Follow Me: real-time in the wild Person Tracking Application for Autonomous Robotics

Thomas Weber, Sergey Triputen, Michael Danner, Sascha Braun, Kristiaan Schreve, and Matthias Rätsch

Reutlingen University, 72762 Reutlingen, Germany,
{thomas.weber, sergey.triputen}@reutlingen-university.de,
WWW homepage: https://www.reutlingen-university.de/home/

Abstract. In the last 20 years there have been major advances in autonomous robotics. In IoT (Industry 4.0), mobile robots require more intuitive interaction possibilities with humans in order to expand its field of applications. This paper describes a user-friendly setup, which enables a person to lead the robot in an unknown environment. The environment has to be perceived by means of sensory input. For realizing a cost and resource efficient *Follow Me* application we use a single monocular camera as low-cost sensor. For efficient scaling of our Simultaneous Localization and Mapping (SLAM) algorithm, we integrate an inertial measurement unit (IMU) sensor. With the camera input we detect and track a person. We propose combining state of the art deep learning with Convolutional Neural Network (CNN) and SLAM algorithms functionality on the same input camera image. Based on the output robot navigation is possible. This work presents the specification, workflow for an efficient development of the *Follow Me* application. Our application's delivered point clouds are also used for surface construction. For demonstration, we use our platform *SCITOS G5* equipped with the afore mentioned sensors. Preliminary tests show the system works robustly in the wild. [1]

Keywords: Mobile Robotics, 3D Perception, Navigation, Human-Robot Interaction, Person Tracking, Machine Learning, CNN, SLAM

1 Introduction

1.1 Motivation

Robotics navigation and control is a complex task to deal with. Robust detection and tracking of people in autonomous robotics applications is a key task nowadays. It is also essential to intuitive human machine interaction. Our aim is providing an easy to implement, flexible and adaptable application for this task. One of our main goals is a robust application for in the wild usage. For head detection, person tracking and robot navigation there are several projects

[1] This work is partially supported by a grant of the BMBF FHprofUnt program, no. 03FH049PX5

providing great and useful solutions. This work combines them to an affordable complete system that performs very well. The paper is structured as follows: First, we describe in a broad overview how our application works. Next, we dive deeper into theory, our approaches and techniques used in the application and state further work and possible enhancements on these approaches. This is followed up by instructions for implementation, hardware and our robot platform *SCITOS G5*, produced by MetraLabs GmbH. Finally, we show our application in early experiments.

1.2 Related Work

Different approaches on person detection and tracking involving autonomous robotics were proposed over time, such as [2, 18]. However, they often lack simplicity and require multiple sensory inputs. For sensory input, we opt for using only a single low-cost consumer monocular web cam, in our case the *Sony PlayStation Eye Camera* and a common IMU chip package, a *Bosch BNO055*. This way our application can be deployed on existing setups inexpensively. The approach of the head detector used in our work is inspired by the success of deep Convolutional Neural Networks (CNN). First successfully introduced by Krizhevsky [15] by winning the LSVRC-2012 ImageNet challenge, CNNs dominate in all vision task challenges today. Convolutional Neural Networks is a machine learning approach for computer vision tasks and were already used for head detection by [26]. Our detector is based on the Single Shot Detector (SSD) from Wei Liu [16], which is more advanced than Faster R-CNN [21]. Our internal research group is involved in SLAM [22] and face recognition experiments, so we have knowledge transfer from our work to other projects.

1.3 Purposed Solution

The main steps in our workflow to realize the *Follow Me* application are shown in fig. 1: First, we take a 2D camera image frame and use it for head detection via a CNN and create bounding boxes. We implement calibration and scaling to real-world dimensions for LSD-SLAM algorithm on embedded hardware *NVIDIA Jetson TK1 developer kit* by utilizing a *Bosch BNO055* IMU. This setup allows us to generate real-world scaled point clouds. The point cloud is reduced to a volume of interest including only the tracked head. To create a surface out of these points we use a Poisson surface reconstruction algorithm. We propose using the results of those processes to construct the points for the head and calculate the head position with respect to the current robot position. We transform the CNN based 2D head detection to 3D world space coordinates. To improve the world space coordinates, we show a workflow to fuse the SLAM and CNN estimated position and track a person. Finally, the positional data is sent to motion controller feedback algorithms. The main contribution of this work is a robust *Follow Me* application based on deep learning CNN, SLAM and a 3D morphable face model (3DMM) [11]. We observe that a fusion of 2D based and 3D SLAM based world space coordination improves the estimation between the robot and the person. The core novelties of the paper are: (1) *Follow Me* using a single monocular low-cost webcam (2) a CNN based surround head detection; this means the occiput is also recognized (3) Taking advantage of scaled SLAM with IMU to improve distance estimation and fusion (4) Robust tracking of person instances using 3D points and 2D transformation fused.

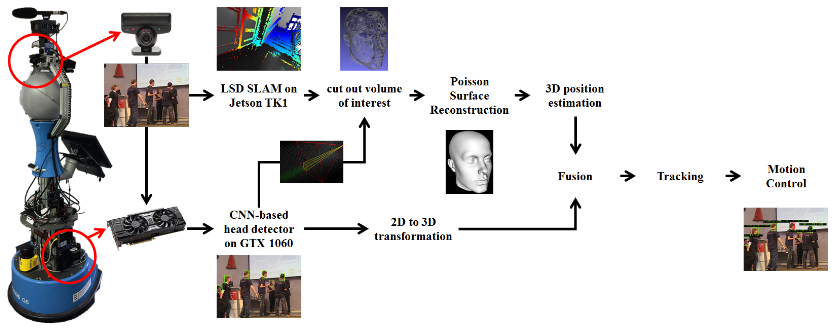

Fig. 1: Workflow of *Follow Me*: 2D image used for parallel CNN head detection and LSD-SLAM point cloud generation. Point cloud reduced to volume of head based on CNN bounding box. 3D surface reconstruction and 2D transformation for head world space coordinates. Fusion of SLAM and CNN head position. Tracking and motion control.

2 2D CNN-based and 3D SLAM-based Person Tracking

2.1 Single Shot MultiBox Convolutional Neural Network based Head Detector

Our detector CNN uses an SSD approach from Wei Liu [16], which needs only a single forward feed for detection of heads on a 2D RGB image, recorded with a low-cost webcam. Several feature maps with different scales are used. Each box predicts the shape and the confidence of the class. Based on this information the bounding boxes are estimated. The base for the network is a VGG-16 network [24]. The feature layers for the feature maps are connected to the end of the network. This results in a high improvement of the speed.

2.2 Training and Evaluation

For training the network the HollywoodHeads Dataset [1] is used. This dataset contains 224,740 video frames from Hollywood movies with annotations of bounding boxes of the heads. Some examples from the data set are shown in fig. 2 (a, b). The data set is split in a training set which contains 216,719 frames, a validation set which contains 6,719 frames and a test set, which contains 1,302 frames. This split is proposed in [26]. The *Caffe* framework [12] is used for training and deployment. The training was performed on a *GeForce GTX 1060 6GB* on *Ubuntu 14.04 LTS*. We achieve 59.7 % Mean Average Precision (mPA). Field tests show, that the SSD detector is stable, even under harsh conditions, as fig. 2 (c, d) shows.

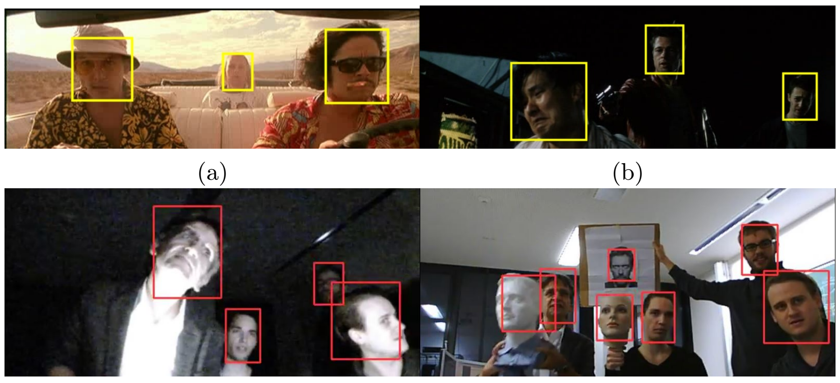
(a) (b) (c) (d)

Fig. 2: Example frames from the HollywoodHeads data set [1] (a, b) and detections results of our trained SSD network (c, d)

2.3 Coordination Transformation

The CNN detection takes place in 2D. The position of the head in world space coordinates plays a major role in robust tracking. Persons crossing can cause problems e.g. when people cross each other at different distances from the camera. To avoid this, a transformation of the detection coordinates in the 2D image into 3D world space coordinates is done. This improves the tracking and enables differentiating between crossing tracks in 2D image coordinates when using 3D world space coordinates. The knowledge of the size of the bounding box from SSD is used to estimate the distance of the head in real world coordinates. Therefore, a data series is recorded representing the distance in relationship to the area of the bounding box. Based on this data a function is estimated which approximates the relationship between the *area* of the bounding box in *pixels* and the distance in world space *cm*: $z_{world} = 15022 \cdot area^{-0.525}$. For the estimation of the x and y in world space, z_{world} is used in calculations $x_{size} = 2 \cdot z_{world} \cdot arctan(\alpha_{FoV}/2)$ and likewise $y_{size} = 2 \cdot z_{world} \cdot arctan(\alpha_{FoV}/2) \cdot 2/3$, which are depended on the used camera sensor parameters. Based on the *area*, relations of pixels in the image to the size in real world in cm are $x_{relation} = x_{size}/x_{camRes}$ and $y_{relation} = y_{size}/y_{camRes}$. The real world coordinates are calculated based on relations between pixels in the image to the size in real world and the image coordinates:

$x_{world} = x_{relation} \cdot x_{imagePos} - x_{size}/2$ and $y_{world} = y_{relation} \cdot y_{imagePos} - y_{size}/2$. Finally, we have a world space position $[x_{world},\ y_{world},\ z_{world}]$ with the camera sensor centre-point as origin.

2.4 SLAM-based 3D Head Extraction

Most positioning, orientation and odometry systems provide only pose data relative to the system start point. It is necessary to accurately define the initial position and scale before motion starts in order to integrate the SLAM system into other robot systems interacting with the environment. Several SLAM algorithms have been developed to build a map based on the key-frames with loop closure detection and to propose a way to identify the current location of the robot [4, 19, 20]. The map can be used to calculate the scale factor of the camera path. Similar results can be achieved by combining monocular camera data with other sensors [5, 8, 25]. There are many existing SLAM algorithms described today. Most of them are dealing with standard navigation tasks (e.g. building key-frame based maps, loop closure detecting or current position localization). One of the more popular approaches is PTAM [13,14] and its variations, such as LSD-SLAM [4, 7] and ORB-SLAM [19, 20]. A significant challenge for monocular SLAM systems is the problem of estimating scaled camera positions. Without scaled metric data, the SLAM output cannot be integrated with other robotic systems relying on accurate environmental information. That is why some approaches use RGB-D or stereo cameras [3, 17]. Depth maps generated from RGB-D cameras can be used to find scaled metric camera positions. Additionally, RGB-D hardware makes 3D environment reconstruction possible [3]. This approach is more expensive than monocular systems. However, scaled environmental information can be obtained if the monocular camera data is combined with other sensors, e.g. laser scanners, etc. The PTAM algorithm uses an altimeter based solution [6]. PTAM does not provide scaled 3D environmental data, though. The authors are not aware of a monocular SLAM system that provides a scaled point cloud representation of the environment aligned to some global coordinate system.

2.5 Spaces Alignment Problem and Solution

The main steps of LSD-SLAM and real-world scale and orientation synchronization are taken during the LSD-SLAM algorithm initialization period. We should remember, that LSD-SLAM systems without spacial equipment (such as an external IMU sensor) have no information about the real-world camera position, orientation and surrounding object sizes. The LSD-SLAM camera position and orientation estimating process is based on tracking special points in a particular frame. It is necessary to have depth information for them. Initially, it uses random numbers for the depth hypotheses, i.e. $d \in [0,\ 1]$ random numbers for each feature point in the first frame. Also, for the first frame camera position and orientation the initial position vector $[0,\ 0,\ 0]$ and quaternion $[1,\ 0,\ 0,\ 0]$ are used. This random distance is used for camera tracking and for defining the estimated camera position and orientation. At the next step the calculated camera position and orientation error is used in the triangulation task to update the point cloud depth information. After some iterations, the calculation results converge into an optimal combination of depths and camera tracking data. It is obvious that after the initialization period the estimated point depths and camera position and orientation data have unpredictable values. The initialization time and final distance scale factor depend on a random distribution of initial hypothetical depths. Now the task can be formulated: we are looking for a combination of rotation, scaling and translation factors that will map LSD-SLAM estimated coordinate system space to the IMU sensor coordinate system. It is very important to have the transformation from LSD-SLAM space to the sensor space, because the IMU sensor coordinate system is aligned to the gravity vector. That means it is aligned to the world coordinate system. It makes it possible to use an octree optimized 3D model representation for the estimated point cloud. In order to find a solution, first of all, it is necessary to formally define the input data. LSD-SLAM estimates data as a set of key-frames which describe camera position and orientation with a set of points or coordinates (or estimated distance from the camera to each feature point). This will be denoted as $\mathbf{L} = \{T,\ \mathbf{P}\}$ where $T \in \mathbb{Sim}(3)$ such that: $T = (s\mathbf{R}\ \mathbf{t},\ 0\ 1)$ with $\mathbf{R} \in \mathrm{SO}(3),\ \mathbf{t} \in \mathbb{R}^3,\ s \in \mathbb{R}^+$. The $\mathbb{Sim}(3)$ group represents transformation and scaling in a three dimensional space. Here s is a scale factor and $\mathbf{t}$ a translation vector in 3D space. $\mathbf{R}$ is an element of the SO(3) group that represents rotation. Each element of the SO(3) group can be represented in several different forms (e.g. rotation matrix, quaternion or vector and angle combination). T represents the camera transformation estimated by LSD-SLAM. Furthermore $\mathbf{P} = \{\ \mathbf{p} = \langle\ u, v, d\ \rangle \mid u \in \mathbb{Z}^+,\ v \in \mathbb{Z}^+,\ d \in \mathbb{R}^+\ \}$ where $[u,\ v]$ are the pixel coordinates on the frame and d is a distance from the camera position to the point. In order to calculate 3D coordinates of the points in a key-frame point cloud, it is also necessary to have the intrinsic camera matrix: $M = [f_x\ 0\ c_x,\ 0\ f_y\ c_y,\ 0\ 0\ 1]$ where: $\mathbf{f} = [f_x,\ f_y]$ - camera focus distance per axis. $\mathbf{c} = [c_x,\ c_y]$ - image position of the principle axis in pixel coordinates. Let $\mathbf{X} = \{\ \mathbf{x} \mid \mathbf{x} \in \mathbb{R}^3\ \}$ represent $\mathbf{P}$ in the 3D space of the LSD-SLAM coordinate system. It can be calculated by mapping: $\delta\colon \mathbf{P} \to \mathbf{X}$. Where δ is: $\mathbf{X} = \delta(\ \mathbf{L}, M\ ) = T\,[(u - c_x)\cdot d/f_x,\ (v - c_y)\cdot d/f_y,\ d,\ 1,\ ]$. We are receiving additional information from an external sensor, in this case an IMU. For LSD-SLAM estimated data, we will use set $\mathbf{S}$, which describes sensor measured data $\mathbf{S} = \{T^{'},\ \mathbf{P}^{'}\}$. For IMU sensor measured data $T^{'} \in \mathbb{Sim}(3)$ is a transformation into the real world metric coordinate system and $\mathbf{P}^{'} \equiv \varnothing$ is an empty set, because the IMU model do not have any information about the 3D real world geometry. Because $\mathbf{P}^{'}$ is empty, $\mathbf{X}^{'}$ cannot be obtained. As described in the previous subsection, we are looking for a combination of rotation, translation and scale conversions from $\mathbf{L}$ to $\mathbf{S}$. Formally that conversion combination can be represented as an element of $\mathbb{Sim}(3)$: $\Lambda = (s\mathbf{R}\ \mathbf{t},\ 0\ 1)$ with $\mathbf{R} \in \mathrm{SO}(3),\ \mathbf{t} \in \mathbb{R}^3,\ s \in \mathbb{R}^+$. Finally, we can formalize our task as follows: we are looking for a mapping $\Lambda\colon \mathbf{L} \to \mathbf{S}$. In order to find Λ we can only use the LSD-SLAM camera position and orientation and then relate it to the IMU sensor measurements. As already mentioned $\mathbf{P}^{'}$

(hence $\mathbf{X}^{'}$ as well) is an empty set for the sensor coordinate system. Therefore the equation for finding Λ becomes: $\Lambda^{*} := argmin_{(s,\mathbf{R},\mathbf{t})} \left(\sum \left\| T^{'} - \Lambda \times T \right\|^{2} \right)$ The element usage is based on the fact that T and $T^{'}$ are part of the $\mathbb{S}\mathfrak{im}(3)$. The solution to this equation is discussed in detail in [9,10]. And so, we are able to reconstruct the point cloud in the IMU sensor coordinate system. With the calculated Λ^{*}, we can transform the LSD-SLAM estimated point cloud to the IMU sensor coordinate system: $\mathbf{X}^{'} \approx \Lambda^{*} \times \mathbf{X} \approx \Lambda^{*} \times \delta(\mathbf{L},\ M)$

Inertial Measurement Unit Calibration Integrated in our IMU there are three gyroscopes and three accelerometers that are orientated along the three axis. We need to calculate continually the current position, orientation and track the motion of the robot for each of the six degrees of freedom $x, y, z, \theta_x, \theta_y, \theta_z$. A disadvantage of an IMU in general is that they suffer from accumulated error. In our experiments, it is a problem to calculate a linear acceleration in a defined time period and additionally there is a misalignment and non-orthogonality error. To improve the results of IMU sensors and correct misalignment, we strap the IMU onto a stepper motor and calibrate it in relation to different angles from 0° to 360° by tracking the sensor values of each step.

Extracting Point Cloud of the Head While moving we will get varying camera positions from where every frame is tracked. The first camera position defines a key frame; any further position delivers a current tracked frame. We extract feature points from the key frame to reconstruct the point cloud and identify the feature points on the tracked frame. This progress is supported by camera calibration parameters which are estimated by the camera position and orientation. These calibration parameters describe a calculated view of the frame, in which we use image and head detection for the current tracked frame. The whole camera volume is cropped to a volume of interest where the head is positioned and we use the z distance of each point to reduce the point cloud to an extraction of the head. In summary, we propose combining the camera calibration parameters, all the data from LSD SLAM (position and orientation) and the bounding box of head detection CNN to get our volume of interest in order to reduce unnecessary information and get an extracted point cloud, as shown in fig. 3.

2.6 Poisson Triangulation based Estimation of the Head Position

There are several methods for surface reconstruction. Some of these can be divided into global and local fitting procedures. Global fitting procedures are in implicit form and can be represented as the sum of radial basis functions (RBFs) at the centre of the points. The RBFs are not linked to the data points but to the surrounding space. Local fitting techniques use the distance from tangent planes to the next point. The Poisson surface reconstruction combines both methods. It uses the Poisson equation, which is also used in other applications of physics.

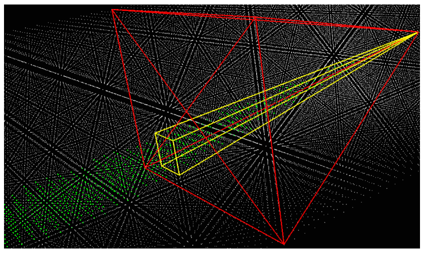

Fig. 3: Simplified example: Camera frustum (red), volume of interest by CNN bounding box (yellow), points in volume of interest (green), other points in grey

The algorithm used here aims to reconstruct a watertight model. Experiments shows a *search radius* of less than 2 generates dents in the surface from volume of interest. The optimum seems a radius of 4. Results of the Poisson Triangulation method are feasible. It is possible to produce a waterproof mesh. Large holes can be closed in the process, like in fig. 4. We propose applying real-time 3D

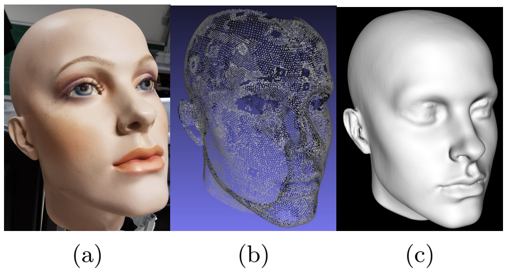

Fig. 4: Demonstration of a surface reconstruction (c) using the Poisson method on a point cloud (b) generated from a dummy head (a)

face fitting via the *Surrey face model*'s first component, the 3D morphable PCA shape model, and its landmark annotations onto the generated mesh. Further information are available in [11].

2.7 Fusion of Positional Data

From the fitted 3DMM, we select the nose tip landmark as coordinate for the tracking z position. In case the occiput is captured, in our first implementation we accept a suboptimal fitting, since the nose tip landmark is still on the surface and therefore a good estimate of the head position. We advocate a simple fusion of the SLAM 3D position from 3DMM fitting landmark P_{SLAM} and the CNN 2D transformed world space position P_{CNN} in form of: $P_{fused} = \alpha \cdot P_{SLAM} + \beta \cdot P_{CNN}$ with $P_i = [x,\ y,\ z]$ and $\alpha + \beta = 1$. P_{fused} is input to a Kalman filter described in section 2.8. Optimal values for α and β depend on actual hardware implementation and should be determined in experiments.

2.8 Global Nearest Neighbour (GNN) Tracking

After the detection in 2D and 3D, the fused world coordinates of the head are input to a tracking algorithm. For single object tracking a combination of a Kalman filter and global nearest neighbour data assignment is used. This can extend it to a state of the art joint probabilistic data-association filter algorithm [23], which enables a robust multi-object tracking. In our first implementation, a single track of the GNN tracking algorithm is used, which is sufficient. Every iteration the tracker receives the detections from the SSD detector CNN transformed in 3D coordinates. First the tracker needs to be initialized. Therefore, a tracker object must be created. For this purpose, a detection is selected, which should be tracked. This can be externally requested from the user. The tracker object initializes a Kalman filter [27] with the coordinates of the selected detection. The state transition function of the Kalman filter is based on Newton's equations of motion with a constant acceleration of the system. For implementation *FilterPy*, a Kalman filtering and optimal estimation library in Python is used. The Kalman filter predicts the position of the tracked object at the next time step based on the process model. The Gaussian probability is calculated between the predicted position and the current detections. To limit this range a gating is applied. The size of the gate increases with increasing covariance of the Kalman filter. The detection with the highest probability inside the gate is assigned to the tracker. The tracker uses the detection to update the Kalman filter. A live counter for the tracked head is updated based on whether a detection could be assigned to the tracker or not. This process is iteratively repeated until the track is lost or a new track is requested.

3 Implementation and Testing

The following describes the hardware and software specifications and technology for the person tracker and their implementation. The hardware used in for the application is an *Intel i7 2620m* with 8 GB of RAM and *GeForce GTX 1060* with 6 GB VRAM. For image capturing a *Sony PlayStation Eye Camera* is used, with a framerate of 60 frames per seconds and low motion blur. The resolution of 640x480 satisfies the required resolution for the detector CNN. The software implementation is written completely in *Python* and takes place on *Ubuntu 14.04 LTS*. For array implementations and operations, we use *Numpy*. The SLAM part runs on a *NVIDIA Jetson TK1 developer kit* with a *Bosch BNO055* IMU.

User Interface: The user interface of the head tracker is implemented with *OpenCV*. The user interface shows the current captured scene. All detections are marked with bounding boxes and their position in world space coordinates. To trigger a tracking request, a user simply clicks inside the target bounding box. Alternative implementation may allow for voice or gesture based selection of a bounding box by ID. The tracked head is marked with a different colour, to distinguish the detection bounding boxes. Figure 5 shows the implemented user interface.

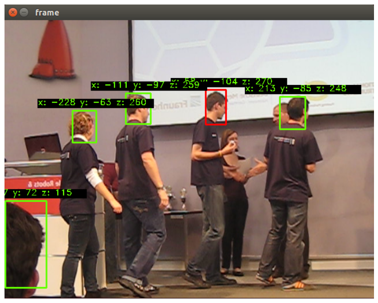

Fig. 5: User interface of the head tracker. Detection bounding boxes in green, tracking target in red. Displayed over each is the head position in world space coordinates in relation to the camera sensor.

Robot Motion Control: The final positional data P from the Kalman filter prediction is sent to the robot's motion controller feedback algorithm, which calculates rotation angle and distance to point P with respect to the robot's coordinate space. The feedback algorithm is set to keep a fixed offset to the target.

Testing: Figure 6 depicts a video image sequence. In the upper left corner, the User Interface is overlaid. The scenery is shot in third person to provide a holistic view of the demonstration. In the image sequence, several persons cross and occlude the tracked person in front of the robot in between person and robot and behind the person. Notice the tracking stays on the target person.

4 Conclusion

We have implemented an early working *Follow Me* application with the potential to achieve better results than existing solutions which use more expensive sensors. Our system is still work in progress, though. Further tasks aim to improve stability in the wild, while keeping low budget hardware. With SLAM, we can create a map of the environment to improve navigation and obstacle avoidance. We suggest upgrading the workflow with the use of blend shapes in the 3D morphable face model. Blend shapes are simply described by the difference between two three dimensional objects. Those objects represent human faces

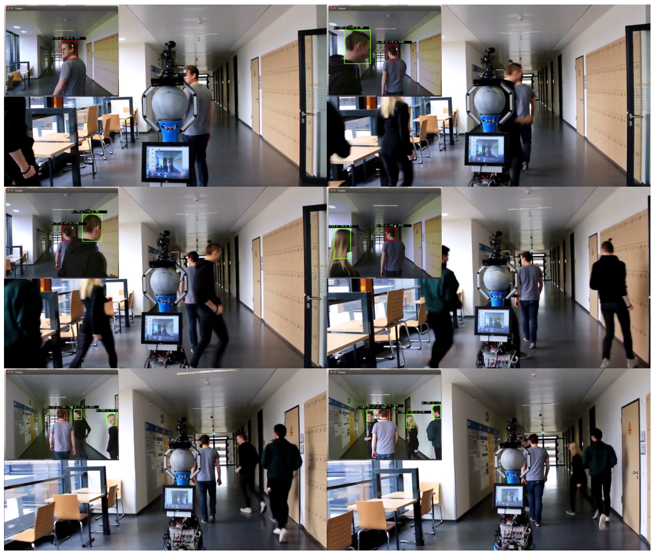

Fig. 6: Video image sequence of the *Follow Me* application

with different expressions. After the registration of the face with the method of a 3D morphable model (3DMM) [11] we can do further normalization via blend shapes and eventually enable a more stable face recognition.

References

1. Hollywoodheads dataset. `http://www.di.ens.fr/willow/research/headdetection/`, accessed: 2017-02-01
2. Eisenbach, M., Vorndran, A., Sorge, S., Gross, H.M.: User recognition for guiding and following people with a mobile robot in a clinical environment. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS). pp. 3600–3607. IEEE (2015)
3. Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., Burgard, W.: An Evaluation of the RGB-D SLAM System. Robotics and Automation (2012)
4. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-Scale Direct Monocular SLAM. European Conference on Computer Vision (ECCV) (2014)
5. Engel, J., Stueckler, J., Cremers, D.: Large-Scale Direct SLAM with Stereo Cameras. International Conference on Intelligent Robots and Systems (IROS) (2015)
6. Engel, J., Sturm, J., Cremers, D.: Camera-Based Navigation of a Low-Cost Quadrocopter. International Conference on Intelligent Robot Systems (IROS) (2012)
7. Engel, J., Sturm, J., Cremers, D.: Semi-Dense Visual Odometry for a Monocular Camera. IEEE International Conference on Computer Vision (ICCV) (2013)
8. Engel, J., Sturm, J., Cremers, D.: Scale-Aware Navigation of a Low-Cost Quadrocopter with a Monocular Camera. Robotics and Autonomous Systems (RAS) (2014)
9. Horn, B.: Closed-form solution of absolute orientation using unit quaternions. Journal of the Optical Society of America (1987)

10. Horn, B., Hilden, H., Negahdaripour, S.: Closed-form solution of absolute orientation using orthonormal matrices. Journal of the Optical Society of America (1988)
11. Huber, P., Hu, G., Tena, R., Mortazavian, P., Koppen, W.P., Christmas, W., Rätsch, M., Kittler, J.: A multiresolution 3d morphable face model and fitting framework, http://www.patrikhuber.ch/files/3DMM_Framework_VISAPP_2016.pdf
12. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
13. Klein, G., Murray, D.: Parallel Tracking and Mapping for Small AR Workspaces. In Proc. International Symposium on Mixed and Augmented Reality (ISMAR'07, Nara) (2007)
14. Klein, G., Murray, D.: Parallel Tracking and Mapping on a Camera Phone. In Proc. International Symposium on Mixed and Augmented Reality (ISMAR'09, Orlando) (2009)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012), http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
16. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: ECCV (2016)
17. Maier, R., Sturm, J., Cremers, D.: Submap-based Bundle Adjustment for 3D Reconstruction from RGB-D Data. In German Conference on Pattern Recognition (GCPR) (2014)
18. Mueller, St., Schaffernicht, E., Scheidig, A., Boehme, H.J., Gross, H.M.: Are you still following me? In: Europ. Conf. on Mobile Robots (ECMR). pp. 211–216. Albert-Ludwigs-Universitaet Freiburg – Universitaetsverlag (2007)
19. Mur-Artal, R., Tardos, J.: ORB-SLAM: Tracking and Mapping Recognizable Features. Robotics: Science and Systems (RSS) Workshop on Multi View Geometry in Robotics (MVIGRO) (2014)
20. Mur-Artal, R., Tard´os, Juan: Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM. Robotics: Science and Systems. (2015)
21. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS) (2015)
22. Sergey Triputen, Kristiaan Schreve, V.T., Rätsch, M.: Closed-form solution for imu based lsd-slam point cloud conversion into the scaled 3d world environment. In: submitted IEEE AIM (2017)
23. Shiry, S., Menhaj, M.B., Daronkolaei, A.G.: Multiple target tracking for mobile robots using the jpdaf algorithm. 2007 19th IEEE International Conference on Tools with Artificial Intelligence 01, 137–145 (2007)
24. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014), http://arxiv.org/abs/1409.1556
25. Usenko, V., Engel, J., Stückler, J., Cremers, D.: Direct visual-inertial odometry with stereo cameras. Robotics and Automation (ICRA) (2016)
26. Vu, T., Osokin, A., Laptev, I.: Context-aware cnns for person head detection. CoRR abs/1511.07917 (2015), http://arxiv.org/abs/1511.07917
27. Welch, G., Bishop, G.: An introduction to the kalman filter. Tech. rep., Chapel Hill, NC, USA (1995)