Human Pose Estimation in UAV-Human Workspace

Ju Wang[1(✉)], Wookjin Choi[1], Igor Shtau[2], Tyler Ferro[2], Zhenhua Wu[1], and Curtrell Trott[1]

[1] Virginia State University, Petersburg, VA 23806, USA
{jwang, wchoi, zwu}@vsu.edu
[2] NSWC, Dahlgren, VA, USA
{igor, tyler.j.ferro}@navy.mil

Abstract. A 6D human pose estimation method is studied to assist autonomous UAV control in human environments. As autonomous robots/UAVs become increasingly prevalent in the future workspace, autonomous robots must detect/estimate human movement and predict their trajectory to plan a safe motion path. Our method utilize a deep Convolutional Neural Network to calculate a 3D torso bounding box to determine the location and orientation of human objects. The training uses a loss function that includes both 3D angle and translation errors. The trained model delivers <10-degree angular error and outperforms a reference method based on RSN.

Keywords: 6D pose estimation · Deep learning · UAV motion planning

1 Introduction

We investigate a deep learning method to estimate the 6D pose of humans for autonomous UAV motion control in both urban and indoor human environments. As autonomous robots/UAVs become integral parts of future workspace, they will interact and co-exist with humans in a close quarter. A fundamental requirement is that the autonomous robot must operate/navigate with safety assurance in unknown environments such as office buildings or factories. From the perspective of robot motion planning, autonomous UAVs must Simultaneously Localize and Map (SLAM), which in turn rely on the construction of a 3D occupancy map of the environment using onboard sensors [3, 4]. The 3D occupancy map allows the UAV to avoid collision into static obstacles. More importantly, the UAVs must detect dynamic human objects, estimate their 6D poses, and predict their trajectory to plan a safe path.

In both 2D/3D human pose estimation [1, 2, 9], keypoints such as the pelvis, arms, head, and calves are often used to construct a mapped skeleton data structure. With the emerging AI frameworks and GPU hardware, many Machine Learning (ML) approaches [2, 5, 9] have been studied with promising results. Compared to many existing methods, the main difference of our method is that we predict 6D human pose

The project is supported by NSF award 1818655, Raytheon Space and Intelligence, and NEEC award N001742110011.

C. Stephanidis et al. (Eds.): HCII 2021, LNCS 13095, pp. 157–167, 2021.
https://doi.org/10.1007/978-3-030-90963-5_13

instead of 2D skeleton keypoints. The skeleton keypoints in existing methods are all 2D, and they do not offer direct information of the body orientation and the distance information. Our network is trained to detect a 3D bounding box instead of the skeleton keypoints. The benefit of our method is that the 3D bounding box is relatively invariant even though the human body can be in vastly different poses. The network processing pipeline consists of (1) a deep CNN network that processes RGB video frames and generates pose proposals of human objects, (2) an extended Kalman filter to select pose candidates using the pose estimation from past frames. We mainly discuss human pose representation and the implication on the design of the training process. As the human body is not rigid, there are many alternative ways to define the bounding box and the human pose. Accordingly, the training process and the loss function will differ. Two pose representations are studied here: (1) our proposed Torso Box Pose (TBP) derived from a set of 3D keypoints, and (2) selected skeleton points used in the coco human dataset. We provide a close examination of the prediction accuracy for both.

The rest of the paper consists of the following sections: Sect. 2 describes UAV perception and motion planning Architecture. Section 3 details the network structure and the regression layer design. We will discuss the human pose representations and the loss function. Section 4 provides a performance evaluation of our system as well as demonstration cases in UAV motion planning.

1.1 Related Work

Robot motion planning requires some form of map and knowledge of its location. In the early systems, a prior knowledge (map) of the environment is often required for motion planning/execution. The navigation map can be a topological or grid-based occupancy map, with the latter containing more terrain detail for turn-by-turn motion commands. In contrast to fixed maps, a more flexible and robust approach is Simultaneous Localization and Mapping (SLAM), which requires the robot to map the environment with onboard sensors on the flight [3, 4]. A brief overview of recent advances in SLAM is presented in [11]. Recently graph-based approaches to solve SLAM problems are gaining popularity [10]. In graph SLAM, the robot's pose nodes are optimized by minimizing the distance to observation nodes and edges.

Human pose information is critical to many computer vision tasks, such as human behavior recognition and human-computer interaction. Deep learning-based human pose detection has attracted many researchers in the last decade [2, 5, 9, and 12]. To accurately recognize a human's posture, body keypoints in appropriate locations such as the pelvis, arms, head, and calves are typically detected. For location-sensitive vision problems such as human pose estimation, semantic segmentation, and object detection, high-resolution representations are required.

In High-Resolution Net (HRNet) [9], keypoints in an image and the location confidence of the keypoint are used to construct mapped skeletal data that represents the original human pose. Frameworks such as ResNet and VGGNet first encode input images with low-resolution representations using subnetworks created by sequentially connecting high-resolution convolutions, and then recover high-resolution representations from encoded low-resolution representations. The HRNet maintains a high-resolution representation through the whole process instead of recovering high

resolution from low resolution and exchanges information across various resolutions. The advantage is that the resulting representation is richer in both semantic and spatial features. A closely related work is Residual Steps Network (RSN) [12]. RSN aggregates feature with the same spatial size (Intra-level features) efficiently to obtain delicate local representations, which retain rich low-level spatial information and result in precise keypoint localization. Most of the existing works have focused on detecting skeleton points while overlooking the body orientation, which is the main focus of our approach.

2 UAV Perception and Motion Planning Architecture

We briefly describe the overall architecture of the UAV navigation stack. To safely operate in complex environments, the UAV's motion is controlled by a collection of subsystems performing sensing, perception and motion planning. The overall system architecture is shown in Fig. 1. The real or simulated UAV provides sensor data include the RGB/depth camera data, lidar scan, the IMU data, the barometer data, and optionally the GPS data. Both the camera depth image and the Lidar scan are used to create a 3D map of the environment and estimate the visual odometry. To navigate a target location, the UAV's high-level controller searches the 3D map and calculates a feasible path based on the static map. The motion planning subsystem corrects the local map using the current sensor data and optimizes the local path at the motion control level. The local correction is essential to avoid collision with dynamic objects such as humans. A significant amount of vision processing is required to obtain the updated information of the 3D occupancy map, and a GPU-based object detection module assists this task.

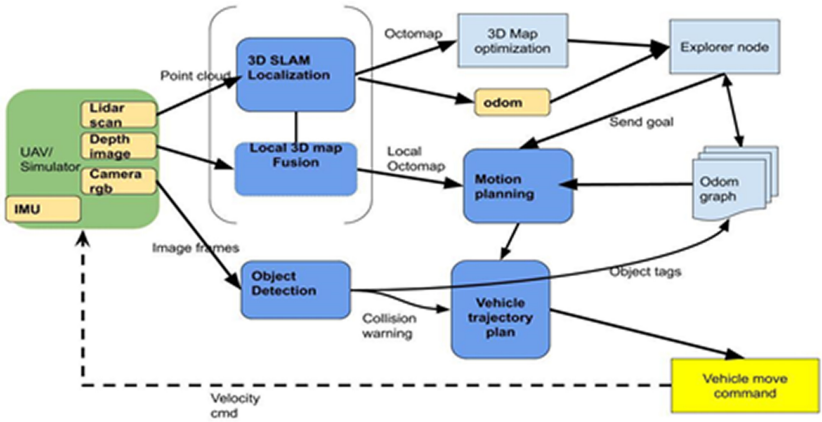


Fig. 1. UAV motion control software stack and information flow.

The perception and motion planning tasks are carried out by several computing nodes:

- 3D-SLAM-node: Construct a 3D occupancy map using the depth image input [7].
- Object-detection-node: This node is a deep Convolutional Neural Network (CNN) trained to recognize human objects and estimate human pose. The rest of the paper will focus on this node.
- Motion-planning-node: This node computes the feasible trajectory to reach a near-range waypoint based on a combined local 3D map.

3 Pose Detection Network Pipeline

We now describe how the 6D human pose can be represented and detected by a deep learning network. We will first discuss our proposed 3D Torso Box Pose (TBP). This is followed by an explanation of the deep CNN and training method. We then discuss the construction of the 6D pose from the coco human pose data serving as a comparison.

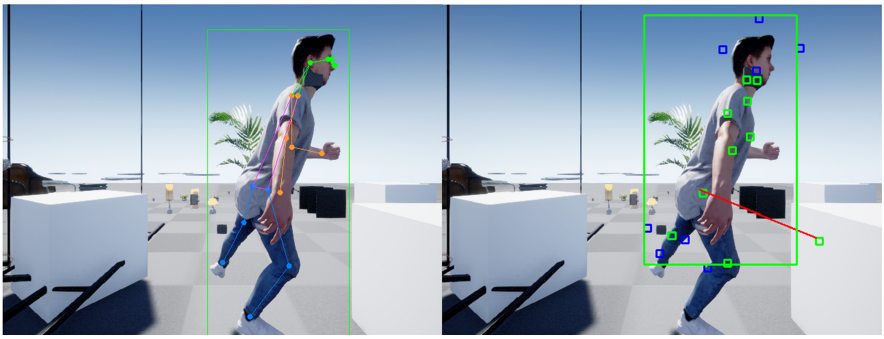


Fig. 2. (left)17 skeleton keypoints in coco pose, arm location and leg locations indicate different body orientation. (right) pose 3D bounding box (blue points) obtained in our model. (Color figure online)

3.1 6D Pose Representation and Reconstruction

In rigid body pose estimation, the body structure is well defined and remains unchanged. A successful prediction of the 6D pose of any body part is sufficient to describe the location and orientation of the entire body. However, the rigid body assumption is not valid in human pose estimation; hence the situation is more complicated. The human poses are typically described by skeleton keypoints such as the pelvis, head, mouth, upper arms, low arms, calves, and legs [8, 9, 12]. However, it is difficult to directly use the bone locations to form a consensus about the human body orientation. This is illustrated in Fig. 2(left), where the arm location and the leg locations indicate a completely different body orientation.

We define the human pose as the minimum 3D bounding box containing selected torso parts.

Assuming the camera coordinate system, we define $\overrightarrow{x_i} = (x_i, y_i, z_i)$, i = 1 … N as a set of N keypoints. The corresponding 3D bounding box is defined by the convex cube:

$$\overrightarrow{B} = \{p = (x, y, z) | min\overrightarrow{x_i} < x, y, z < \max(\overrightarrow{x_i})\}$$

An immediate question arises: *which keypoints should be included to construct the 3D bounding box?* Intuitive thought is to use all skeleton keypoints like the case of 2D bounding box for simple human detection. The problem, however, is that the resultant 3D bounding box might not align with the 3D torso bounding box. On the other hand, limiting to only a few torso keypoints will make the network training difficult since fewer human features are used. As a tradeoff, we include obvious joints such as the pelvis, upper arm, and calves. We further decide to have the head keypoint to allow enough features in the detection phase. Elbow and front arms are considered too 'free-moving and not used here.

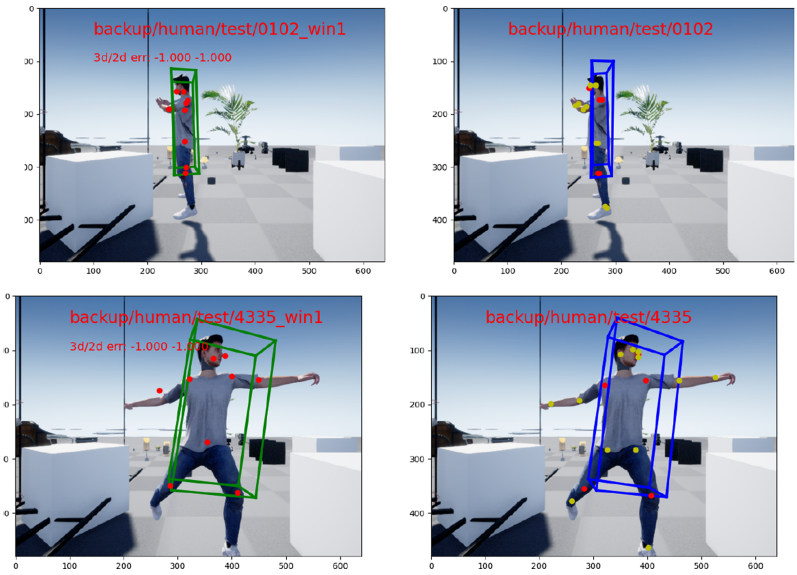


Fig. 3. 3D bounding boxes: ground truth vs. estimated from two different cases. (Color figure online)

With the 3D bounding box of the human body defined, the 6D pose can be estimated using the predicted 2D keypoints using a Perspective-n-Point (PnP) pose estimation method [17]. In our case, PnP uses only eight such control point correspondences and provides an estimate of the 3D rotation matrix R and 3D translation t of the object in the camera frame. Mathematically, the PnP find the optimum 3D transformation parameters $(R, t)^*$ that has the minimum 3D-2D projection errors over the 3D bounding box:

$$(R,t)^* = argmin_{R,t}(\sum\nolimits_{i:keypoints} ||p_{im,i} - A[R|t]P_{c,i}||)$$

Here $p_{im,i}$ represent the pixel location in the image plane, and $P_{c,i}$ is the keypoint in camera coordinate.

Figure 3 shows the estimated 6D pose (blue-box) and the ground truth (green-box) for two test cases. The distance of the human object to the camera is 8 m and 4 m, respectively. In both cases, the predicted 6D poses are very close to the ground truth.

3.2 Torso Box Pose Detection Network

Our approach is inspired by the success of the single-shot 6D pose estimator [5]. Our network architecture is shown in Fig. 4, which utilizes the convolutional layers of a YOLOv2 to predict the 2D projections of 9 keypoints for a human's 3D torso bounding box. We adopt the grid-cell concept of the YOLO2 network, where the original image is divided into 13 × 13 grids, and the network will be trained to predict one 3D bounding box per grid cell. In this study, we fixed to one anchor box dimension since the shape variation of the human body is relatively small compared to random objects.

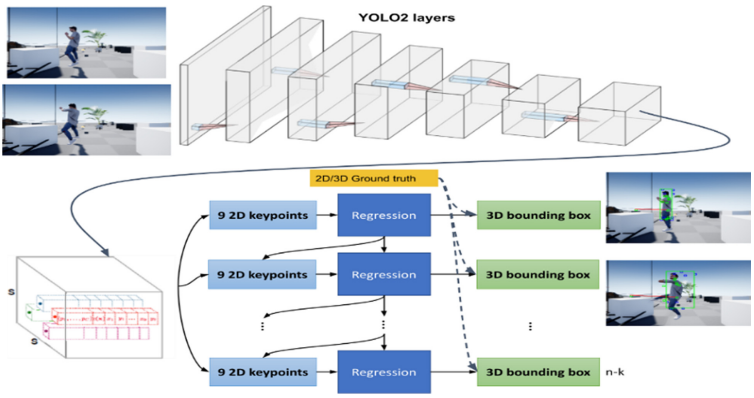


Fig. 4. Torso Box Pose detection network

For a 13 × 13 grid, the output of the last convolutional layer consists of about 169 3D bounding box proposals, each with the center inside the corresponding grid cell. Each box proposal predicted will contain nine pairs of normalized pixel coordinates plus the classification and confidence. After the convolutional layers, a final regression layer follows. We use a loss function that explicitly calculates the 3D bounding box errors to train the network. The loss function L consists of three parts:

1. l_{angle}: the error measured in quaternion distance between the projected rotation matrix R and the ground truth R*, We minimize the following loss function to train our complete network.

2. l_{trans}: the 3D distance error between the projected translation vector t and the ground truth t*.
3. $\sum_{i:keypoints} ||p_i - p_{i,gt}||$: the sum of the pixel distance between the projected and the actual keypoints.

$$L = \lambda_1 l_{angle} + \lambda_2 l_{trans} + \lambda_3 \sum_{i:keypoints} ||p_i - p_{i,gt}|$$

We also use the predicted results of the past k frames to exploit the temporal correlation in the video data. Since the predictions of the previous frames are strongly correlated to the current frame, they provide a tie-break for complex cases when human objects are partially observed or occluded. During network training, the old predictions contribute to the loss function and gradient calculation. At the testing/inferencing time, the old predictions refine the current prediction to improve the spatial accuracy.

3.3 RSN Comparison

To compare to some existing methods in human pose estimation, we also tested a state-of-the-art 2D pose estimation method using the MMPose toolbox, an open-source toolbox for pose estimation based on PyTorch and is a part of the OpenMMLab project [13]. The reference method we will compare to is the Residual Steps Network (RSN) [12], which won the 2019 COCO Keypoint Challenge.

The multi-stage network architecture of the RSN pose estimation is cascaded by multiple RSN. RSN uses effective intra-level feature fusion to learn delicate local representations. RSN differs from ResNet in the architecture of constituent units. RSN consists of Residual Steps Blocks (RSBs), while ResNet consists of "bottleneck" blocks. RSN is designed for learning delicate local representations through dense element-wise sum connections. Each human joint has a different scale. The scale of the eye, for example, is small, whereas the scale of the hip is large. As a result, architecture with a broader range of receptive fields is better suited to extracting characteristics related to various joints. Furthermore, a large receptive field aids in learning more discriminating semantic representations, which is beneficial to the keypoint classification task. The RSN creates extensive connections within RSB features with small-gap receptive fields. The deeply connected architecture aids in learning delicate local representations, which are critical for accurate human pose estimation. A Pose Refine Machine (PRM) is used in the last stage. Features are mixed after intra- and inter-level aggregation, containing low-level precise spatial information and high-level discriminant semantic information. Keypoint localization benefits from spatial information, whereas keypoint classification benefits from semantic information.

It is noteworthy that the keypoints location predicted by RSN, like many similar works, is in the 2D image domain. This makes it difficult to make a direct comparison between RSN and our model. To use the output of the RSN in 6D pose reconstruction, the 17-points coco keypoints are converted to the 8-points keypoints for the 3D bounding box by a simple process. We use the shoulder skeletons to derive the approximate location of the four (4) upper body torso keypoints. The offset of the nose

to the center of the shoulder points determines the squadron. The conversion algorithm to obtain four shoulder points works as following:

```
def SkeletonTo4ShoulderPoints((shoulder_l, shoulder_r, knee_l, knee_r,
nose):
    nose_off=nose-(shoulder_l+shoulder_r)/2
    if nose_off[0]>0:
        faceleft=1
    else:
        faceleft=0
    m,c = get_line_equs(shoulder_r, shoulder_l)
    m1,c1,m2,c2= get_lines_tang (shoulder_r, m,c) # approx with tangent line
    p1,p5 =get_line_y(m1,c1, shoulder_r)    ## front/back keypoints near left
shoulder
    p0,p4 =get_line_y(m2,c2, shoulder_l)    ##    keypoints near right shoulder
```

The keypoints near the knee are obtained similarly. After this, we apply the PNP method to estimate the coordinate transformation matrix representing the 6D pose. Figure 5 illustrates the keypoints outcome of the conversion process.

Fig. 5. Pseudo 3D box keypoints derived from RSN skeleton points

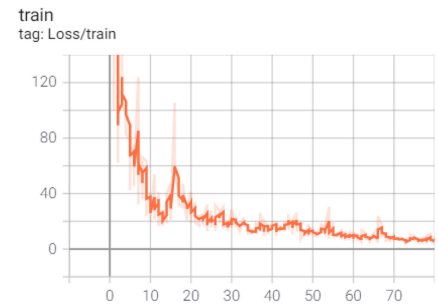
4 Experiment Results

4.1 Train Result

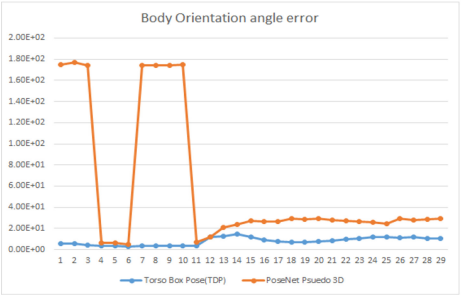
We use a game engine with 3D graphics and a physics engine to generate photo-realistic images and ground truth poses data to train and evaluate our network. Figure 6 shows the results of our model. Using a total 190 training images, the network

parameters converge within 40 epochs of training. The mean error of the keypoints is 8.6 pixels, while the average angular error is less than 8°.

The detection speed of the network benefits from the efficient implementation of the Yolo2. Our experiment shows that the system can process 20 frames/sec on NVIDIA Jetson TX2.



Meancornererror	8.608686
Mean3Derror	0.107596
Acc-15-px-2D-Projection	96.55%
Acc-using-0.18-3D-Transformation	89.66%
Translation-error:	0.084083
angle-error:	8.016824



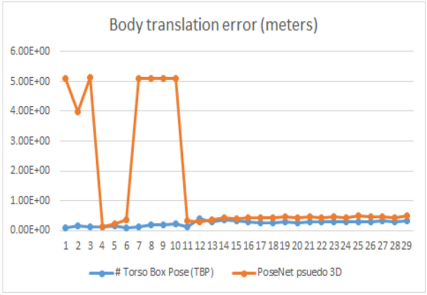


Fig. 6. Model train result and error: (a) loss function of a training session, (b) overall 2D and 3D errors, (c) 3D angle errors comparison, (d) 3D translation error comparison. (Color figure online)

Figure 6 C and D show the 6D pose error comparison between the proposed Torso Box Pose and the modified RSN method. The angle error is computed between the projected 3D pose vector and the ground truth pose vector. The angle error for TBP is uniformly under 20° and half of which are within a single-digit error.

We also noticed seven fail cases for the modified RSN method where the angle errors reach almost 170–200°. The failure is due to the fact that the simple 2D-3D mapping algorithm could not calculate a pose solution when the left and right shoulder points are too close. Such behavior is expected since the 2D skeleton points have limited 3D expressing power. Among the good testing cases, the angle error for the modified RSN method is about 30% higher than that of TBP. For translational error, the two methods are very comparable, with TBP has a slight edge.

4.2 Integrated Motion Planning Test

To test the pose detection algorithm, we created a testbed consisting of a 3D game environment (Unreal Engine 4) and an external software stack described in Sect. 2. The autonomous UAV in the game engine communicates with the external software components through the Airsim plugin framework [14]. The game engine and the UAV software stack are run in two separate computers to separate their GPU demand. Figure 7 (left) shows a snapshot of a testing instance with the camera feed in the right corner. Figure 7 (right) shows a local motion path generated by the motion planner. The motion planner will use the 6D pose information to estimate the potential trajectory of the human objects to amend the current map and generate a feasible path.

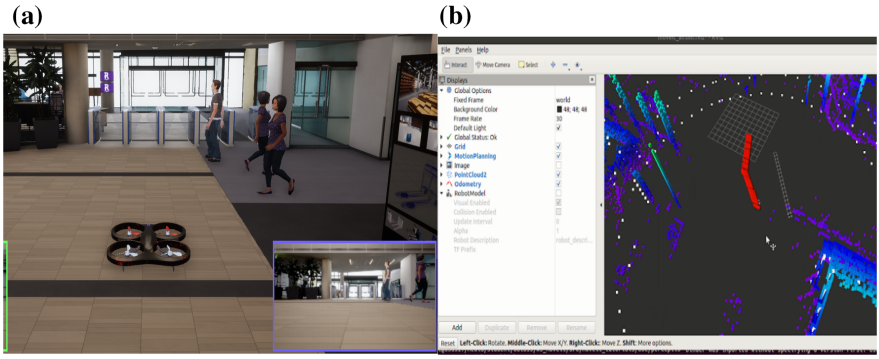


Fig. 7. Integrated testbed: (a) virtual office building with simulated UAV and human objects. (b) Visualized 3D motion path (red marks) in the 3D occupancy map.

5 Conclusion

We present a deep CNN-based pose estimation method to predict the true 6D pose information of human objects. This algorithm is an integral part of the motion planning stack for autonomous UAV control in human environments. We utilize a deep Convolutional Neural Network to estimate a 3D torso bounding box to determine the location and orientation of human objects. Our results show that the torso bounding box reflects the human pose well and is trainable through transfer learning.

References

1. Munea, T.L., Jembre, Y.Z., Weldegebriel, H.T., Chen, L., Huang, C., Yang, C.: The progress of human pose estimation: a survey and taxonomy of models applied in 2D human pose estimation. IEEE Access **8**, 133330–133348 (2020). https://doi.org/10.1109/ACCESS.2020.3010248
2. Pavllo, D., Feichtenhofer, C., Grangier, D., Auli, M.: 3D human pose estimation in video with temporal convolutions and semi-supervised training. In: CVPR (2019)

3. Laird, J.E., et al.: A standard model for the mind: toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics, AI Magazine **38**(4), 13–26 (2017)
4. 3D Mapping & Navigation. https://www.wilselby.com/research/ros-integration/3d-mapping-navigation/
5. Tekin, B., et al: Real-time seamless single shot 6D object pose prediction. In: CVPR (2018)
6. Li, S., Chan, A.B.: 3D human pose estimation from monocular images with deep convolutional neural network. In: Cremers, D., Reid, I., Saito, H., Yang, M.H. (eds.) ACCV 2014. LNCS, vol. 9004, pp. 332–347. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16808-1_23
7. Sünderhauf, N., Protzel, P.: Towards a robust back-end for pose graph SLAM. In: Proceedings of International Conference on Robotics and Automation, pp. 1254–1261 (2012)
8. Lin, T.Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
9. Wang, J., et al.: Deep high-resolution representation learning for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. (2020). https://doi.org/10.1109/TPAMI.2020.2983686
10. Thrun, S., Montemerlo, M.: The graph SLAM algorithm with applications to large-scale mapping of urban structures. Int. J. Robot. Res. **25**, 403–429 (2006)
11. Cadena, C., et al.: Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. IEEE Trans. Robot. **32**, 1309–1332 (2016)
12. Cai, Y., et al.: Learning delicate local representations for multi-person pose estimation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) ECCV 2020. LNCS, vol. 12348, pp. 455–472. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58580-8_27
13. MMPose Contributors: OpenMMLab Pose Estimation Toolbox and Benchmark (2020). https://github.com/open-mmlab/mmpose
14. Shah, S., Dey, D., Lovett, C., Kapoor, A.: AirSim: high-fidelity visual and physical simulation for autonomous vehicles. In: Hutter, M., Siegwart, R. (eds.) Field and Service Robotics. SPAR, vol. 5, pp. 621–635. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67361-5_40