..., 20$, before making a prediction.

Figure 6: Evaluating the impact of number of control steps on average accuracy score (Fig. 6a), as well as showing the distribution of control steps needed in order to acquire the best possible view (Fig. 6b).