IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

AUTONOMOUS DRONE FOR SPECIES IDENTIFICATION

Dr.P.S.IMMACULATE
Associate Professor, MBA dpt
SRI SAIRAM INSTITUTE OF TECHNOLOGY
CHENNAI, INDIA

Dr.SUGANTHI SU
Associate Professsor&HOD
ARTIFICIAL INTELLIGENCE&DATA SCIENCE
SRI SAIRAM INSTITUTE OF TECHNOLOGY
CHENNAI, INDIA
hodai@sairamit.edu.in

Mr. HEMANTH KUMAR C S
ARTIFICIAL INTELLIGENCE
SRI SAIRAM INSTITUTE OF TECHNOLOGY
CHENNAI, INDIA
sit20ad033@sairamtap.edu.in

Mr. ADITYA GURJALE S
ARTIFICIAL INTELLIGENCE
SRI SAIRAM INSTITUTE OF TECHNOLOGY
CHENNAI, INDIA
sit20ad005@sairamtap.edu.in

Abstract—People around the globe have a deep desire to discover lands that were never touched by any human before. People would like to come across such vast untouched lands and would like to explore their flora and fauna and would always make a comparison with the existing species and make the environmental mapping. **'Autonomous Drones'** resolve this problem by telecasting the untouched worlds and exploring their species. Some dense neural networks are being used for localization, mapping the environment, and detecting and recognizing the objects around it. The navigation process is achieved using the tangent bug algorithm. The whole process is created in a network, where each one communicates with the other and shares information. The information is then sent back to a base station where it is analyzed by botanist, zoologist, etc.

Keywords—Autonomous Drone, Localization, Mapping, Tangent Bug Algorithm

I. INTRODUCTION

Discovering flora and fauna involves exploring the natural environment to observe and identify different plant and animal species. This can be done in a variety of ways, including taking a hike through a wilderness area or consulting field guides and other resources. Through this exploration, one can learn more about the local ecology and appreciate the beauty and diversity of the natural world. The primary problem is **discovering new species in environments that are difficult for humans** to discover in most parts of the world.The current drone in use is a manually controlled drone that is used to explore the environment. It is controlled by people and it only captures images or videos of the terrain it visits. This project proposes to implement an autonomous drone whose mission is to accurately classify images using deep learning algorithms and computer vision techniques to identify objects, features and identify a new species in the environment. The main idea is to use**tangent bug algorithm** for path decisions. The primary goal of this project is to discover new species in the environment and help Botanists and Zoologists to make research easier.

II. EASE OF USE

The project involves the use of autonomous drone for unearthing new species present on the orb that are yet to be discovered. Some plants and animals would havealready been discovered, but due to lack of proper imagery, it became difficult for zoologists and botanists to classify the new species. They even found it difficult to know what the new species looked like due to the lack of data. These issues are resolved by this autonomous drone. The main purpose of this drone is to maneuver those places where humans can't reach easily and capture the flora and fauna present in that region for analysis. If in case it captures a new animal, it'll search it's database to check whether this animal exists or not. If not, it'll start capturing images of the new species and immediately sends it to a nearby base station for the corresponding people to look at.

III. FIELD WORK

Today's commercial MAVs have limited ability to autonomously avoid obstacles, are mostly teleoperated, and use GPS for navigation. Despite this, the study community has made great progress towards incorporating more visual-based processing to aid in state estimation and obstacle avoidance. From a single forward-facing camera, Alvarez et al. use a structure-from-motion (SfM) algorithm to determine depth and subsequently avoid obstacles. Using an RGBD camera, Bachrach et al. create an image that is then used for planning and localization. In order to localise a fixed-wing MAV and enable reliable flying in enclosed

spaces, Bry et al. combine an inertial measurement unit (IMU) with a laser range finder. Faessler et al. create a map of the surroundings using an IMU and a downward-facing camera and the SVO algorithm.For the purpose of creating maps and estimating states, Fraundorfer et al. combine a downward-facing camera with a binaural pair that faces forward. For the purpose of avoiding obstacles and creating maps, Scaramuzza et al. also incorporate an IMU and three cameras. According to Scherer et al., flying low over obstructions presents some difficulties. Recently, the application of deep reinforcement learning (DRL) to MAVs has also received notice. Bhatti et al.'s seminal work describes a method for learning to play first-person video games using DRL that combines visual mapping and visual recognition. Like this, Lillicrap et al. use DRL to discover end-to-end strategies for a variety of traditional continuous physics issues. Ross et al. use DRL to train an MAV to avoid obstacles while flying through a woodland. After only receiving instruction on artificial images, Sadeghi and Levine use DRL to teach an MAV how to fly inside buildings. Zhang et al. employ DRL to teach a simulated quadrotor object avoidance strategy.

Our research draws a lot of its inspiration from and is most closely related to the trail DNN work of Giusti et al., who developed a convolutional DNN that forecasts view orientation (right/left/straight) by amassing a large dataset using a head-mounted rig made up of three cameras. The steering instructions for an MAV to follow a trail were then determined using the trained DNN model. Rasmussen et al.'s earlier ground-based robotic research serves as a foundation for this study.

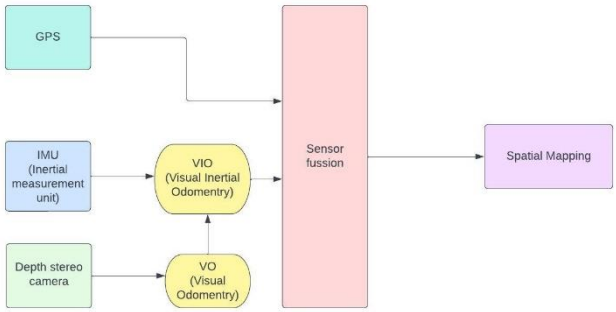
We also took inspiration from NVIDIA's DNN-controlled self-driving car, which gathers training data for three distinct directions using three dashboard-mounted cameras. To keep the vehicle straight and in the lane while training, the system calculates the proper steering effort for each virtual view (produced by interpolating video between the actual cameras). The DNN gains the ability to link visual data to the appropriate steering command in this way. The car is then driven using a single camera and the trained DNN in a range of environments and terrains. We modified this method by adding our own video, which was taken using a three-camera wide-baseline rig to capture side views, to the IDSIA woodland trail dataset. Thus, both lateral offset and view orientation inside the trail may be estimated by our system.

IV. PROPOSED SYSTEM

The concept entails the use of an autonomous drone to discover previously unknown species on the orb. Several plants and animals might have already been discovered, but due to a lack of good imaging, zoologists and botanists struggled to classify the new species. Due to a lack of evidence, they were unable to determine the appearance of the new species. This autonomous drone resolves these concerns. The major objective of this drone is to move into areas where humans cannot easily reach and capture the flora and wildlife there for analysis. If it captures a new animal, it will examine its database to see if the animal already exists. If not, it will begin capturing photographs of the new species and sending them to a nearby base station for the corresponding people to view.

This paper presents an autonomous navigation system using only Zed 2i stereo depth camera and a GPS receiver. The proposed method consists of visual odometry, post estimation, obstacle detection, local path planning and a way point follower.

The VO (Visual Odometry) computes relative pose between two pairs of stereo images. However, the VO suffers from noise accumulation over time. The GPS provides absolute locations that can be used to correct VO noise. The Zed2i stereo depth camera consists of IMU's (Inertial Measurement Unit) such as accelerometer, gyroscope, magnetometer. Initially, the VO data and IMU data are fused to get Visual Inertial Odometry. The VIO and GPS are fused to achieve more accurate localization both locally and globally using an Extended Kalman Filter (EKF). To detect obstacles, a depth sense map is constructed by stereo disparity estimation and transformed into a 2D occupancy grid map.



Local path planning computes temporary way points to avoid obstacles and a way point follower navigates the drone towards the goal point. This proposed method is evaluated in a simulation done in Gazebo using ROS (Robotic Operating System).

Drone Simulation

Environment

A. *Visual Odometry*

The first step is to obtain the cameras' intrinsic and extrinsic parameters. To accurately estimate the intrinsic camera characteristics, many photos of the calibration pattern at different poses must be captured. Second, when photos from both the left and right cameras are received, we must remove the lens-induced image distortion. Wide angle lenses typically have significant distortion at the image edges. The lens distortion parameters are included in the camera calibration output. To remove image distortion, we simply reverse the distortion process using these distortion settings.Third, since stereo VO involves feature matching between left and right pictures, it is important to rectify left and right image after un-distortion to make the same feature point lie on the same horizontal line. The feature searching space is decreased from 2D to 1D as a result, greatly enhancing algorithm performance.

Image rectification is the process of projecting both the left and right images onto the same image plane. This new picture plane must be parallel to the line connecting the left and right camera centers, ensuring that scene point projections are on the same horizontal line on both left and right rectified images. It also ensures that the inherent camera settings of the rectified left and right images are the same. Once the new picture plane is specified, we can simply use homography warping to rectify both images.

We estimate the drone movement between times *s-1* and *s* using two stereo pairs. We begin by establishing feature correspondences between these four photos. Feature matches between the left and right photos are utilized to reconstruct the 3D positions of the scene's features. We can find the 3D positions of the same set of feature points at both time *s-1* and time s. The transformation of those 3D locations is essentially the inverse of the robottransformation under the premise of rigid bodytransformation.

So, in the least squares sense, we estimate the rigid transformation (rotation R and translation t) of those 3D points by minimizing the following term:

$$\sum_{i=1}^{n} \left\| \boldsymbol{R}\boldsymbol{p}_{i,s-1} + \boldsymbol{t} - \boldsymbol{p}_{i,s} \right\|$$

Once the left and right images are corrected, we can simply retrieve the 3D position of feature points by defining the coordinate system in the center of the left camera:

$$z = \frac{fB}{d}, x = \frac{uz}{f}, y = \frac{vz}{f}$$

To estimate the drone's present attitude, we must aggregate frame-to-frame alterations from a known start point. If we suppose that the pose of the first frame is [0, 0, 0], then the stance of the robot at time s can be calculated using the following equation:

$$\begin{bmatrix} \boldsymbol{O}_s & \boldsymbol{P}_s \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_s & \boldsymbol{t}_s \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} \boldsymbol{O}_{s-1} & \boldsymbol{P}_{s-1} \\ 0 & 1 \end{bmatrix}$$

B. EKF Pose Estimation

Although VO computes a relative position between two pairs of stereo pictures accurately, noise accumulation occurs over time. A GPS device gives precise positions that can be used to correct VO drift. Using an EKF, we fuse data from VIO and GPS to provide more precise localization both locally and internationally.

Prediction and correction are the two processes in EKF pose estimation. The robot posture and covariance at time *s* can be predicted in the prediction step by:

$$\hat{\boldsymbol{x}}_s = f(\boldsymbol{x}_{s-1}, \boldsymbol{u}_s)$$
$$\hat{\boldsymbol{P}}_s = \boldsymbol{F}_x \hat{\boldsymbol{P}}_{s-1} \boldsymbol{F}_x^T + \boldsymbol{F}_u \boldsymbol{Q}_s \boldsymbol{F}_u^T$$

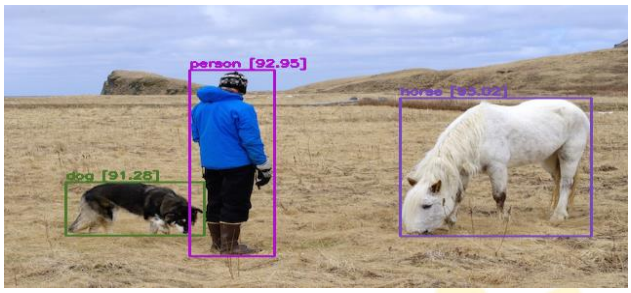
As new data from VIO or GPS becomes available, the prediction and correction stages are repeated. If new data from both VIO and GPS is available at time s, EKF estimates the robot pose by executing the prediction and correction steps consecutively. If new data from only VIO is available at time *s*, EKF updates the robot position by merely conducting the prediction step.

C. YOLO v4

YOLOv4 is a real-time Object Detection model that is SOTA (state-of-the-art). The YOLO detector is a one-stage detector. The One-stage approach is one of the two main cutting-edge methods used for Object Detection, and it prioritizes inference speeds. The classes and bounding boxes for the entire image are predicted in one-stage detector models since the ROI (Region of Interest) is not selected. As a result, they are faster than two-stage detectors.

YOLO's first version was written in the Dark Net Framework (which is a high performance open source framework for implementing neural networks written in C and CUDA). DarkNet is commonly used as a backbone network.
It divides the object-detection task into two parts: regression and classification. Regression predicts classes and bounding boxes for the entire image in a single run and aids in object positioning. The class of an object is determined via classification.

Species being recognized by YOLOv4



D. Obstacle Detection

This section describes the obstacle detection method based on dense stereo matching. Then, the cameras are pre-calibrated and installed on the platform with predefined pitch angle and ground plane height. Second, the cameras have a smooth trajectory, which allows them to avoid expensive computation for rotation and scale invariant feature descriptors. We can project between the world coordinate system, the camera coordinate system, and the image plane.

The algorithm's first step is dense stereo matching. For stereo matching in this system, we used the Library for Efficient Large-scale Stereo Matching. The library is fast enough to produce real-time matching results. The disparity map can be used to create 3D coordinates with the camera matrix from calibration.

The 3D coordinates of a matched point in camera coordinate system can be determined using the following equations given focal length f, principal point [Xc, Yc], and baseline B:

$$X_c = \frac{f(x - x_c)}{d}$$
$$Y_c = \frac{f(y - y_c)}{d}$$
$$Z_c = \frac{fB}{d}$$

With the extrinsic camera matrix and [Xc, Yc, Zc] coordinates in camera coordinate system, the point can be projected to world coordinate system as follows:

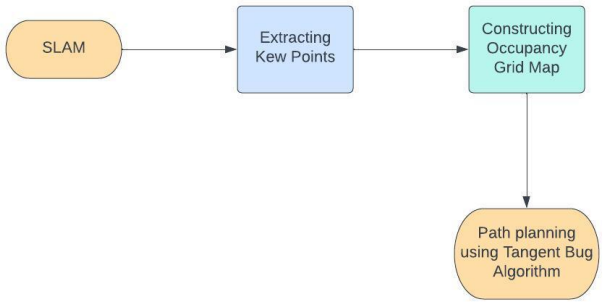
$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -sin\theta & -h \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

RANSAC is used to fit only one plane under this assumption. The RANSAC algorithm from the Point Cloud Library (PCL) was used to estimate planes in this algorithm. While it is powerful, the large number of points in a point cloud necessitates a lengthy computation time. So, in order to run online, the plane estimation process was designed to run periodically rather than every frame.To make plane estimation more computation efficient, some optimizations are made to the algorithm. The primary reason of the costly computation time is the large number of points as input to the RANSAC algorithm. Previously, we used all the points generated by the disparity map. Because the random sampling procedure, it requires all the points be included in error measurements. The big number of outliers also significantly increases the number of rounds. The aim here is to keep the input size as little as possible.

Because the algorithm's purpose is to estimate the ground plane, we can restrict the input points to a certain location in 3D space. Because the ground is still the dominant plane in the subsample of the point clouds, using the points in this region can still estimate the ground plane accurately. The sub sample point cloud also has less outliers, such as sky, trees, therefore the algorithm may converge in fewer iterations. It can shorten the computation time of RANSAC algorithm and attain real-time speed.

After estimating the ground plane, the inliers of this plane are removed. The leftover points are then project the points into an occupancy grid. Occupancy grid is a 2D grid map with each grid cell representing a block in the 3D space. Based on their location, points are projected to the ground plane and into different cells on the grid map. A cell is considered as an obstacle once the number of points in the cell hits a pre-defined threshold.

E. Simultaneous Localization and Mapping (SLAM)



The Simultaneous Localization and Mapping is performed using the Zed 2i stereo depth camera where it constructs 3D R-TAB Map which is constructed using point cloud mapping method. Using computer vision techniques, the key points are extracted from the map. These key points represent the obstacles in the real world. Then the extracted key points are used to construct occupancy grid map. The tangent bug algorithm is used to navigate the drone autonomously and to avoid the obstacles.

F. Local Path Planning

We use local path planning to steer clear of impediments and rejuvenate the way. Local path planning employs EKF posture estimate and Obstacle Detection (OD) discussed in the preceding sections. EKF pose estimation provides the drone pose for local path planning to localize the drone in the world coordinate. OD provides an occupancy grid map which is utilized for local path planning to avoid obstacles. We implement a Tangent Bug (TB) algorithm for local path planning.

1) Algorithm 1

We developed a DroneKitPython function to slowly increase the output of RC channel 3, the throttle channel, while continuously monitoring the quadcopter's altitude, until the target altitude is reached, at which point the function terminates. The function accepts the target altitude in centimeters as an argument.
This function is provided in Algorithm 1.

Algorithm 1 Take Off to Target Altitude

```
Input: A Target Altitude target
Output: NA
while 1 do
    channel[3].override ← channel[3] + 3
    if Altitude ≥ target * 0.95 then return
    end if
    time.sleep(1)
end while
```

This function may seem quite simple, which it is, and using RC overrides can produce some weird results from time to time. This is since you cannot predict what the RC channel will be when the method is called and that, there is no way to preset an RC channel to a specific value; instead, the override depends on the value of the current channel. Although this method lifts the quadcopter, it does so slowly as the propellers gradually rotate.

2) Algorithm 2

After the quadcopter has achieved a specified height, we want it to retain this altitude while all other activity is being done, until we trigger a landing mechanism. The easiest way to do this would be to put the APM firmware on ALT HOLD mode and allow the flight controller handle the labor-intensive tasks. However, because of RCoverrides this is not possible because it is not sure as to what the throttle channel value is, also even if the value is within a desired range, as stated earlier ALT HOLD mode will still allow the quadcopter to drift in altitude. This drift is normally minor but indoors it might pose complications. As a result, we added another function to DroneKit-Python that continuously monitors the quadcopter's altitude and makes minor adjustments to the throttle as needed. This function is called after reaching a target altitude and is executed in a separate thread to ensure that the target altitude is always maintained. This function is provided in Algorithm 2.

3) Algorithm 3

This function accepts 3 arguments, the first parameter is a 1 or -1 corresponding to direction. The second parameter is the period the movement should occur. The third parameter is the channel to override. The function is provided in Algorithm 3.

Algorithm 2 Holding Target Altitude

Algorithm 3 Moving Pitch, Roll, and Yaw

Input: Direction(1 or -1) **Direction**, Duration in seconds **Duration**, Channel to Override **Channel**
Output: NA

$channel[Channel].override \leftarrow channel[Channel] + (Direction * 5)$
for i **to** $Duration$ **do**
    time.sleep(1)
end for
$channel[Channel].override \leftarrow channel[Channel] - (Direction * 5)$

end if

The function, which moves the quadcopter slowly and steadily, overrides the corresponding channel by 5 units in the direction that was passed as an argument, as can be seen from Algorithm 3. It then waits for the duration of time that was passed as an argument before resetting the corresponding channel to its initial value.In order to be clear, direction 1 on channel 2 stands for forward pitch direction, direction -1 on channel 2 for backward pitch direction, direction 1 on channel 1 for right roll direction, direction -1 on channel 1 for left roll direction, direction 1 on channel 4 for clockwise rotation of yaw, and direction -1 on channel 4 for anticlockwise rotation of yaw.

4) Algotirhm 4

To land the quadcopter, we would simply switch the flight controller into LAND mode using the corresponding DroneKit-Python command and let the flight controller do the rest.

Algorithm 4 Landing the Quadcopter

Input: NA
Output: NA
while $Altitude > 13cm$ **do**
    $channel[3].override \leftarrow channel[3] - 5$
    time.sleep(1)
end while
$VehicleMode \leftarrow "LAND"$

V. References

[1] H. Alvarez, L. Paz, J. Sturm, and D. Cremers. Collision avoidance for quadrotors with a monocular camera. In International Symposium on Experimental Robotics (ISER), pp. 195–209, Springer, 2016.

[2] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy. Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments. Intl. J. Robotics Research, 31(11):1320–1343, 2012.

[3] S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N. Siddharth, P. H. S. Torr. Playing Doom with SLAM-augmented deep reinforcement learning. arXiv preprint, arXiv:1612.00380, 2016.

[4] M. Bojarski et al. End to End Learning for Self-Driving Cars. arXiv preprint, arXiv:1604.07316v1, 2016.

[5] A. Bry, A. Bachrach, and N. Roy. State estimation for aggressive flight in GPS-denied environments using onboard sensing. In International Conference on Robotics and Automation (ICRA), 2012.

[6] D.-A. Clevert, T. Unterthiner, S. Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). International Conference on Learning Representations (ICLR), 2016.

[7] D. Eigen, C. Puhrsch, R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In Advances in Neural Information Processing Systems (NIPS), pp. 2366–2374, 2014.

[8] J. Engel, V. Koltun, D. Cremers. Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[9] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza. Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle. Journal of Field Robotics, 33(4):431–450, June 2016.

[10] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semidirect monocular visual odometry. In International Conference on Robotics and Automation (ICRA), 2014.

[11] F. Fraundorfer, H. Lionel, D. Honegger, G. Lee, L. Meier, P. Tanskanen, and M. Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor MAV. In International Conference on Intelligent Robots and Systems (IROS), Nov. 2012.

[12] R. B. Girshick. Fast R-CNN. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2015.

[13] A. Giusti, J. Guzzi, D. C. Cires¸an, F.-L. He, J. P. Rodr´ıguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, D. Scaramuzza, L. M. Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. IEEE Robotics and Automation Letters, 1(2): 661–667, July 2016.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016.

[15] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys. An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications. In ICRA, 2013.

[16] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. 2016

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS), 2012.

[18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. arXiv preprint, arXiv:1509.02971, 2016.

[19] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.

[20] J. Michels, A. Saxena, and A. Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. Proceedings of Intl. Conference on Machine Learning (ICML), 2005.

[21] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. arXiv preprint, arXiv:1602.01783, 2016.

[22] G. Nutzi, S. Weiss, D. Scaramuzza, R. Siegwart. Fusion of IMU and ¨ vision for absolute scale estimation in monocular SLAM. Journal of Intelligent and Robotic Systems, 61(1):287–299, 2011.

[23] M. Pizzoli, C. Forster, and D. Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time, IEEE International Conference on Robotics and Automation (ICRA), 2014.

[24] C. Rasmussen, Y. Lu, and M. Kocamaz. A trail-following robot which uses appearance and structural cues. In Field and Service Robotics, pp. 265–279, 2014.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. arXiv preprint, arXiv:1506.02640, 2015.

[26] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. Learning monocular reactive UAV control in cluttered natural environments. IEEE International Conference on Robotics and Automation (ICRA), May 2013.

[27] F. Sadeghi and S. Levine. (CAD)2RL: Real single-image flight without a single real image. arXiv preprint, arXiv:1611.04201, 2016.

[28] A. Saxena, M. Sun, and A. Y. Ng. Make3D: Learning 3-D scene structure from a single still image. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 31(5):824–840, May 2008.

[29] D. Scaramuzza et al. Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in GPS-denied environments. IEEE Robot. Auton. Mag., 21(3):26–40, Sep. 2014.

[30] S. Scherer, S. Singh, L. Chamberlain, and M. Elgersma. Flying fast and low among obstacles: Methodology and experiments. International Journal of Robotics Research, 27(5):549–574, 2008.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[32] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, J. Dolan, D. Duggins, D. Ferguson, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, A. Kelly, D. Kohanbash, M. Likhachev, N. Miller, K. Peterson, R. Rajkumar, P. Rybski, B. Salesky, S. Scherer, Y. Woo-seo, R. Simmons, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, J. Ziglar, J. Struble, and M. Taylor, "Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge," Defense, vol. 94, no. 4, pp. 386–387, 2007.

[33] C. Stiller and J. Ziegler, "3D perception and planning for self-driving and cooperative automobiles," International Multi-Conference System Signals Devices, pp. 1–7, 2012.

[34] C. Siagian, C. K. Chang, and L. Itti, "Autonomous Mobile Robot Localization and Navigation Using a Hierarchical Map Representation Primarily Guided by Vision," Journal of Field Robotics, vol. 31, no. 3, pp. 408–440, 2014.

[35] M. Bibuli, M. Caccia, and L. Lapierre, "Autonomous Driving in Urban Environments: Boss and the Urban Challenge," IFAC Proc., vol. 7, pp. 81–86, 2007.

[36] F. Dayoub, T. Morris, B. Upcroft, and P. Corke, "Vision-Only Autonomous Navigation Using Topometric Maps," IEEE International Conference on Intelligent Robots and Systems, pp. 3-7, 2013.

[37] H. Morioka, S. Yi, and O. Hasegawa, "Vision-based mobile robots SLAM and navigation in crowded environments," IEEE Int. Conf. Intell. Robots and Systems, pp. 3998-4005, 2011.

[38] K. Konolige, E. Marder-Eppstein, B. Marthi, "Navigation in hybrid metric-topological maps," IEEE International Conference on Robotics and Automation (ICRA), pp. 3041-3047, 2011.