ActiveFace: A Synthetic Active Perception Dataset for Face Recognition

Charalampos Georgiadis, Nikolaos Passalis, Nikos Nikolaidis
Department of Informatics, Faculty of Sciences
Aristotle University of Thessaloniki
Thessaloniki, Greece
{georgicd, passalis, nnik}@csd.auth.gr

***Abstract*—Active vision aims to enhance the efficiency of computer vision methods by enabling the capturing sensor, usually placed on a robot or, more generally, an autonomous system, to dynamically adjust its viewpoint, position or parameters in real-time. This capability allows for more precise decision-making by the model. However, training and evaluating an active vision model often necessitates a substantial number of annotated images, which must be captured under various sensor and environmental settings. These diverse images enable the model to learn the underlying dynamics of the active perception process. Unfortunately, collecting and annotating such datasets is a challenging and expensive task. It involves not only providing hand-crafted ground truth annotations but also ensuring that actions, such as moving around / towards / away from a person, are properly "imitated" to enable active vision approaches to model the corresponding active perception dynamics. To address these limitations, in this paper we propose a synthetic facial image generation pipeline specifically designed to support active face recognition, developed using a highly realistic simulation framework based on Unity. The developed pipeline allows for the generation of facial images for a set of persons at various view angles, distances, illumination conditions, and backgrounds. We demonstrate the effectiveness of our approach by training and evaluating a recently proposed embedding-based active face recognizer, as well as extending it to perform 2 axis control, leveraging the additional information provided by the generated dataset. To facilitate replication and encourage the use of the generated dataset for training and evaluating other active vision approaches, we also provide the associated assets and the developed dataset generation pipeline.**

***Index Terms*—Active Vision, Synthetic Data Generation, Unity, Face Recognition**

I. Introduction

Active vision is a field of computer vision that was inspired by our ability to move around in our environment in order to get a better view or understanding of our surroundings. Active vision aims to improve the efficiency of traditional computer vision methods by allowing the capturing sensor(s), placed on an autonomous system such as a robot, to alter their viewing position, direction or parameters in real-time so that a model can make better decisions about the subject of interest. Models that employ this approach can be used in a multitude of computer and robotic vision tasks, such as face and object recognition/detection, human pose/posture estimation and have been shown to be faster [1], smaller [1], [2] and more accurate [1], [2], [11], [12] than models that utilize a static approach.

While there is an abundance of datasets containing millions of annotated images that are being used in numerous computer and robotic vision tasks, datasets suitable for active vision problems are rather limited. Training and evaluating an active vision model often requires the use of hundreds of thousands, or even millions of annotated images captured at various sensor and environmental configurations in order to learn the dynamics that govern the active perception process. At the same time, collecting and annotating such datasets is a difficult and costly process that involves not only providing handcrafted ground truth annotations but also ensuring that actions such as moving around or towards a person, are appropriately modeled or imitated. Two notable examples of datasets that can support active vision tasks are the ModelNet [3] and Active Vision [4] datasets which can both be used in active object detection and recognition. ModelNet consists of more than 150,000 3D CAD model images at various angles for 161 object categories, while the Active Vision Dataset is comprised of more than 30,000 RGBD real-world images for 15 different scenes, accompanied by more than 70,000 2D bounding box annotations. In terms of face detection/recognition tasks, there have been instances where active vision models were trained using smaller datasets, such as the HPID [5], or Head Pose Image Database, dataset [1]. However, these datasets are rather limited in size. As an example, the HPID dataset consists of merely 2,790 facial images, significantly restricting, as a result, the use of such models in practice.

In recent years, there have been a few attempts towards synthetic annotated dataset generation frameworks that have been used in various computer and robotic vision tasks, including active vision. Two notable examples are BlenderProc [6], an open-source extension of Blender [7] that provides a modular procedural pipeline able to generate realistic annotated synthetic images, and Nvisii [8], which allows the generation of synthetic path-traced photorealistic images with the ability to produce metadata such as 2D/3D bounding boxes, segmentation masks, optical flow vectors, etc. However, these frameworks lack in terms of simulation realism and/or physics compared to more recent engines, e.g., Unity's Perception package [13].

Another valid approach for training and testing active vision methods is the use of photorealistic simulators for autonomous systems, robots or embodied AI such as Habitat-Sim[1]. For example, a real-time active vision humanoid soccer robot was trained and evaluated in a simulation environment by Khatibi et al. [10] using deep reinforcement learning, demonstrating how using a simulation environment can indeed be very effective in active vision tasks. However training and testing of such algorithms upon appropriate image/ video datasets is often easier, at least at the first stages of algorithm development, since it spares the handling of the robot motion.

Motivated by the aforementioned observations, in this paper, we introduce a realistic synthetic facial image generation pipeline, using a modified version of Unity's Perception package [13] installed on a URP project, that has been designed to support active face recognition. The developed pipeline enables generating images under a wide range of different view angles and distances, as well as under different illumination conditions and backgrounds. Some examples of the synthetic images contained in the generated dataset, called *ActiveFace*, are shown in Fig. 1. Furthermore, we also employ and extend an embedding-based active face recognizer [1] to demonstrate the usefulness of the generated dataset. Using the proposed extension it is possible to provide 2 axis control for active face recognition, going beyond the initial method and achieving superior results. The dataset, the associated assets and the dataset generation pipeline are publicly available at https://github.com/opendr-eu/datasets in order to allow anyone to seamlessly replicate them, as well as use the generated dataset for training and evaluating other active vision approaches.

The rest of the paper is structured as follows. First, the dataset generation process is introduced in Section II along with a detailed description of how the proposed dataset is generated. Subsequently, the active vision method used to evaluate the facial image dataset is presented in Section III, followed by an extensive experimental evaluation provided in IV. Finally, conclusions are drawn and suggestions for further development of the proposed dataset generation pipeline are discussed in Section V.

II. Dataset Description

The dataset generation process can be described by a sequence of nested for-loops, as shown in Algorithm 1. For a given set of possible environments $\mathcal{E}$, human models $\mathcal{H}$ and lighting conditions $\mathcal{L}$ the algorithm iterates over all possible combinations and captures different views by varying the camera angle and distance from the given human. The generated dataset was constructed using free environmental assets and human models downloaded from the Unity Asset Store, Maximo and Turbosquid, as well as human models created using MakeHuman. The Unity Perception package project was set up in such a way that anyone can add a new environment or human without requiring any change to the scripts that are responsible for changing the environmental or sensor configurations of each captured image (randomizer).

[1]https://github.com/facebookresearch/habitat-sim

Algorithm 1 : Dataset generation algorithm.

```
for each environment $E$ in $\mathcal{E}$ do
  for each human $H$ in $\mathcal{H}$ do
    for each lighting condition L in $\mathcal{L}$ do
      for each camera position $P$ from $[1m - 4m]$ in increments of $0.5m$ do
        for each camera angle $\Theta$ 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
```

First, we captured 1600×900 images of every valid combination of 8 environments, 33 humans, 4 lighting conditions, 7 camera distances and 36 camera angles, where the human is always at the center of the image, as shown in Fig. 2. A combination is rendered invalid if the camera collided with another object. The aim is to emulate robot motion in all possible (and "allowable", see below) locations around a human, under various lighting conditions, thus simulating different times of a day, in realistic environments. The environments are meant to simulate various rooms (living room, bedroom, kitchen) and include furniture such as tables, chairs, beds etc. Due to the existence of these furniture, not all locations in a room are accessible by the camera-equipped robot. Thus, the dataset does not contain images from such inaccessible locations, occupied by furniture.

Out of the 33 humans, 17 are females (1 infant and 16 adults), while the remaining 16 are males (1 infant and 15 adults). After capturing the initial images, the final dataset was created by cropping only the face regions. A total of of $175,428$ RGB facial images were created through this procedure, as shown in Fig. 1.

III. Active Face Recognition

In this work we build upon the embedding-based active face recognition method presented in [1] in order to a) extend it by supporting control in one additional axis and b) evaluate the proposed dataset generation pipeline and demonstrate its effectiveness. 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. This is achieved with the use of a trainable controller which, when given an image $\mathbf{x}^{(t)}$ at a time t, dictates the controller 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. $a_t = g_{\theta_c}(\mathbf{x}^{(t)})$ represents the trainable controller, where θ_c denotes a set of trainable action parameters.

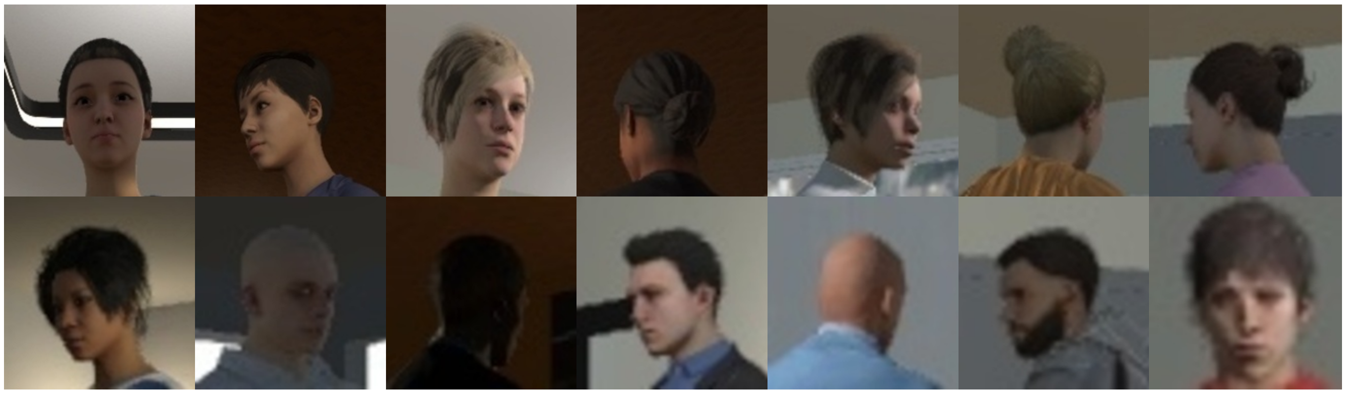


Fig. 1: Examples of images generated using the proposed approach. Note that lower resolution images correspond to larger distances between the person and the camera.

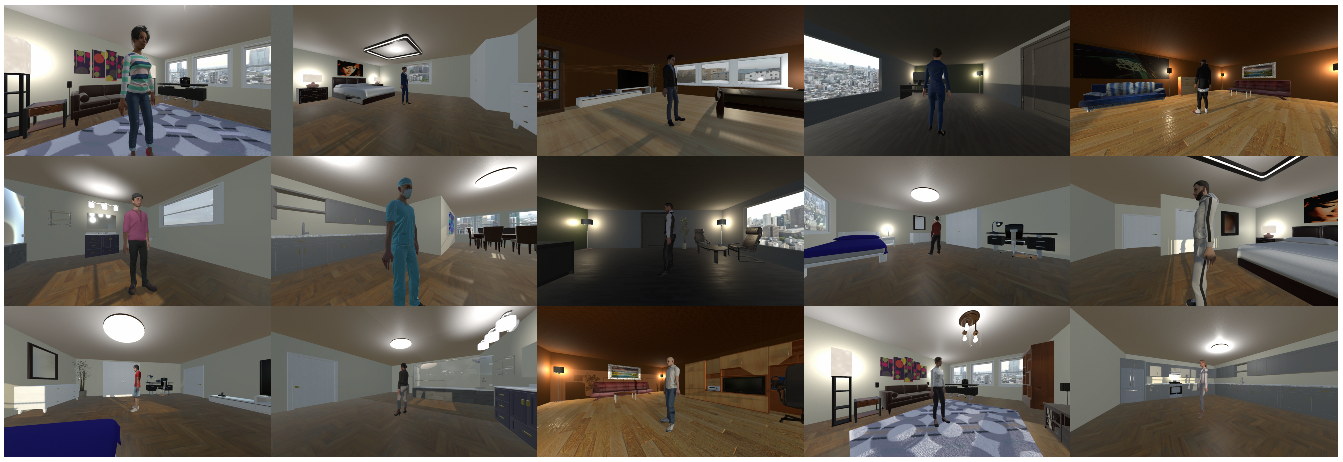


Fig. 2: Raw generated RGB image examples (before performing cropping).

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 [1]. 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:

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

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:

$$L_g = \sum_{i=1}^{N} L_x(g_{\theta_c}(\mathbf{x}_i), d_i^{(a)}), \tag{3}$$

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

$$L_f = \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} L_e(f_{\theta_r}(\mathbf{x}_i), f_{\theta_r}(\mathbf{x}_j), d_{ij}), \tag{4}$$

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 [1], 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 $\sqrt{m}$:

$$L_e(\mathbf{y}_i, \mathbf{y}_j, d_{ij}) = d_{ij}||\mathbf{y}_i - \mathbf{y}_j||_2^2 + \\ (1 - d_{ij}) \max(0, m - ||\mathbf{y}_i - \mathbf{y}_j||_2^2), \tag{5}$$

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 (3) and (4):

$$L = L_g + L_f \tag{6}$$

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, in a circle centered at the human subject. Since the classes involved in (3) 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.

IV. 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 facial image 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.

A. Experimental Setup

The active vision model was evaluated on both the entire 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, 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 dataset (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 datasets. 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 datasets. 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 datasets 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.

B. Experimental Results

The evaluation results are shown in Tables I, II and III. As a reninder, **Set 1** represents the full facial image dataset, while **Set 2** denotes the dataset with the reduced pan range.

TABLE I: Static vision model evaluation.

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

Evaluation results for the static model are presented in Table I. Clearly, the models perform best when trained for 5 epochs, reaching accuracy scores of $\sim$51.1% and $\sim$60.3% for **Set 1**

TABLE II: Active vision model evaluation **with** the additional `Front` movement/action.

Model	Accuracy (Set 1)	Accuracy (Set 2)
Active (10 epochs)	$67.9 \pm 6.8\%$	$76.9 \pm 6.5\%$
Active (20 epochs)	$69.2 \pm 7.6\%$	$79.1 \pm 1.7\%$
Active (30 epochs)	$67.4 \pm 8.6\%$	$78.3 \pm 6.8\%$

TABLE III: Active vision model evaluation **without** the additional `Front` movement/action.

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

and **Set 2**, respectively. Increasing the number of epochs seems to cause an overfit of the model on the training data.

Once the active approach is employed, a substantial increase in prediction accuracy can be observed for both datasets by a maximum of $\sim$18.1% and $\sim$18.8%, respectively, as seen in Table II. 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 III) the model is $\sim$8.9% and $\sim$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 3 depicts an example of how the 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.

V. Conclusions

In this paper, a synthetic realistic data generation pipeline for training and evaluating active face recognition methods using Unity's Perception package was introduced. One dataset was generated using this pipeline, consisting of face images, cropped from the originally captured 1600×900 images. The facial images were used to train and evaluate both a static and active vision embedding-based face recognizer in order to demonstrate how employing an active approach can significantly increase the prediction accuracy of a static model. The experimental results showed an substantial increase in recognition performance when comparing the static to the active model by a maximum of $\sim$ 18.8%. Additionally, we demonstrated that, by enabling the robotic system to move towards the subject, thus not being limited to left or right movements as in [1], yields better recognition accuracy scores by a maximum recorded difference of $\sim$ 8.9%. One limitation of this work, however, was that the trained model could not be evaluated on real faces, since, to the best of our knowledge, no face image dataset that can support left, right and front movements in equally-sized increments exists.

The proposed data generation pipeline allowed for the fast generation of an annotated realistic synthetic dataset which can be successfully used for training and evaluating face recognition methods, including active ones. However, there are numerous ways in which the presented pipeline can be improved. First, the pipeline can be easily modified to generate a dataset depicting the entire environment including full-body depictions of the subjects and annotations such as bounding boxes for the subjects and furniture, semantic segmentation ground truth etc. Such a dataset is indeed in the final stage of construction. Moreover, the inclusion of additional environments, both indoors and outdoors, as well as the ability to manipulate other lightning sources apart from each environment's directional light would increase the number of training samples as well as the image variety of the dataset. Additionally, a more advanced randomization approach that can place objects in front of the human subject could allow for more robust models that can deal with occlusions. Also, the dataset could be extended in order to be usable for active human pose or activity recognition by enabling the Unity project's options for playing and switching between different animations. Furthermore, the Unity project could be converted to Unity's HDRP which would allow for much more realistic and high quality rendered images in 2K or even 4K resolution. Finally, in terms of improving the active vision method itself, the implementation of obstacle avoidance may result in both higher prediction accuracy and a more robust model.

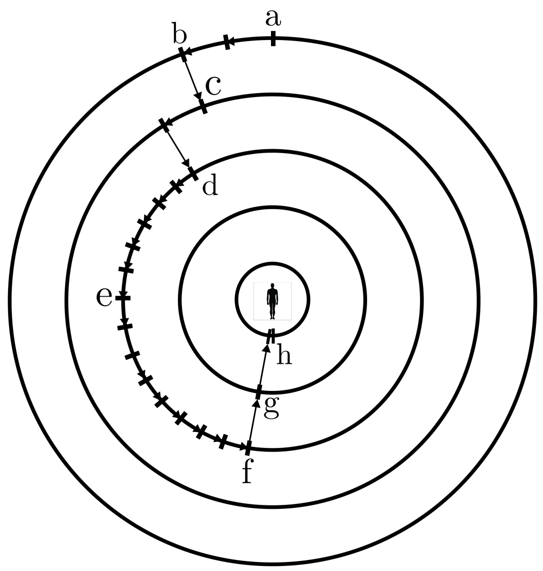
Acknowledgment

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

[1] N. Passalis and A. Tefas, "Leveraging Active Perception for Improving Embedding-based Deep Face Recognition," Proceedings of the IEEE 22nd International Workshop on Multimedia Signal Processing, 2020.

[2] N. Pan, R. Zhang, T. Yang, C. Cui, C. Xu and F. Gao, "Fast-Tracker 2.0: Improving autonomy of aerial tracking with active vision and human location regression", IET Cyber-Systems and Robotics", vol. 3, no.4, pp. 292-301, 2021.



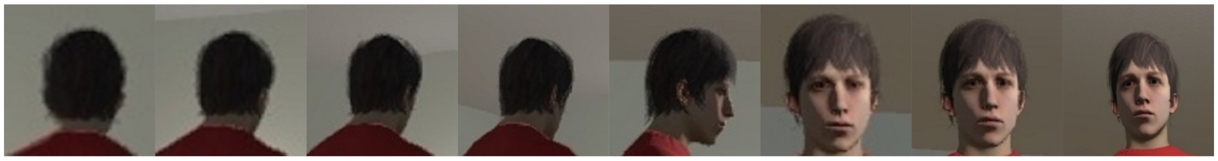


Fig. 3: Example of control branch movements. Images from left to right correspond to the robot locations depicted on the diagram: (a) distance $3m$, angle 180°; (b) distance $3m$, angle 160°; (c) distance $2.5m$, angle 160°; (d) distance $2m$, angle 140°; (e) distance $2m$, angle 90°; (f) distance $2m$,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

[3] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao, “3D ShapeNets: A Deep Representation for Volumetric Shapes”, Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[4] P. Ammirato, P. Poirson, E. Park, J. Kosecka and A. C. Berg, “A Dataset for Developing and Benchmarking Active Vision”, Proceedings of the IEEE International Conference on Robotics and Automation, 2017.

[5] N. Gourier, D. Hall, J. L. Crowley, “Estimating Face Orientation from Robust Detection of Salient Facial Features”, Proceedings of Pointing, ICPR, International Workshop on Visual Observation of Deictic Gestures, Cambridge, UK, 2004

[6] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, H. Katam and H. Teja, “Blenderproc, “arXiv preprint arXiv:1911.01911”, 2019.

[7] B. O. Community, “Blender - a 3D modelling and rendering package”, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

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

[9] Borkman, A. Crespi, S. Dhakad, S. Ganguly, J. Hogins, Y.-C. Jhang, M. Kamalzadeh, B. Li, S. Leal, P. Parisi, et al., “Unity perception: Generate synthetic data for computer vision,” arXiv preprint arXiv:2107.04259, 2021.

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

[11] Kakaletsis, Efstratios, and Nikos Nikolaidis. “Using Synthesized Facial Views for Active Face Recognition.”, Machine Vision and Applications, accepted, 2023.

[12] P. K. Murali, A. Dutta, M. Gentner, E. Burdet, R. Dahiya and M. Kaboli, ”Active Visuo-Tactile Interactive Robotic Perception for Accurate Object Pose Estimation in Dense Clutter,” in IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 4686-4693, April 2022, doi: 10.1109/LRA.2022.3150045.

[13] Unity Technologies, ”Unity Perception Package”, 2020, https://github.com/Unity-Technologies/com.unity.perception.