

Robust UAV Visual Teach and Repeat Using Only Sparse Semantic Object Features

Amirmasoud Ghasemi Toudeshki*, Faraz Shamshirdar* and Richard Vaughan

Autonomy Lab, School of Computing Science

Simon Fraser University

Burnaby, BC, Canada

ghasemit, fshamshi, vaughan@sfu.ca

Abstract—We demonstrate the use of semantic object detections as robust features for Visual Teach and Repeat (VTR). Recent CNN-based object detectors are able to reliably detect objects of tens or hundreds of categories in video at frame rates. We show that such detections are repeatable enough to use as landmarks for VTR, without any low-level image features. Since object detections are highly invariant to lighting and surface appearance changes, our VTR can cope with global lighting changes and local movements of the landmark objects. In the teaching phase we build extremely compact scene descriptors: a list of detected object labels and their image-plane locations. In the repeating phase, we use Seq-SLAM-like relocalization to identify the most similar learned scene, then use a motion control algorithm based on the funnel lane theory to navigate the robot along the previously piloted trajectory.

We evaluate the method on a commodity UAV, examining the robustness of the algorithm to new viewpoints, lighting conditions, and movements of landmark objects. The results suggest that semantic object features could be useful due to their invariance to superficial appearance changes compared to low-level image features.

I. INTRODUCTION

Visual Teach and Repeat (VTR) has become a canonical task in robotics, and has many practical applications including surveillance patrols, inspection of structures and transporting goods. The challenge is to conveniently teach and reliably repeat a path in a way that is robust to normal environmental changes such as lighting, weather, and appearance changes due to local activity. An interesting extreme version of the task is to use just a monocular camera: a cheap and ubiquitous sensor that does not rely on ambient infrastructure and could run in GPS-denied environments. VTR has been studied using land, air and water vehicles. Here we consider monocular VTR on Unmanned Air Vehicles (UAVs).

VTR has two phases. In the teaching phase, a map-like memory is recorded from observations made while the robot (or other training device) is piloted over the desired path. Then in the repeat phase the robot must (re)localize itself on the prior path, and repeat the remainder of it based on the prior observations in memory. The design of the memory

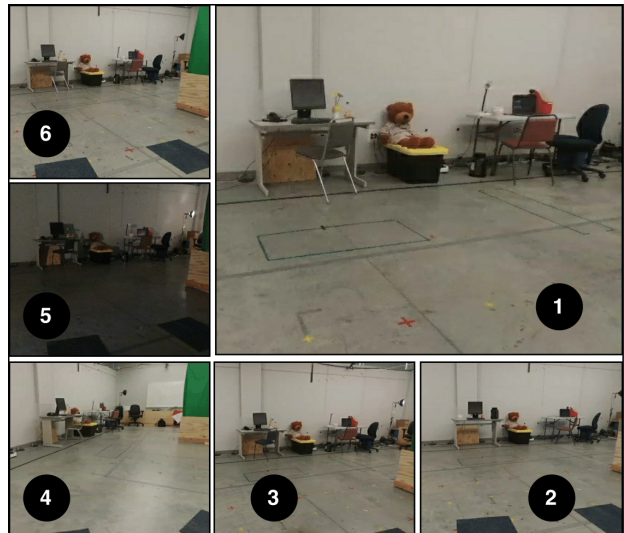


Figure 1. Relocalization robustness: the UAV is able to repeat a learned trajectory despite large changes in viewpoint and scene contents and appearance. (1) is the first image in the taught sequence; (2-6) are examples where the robot has successfully relocalized by matching a sequence of object locations (chair, screen, bear, etc.) from a sequence of descriptors stored during teaching and correctly located (1) as the closest matching image. Note the missing and moved objects in the repeats.

data structure is important. The two main approaches are (i) to make a 3D map using SLAM, with landmarks in 3D space; or (ii) to record a sequence of keyframes, with landmarks in image space. We will use object detections in image space to create compact keyframe descriptors.

Robust VTR requires invariance to changes in the appearance of the environment. Here we examine the use of semantic object detections as high-level image descriptors that are invariant to superficial appearance changes. Low-level features have some important drawbacks: direct descriptors which use image patches are sensitive to camera viewpoint and illumination changes. Point features like SIFT [1], SURF [2], ORB [3] and BRIEF [4] are less sensitive to illumination, but are still sensitive to the viewpoint of the camera, particularly in scale.

Recent deep convolutional neural networks (CNNs) trained on millions of examples provide excellent results

*These authors contributed equally to this work.

in detecting a wide variety of objects under enormous variations in illumination, occlusion and viewpoint. Here we exploit this to find objects in keyframes and use them as landmarks that are re-detected under very large scale, illumination and surface appearance changes, including translating or completely rotating the object, or replacing it with another object of the same category. Our intuition is that discrete, recognizable objects such as ‘lamppost’, ‘park bench’, and ‘traffic light’ should work as robustly salient features for remembering directions, as they do for humans.

Our approach to relocalization is similar to appearance-based algorithms but based on high-level features. We use sequential-temporal relocalization inspired by Sequential SLAM [5]. Using a temporal sequence of landmarks reduces the effect of perceptual aliasing caused by sparse landmarks and a limited landmark vocabulary. Our motion controller is based on the funnel control theory [6] with a novel extension that exploits the scene memory to anticipate the future.

II. RELATED WORK

VTR using UAVs is only recently feasible, with few early examples [7] [8]. Pfrunder et.al [7] demonstrate a quadrotor with downward-facing camera that repeats a simple path starting from the same position with no changes in the environment. The method is similar to [9] and based on SURF descriptors and pose-based localization. Nguyen et al [8] also choose SURF features for descriptors but their localization technique is appearance-based. Experiments are in simulation, using an AR-Drone model in Gazebo, and again the environment is unchanged between teach and repeat. Surber et al [10] build a map using structure from motion (SfM) from data captured in the teaching phase and execute visual-inertial odometry on a real UAV in the repeating phase for localization, using a previously constructed map as prior knowledge. They use BRIEF descriptors and report that relocalization is successful only where the appearance is similar enough to the reference map.

Barfoot has a significant body of work on Teach and Repeat navigation, including [11] which tested a ground robot in a path over 1km, demonstrating a color-constant method to handle changes in illumination. However, this work is sensitive to changes of viewpoint and hence requires the initial position to be similar. To handle gradually-accumulating environmental changes, [12] employs a ‘multi-experience’ method for VTR, which was introduced in [13]. Krajcik et al [14] examines combinations of feature detectors and descriptors for VTR, favouring the STAR feature detector and GRIEF feature descriptor for good accuracy with computational speed. GRIEF is a binary feature descriptor intended to be robust to seasonal changes in appearance. So-called ‘semantic’ segmentation of foreground objects from images is a traditional topic in computer vision and robotics [15], [16]. The recent development in deep neural networks, particularly CNNs has this area flourishing [17], [18], [19],

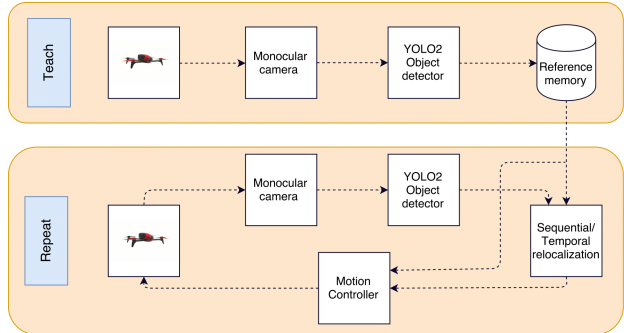


Figure 2. System Overview

[20]. New object detection algorithms provide much better accuracy and speed than was previously available [21], [22], [23] which inspired our attempt to use them as online real-time feature detectors.

III. METHOD

A. System Overview

Figure 2 outlines the structure of our system. During teaching, the UAV is flown or carried along the target trajectory. Video frames from the UAV are periodically input to the YOLO-2 CNN object detector. The CNN output is processed to obtain a scene descriptor of the locations of objects in the image. Each descriptor is stored in sequence. The finished sequence is the robot’s memory of the learned trajectory.

In the repeat phase, the UAV motion is actively controlled by the VTR system. Again, images from the robot are input to the object detector to obtain a scene descriptor. The relocalization module then finds the closest match between the current and memorized scenes. Once localized, the robot’s motion controller attempts to move the robot so that the next scene will look like the next scene in the stored sequence. These modules are described in more detail below.

B. Detection

We use the YOLO-2 [21] CNN object detector trained on the COCO dataset. It is notable for its speed in comparison with other deep networks like Faster RCNN [22]. We can detect objects at 40fps on a commodity PC with NVIDIA-GTX 1080Ti GPU.

The pre-trained YOLO model detects a variety of common objects and animals. It was not designed for UAV landmark detection. We use YOLO and an office/lab environment that contains objects detected by YOLO as a placeholder for a dedicated UAV landmark detector and outdoor flights in this initial study. The system should work as described without modification in real-world application environments given an appropriately trained feature detector. For example, VTR across a city park might use signposts, path junctions,

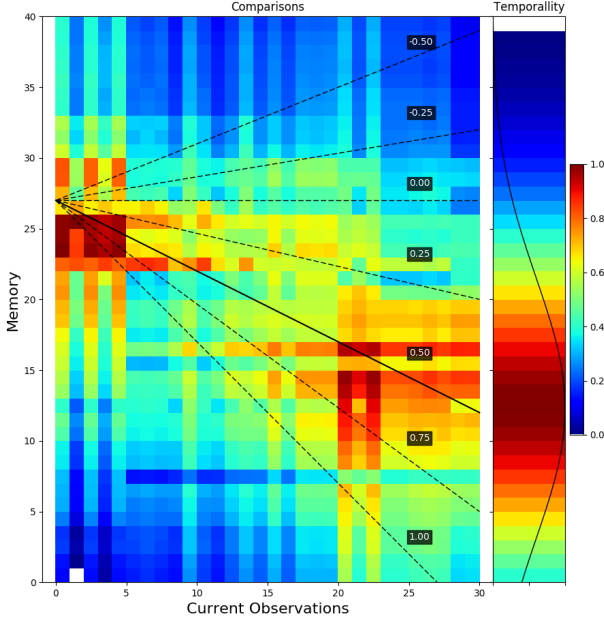


Figure 3. Extended SeqSLAM relocalization method: heatmap of the similarity of a sequence of 10 new scene descriptors compared to memorized descriptors. The line segment with the highest sum of values gives the best estimate of current UAV position and velocity. After initial relocalization we weight nearby scenes to impose a temporality constraint that avoids being confused by time-extended loop-closures.

benches, fence posts, flower beds, tennis courts, and swimming pools as semantic features.

C. Reference Memory

The CNN provides a bounding box, confidence and class label for each detected object. We discard detections below a confidence threshold (we used confidence ≥ 0.55). We also discard some object labels that are not reliable landmarks such as *person* and *cat* since their world positions are not fixed. Each detection is stored as a vector of five floating point values describing the object category and bounding box in the image. We found that each scene might contain two to eight objects, so the resulting whole-scene descriptor is small: of the order of $5 \times 8 \times 4 = 160$ bytes. For comparison an single original SIFT point descriptor is 512 bytes (both using 32-bit floats), and in typical use scenes contain 500 or more points, for 256,000 bytes per scene. Our scene memory is thus around 1600 times smaller than a SIFT-based method while achieving reliable VTR.

D. Relocalization

After the teaching phase, the reference memory contains a sequence of scene descriptors. Starting from an initial robot pose that can be arbitrarily far from the original trajectory, the robot needs to localize itself in the environment to be able to take a proper action. Given the current observation,

it should find the best-matching descriptor in the reference memory.

For matching, we measure the similarity of two individual object detections with matching labels as the ratio of the area of overlap of their bounding boxes to the sum of their bounding box area:

$$S_O = \frac{(\alpha s' \cap \alpha s'')}{\alpha(s' + s'')} \quad (1)$$

Where s' and s'' denotes the area of the 2D bounding boxes of objects recorded in the target and tested images respectively. α is a weighting factor to tune matching sensitivity. We chose this by experiment, to get a working value of 4. The similarity of two scenes is computed by averaging the similarity scores of all their objects (S_O):

$$S_I = \frac{\sum_{n=1}^N S_O^n}{N} \quad (2)$$

where N is the number of objects in the scene.

Compared to point-feature approaches, we have very few features in each scene, and a dictionary of only a 80 object categories. Thus scene descriptors are not highly discriminative: in our setting lots of scenes contain a chair below a keyboard below a screen. This problem is also present in methods where low-resolution whole-images are used as descriptors, so we borrow the SeqSLAM technique developed for that domain [5] where localization is done over a contiguous sequence of images. The last N observations are compared with the memorized observations and a score table is recorded. Since the robot may have different velocities in the teach and repeat phases, we must test different velocities to find a good match. Figure 3 shows an example score table and line segments corresponding to different UAV velocities. We find the line that passes through score cells with the highest sum. The gradient of the line gives the UAV velocity and the peak value along the line gives the current position along the trajectory.

Testing relocalization, we found that in trajectories with repeated sections (time-extended loop-closures), basic sequence matching could match the wrong location and the robot could become stuck in a loop. To address this, after initial relocalization, we positively weight descriptors nearby in the sequence to add a temporality constraint. With this mechanism, the robot can correctly repeat trajectories with repeated parts.

Adding the temporality constraint, the similarity measure becomes:

$$S_t = \frac{S_s(1 + \beta \text{norm}(x = i, \mu = i', \gamma))}{(1 + \beta)} \quad (3)$$

Where S_s is the score of matching that has been done by sequential comparison and β is the factor of temporality. i is the index of the memory which is being calculated and i' is the average of the last 5 matched indexes of the memory.

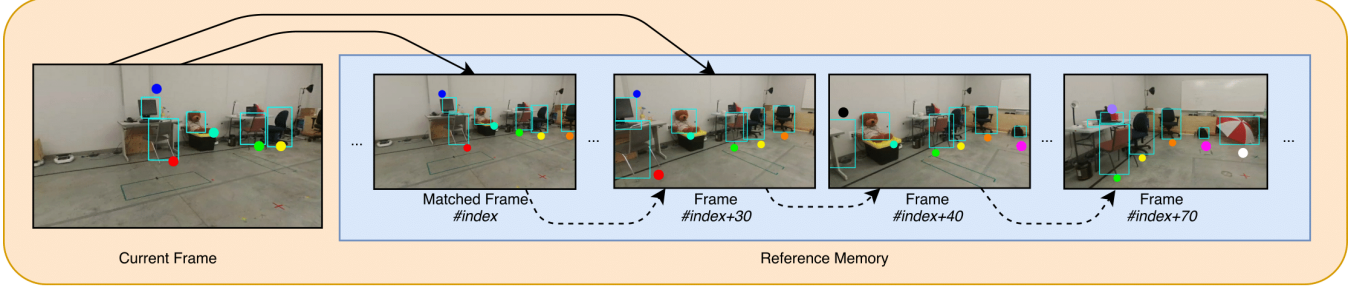


Figure 4. Funnel Lane with short look-ahead window plus Virtual Funnel Lane with long look-ahead window to react to unseen but predicted objects.

E. Motion Control

Our learned trajectory is in the form of object locations within keyframes, so we use a following controller afforded by this information. ‘funnel lane’ path following controller [6] method uses image-space point landmarks to achieve robust visual servoing. We extend it here for the case of object landmarks. We control robot pose on a 2D plane of constant altitude only, but the method is trivially extended to altitude control. Three points are considered for each object detection: one at the center and one at each extreme of its bounding box in the horizontal axis. Using the extent of the object allows us to reason about its scale change (but not its size in 3D). Figure 5 sketches the geometry of how two detections of the same object can be used to construct a ‘funnel lane’ that guides the motion of the robot to obtain the second camera position given the first.

One object detection is enough to repeat an (x, y) trajectory with respect to that object, but the approach direction is not constrained. A second object allows us to control yaw, to get replay of the full sequence of 2D poses (x, y, θ) .

Equation 4 shows the funnel lane controller response for the single landmark on the left x-extent of the recognized object in Figure 5. c_1 and d_1 are the horizontal distance of the land mark in image space of the current position and desired position respectively, and $\phi(c_1, d_1) = \frac{1}{\sqrt{2}}(c_1 - d_1)$.

Equation 5 averages the response for a whole object over its three landmarks. The final response is a command for heading adjustment. We scale the output by a constant gain for tuning (we used a factor of 12).

$$\theta_{c_1} = \begin{cases} \gamma \min\{c_1, \phi(c_1, d_1)\}, & \text{if } c_1 > 0 \text{ and } c_1 > d_1 \\ \gamma \max\{c_1, \phi(c_1, d_1)\}, & \text{if } c_1 < 0 \text{ and } c_1 < d_1 \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

$$\theta_{obj} = \frac{(\theta_{c_1} + \theta_{c_2} + \theta_{c_3})}{3} \quad (5)$$

Often, several objects are detected in a single scene. When this occurs, the controller calculates the response for each object individually using Equation 1. Then it performs the

funnel lane method for all objects as sketched in 5 (a full description is omitted for space).

$$\theta_{image} = \frac{\sum_{n=1}^N \theta_{obj}}{N} \quad (6)$$

The basic funnel controller has a failure mode that is difficult for our application: it steers the robot to align the currently-perceived objects as they should appear in later frames. But it does not react to upcoming objects that can not yet be seen. Thus with a narrow field of view camera it fails to turn the robot in sharp corners. To address this, we augment the funnel lane controller with what we will call a *virtual funnel lane*: we simulate the action of the normal funnel lane controller by running it on the sequence of stored images from the current best match up to some window lookahead size. The robot simulates what will happen in the future, assuming it is correctly localized. This was the robot can ‘see around’ upcoming corners. The robot records the heading changes prescribed by the controller in this virtual look-ahead flight, and averages them to obtain a predicted future yaw value. This is then blended with the original funnel lane control by Equation 9. The intuitive effect of these two control components is that the ‘real’ funnel lane aligns the robot well to the things it can see now, while the virtual funnel lane pre-aligns the robot to objects it can not yet see. Thus the robot will turn nicely around corners even when the set of objects in view changes completely. A detailed analysis of this method is beyond the scope of this paper.

$$\theta_{Virtual} = \frac{\sum_{i=I_m}^{W_v} \theta_{image}(i, i+1)}{W_v} \quad (7)$$

$$\theta_{Funnel} = \frac{\sum_{i=I_m}^{W_f} \theta_{image}(current, i)}{W_f} \quad (8)$$

where I_m is the index of the matched keyframe and W_v and W_f denote the size of the virtual funnel window and funnel control window respectively. We used $W_v = 70$ and $W_f = 30$.

$$\theta_{total} = \alpha \theta_{funnel} + (1 - \alpha) \theta_{virtual} \quad (9)$$

Where θ_{funnel} This enhanced controller is illustrated in Figure 4. The current image is only being compared with the next 30 key-frames in the memory to generate a funnel lane control command, but the virtual funnel lane looks ahead 70 key-frames to consider objects not yet in view such as the umbrella. The look-ahead anticipates the turn towards the umbrella and forces the controller to change the robot's heading towards it.

To complete control logic is: (i) rotate on the spot if not well localized; (ii) if yaw error is below a threshold, go forward with constant velocity, else yaw to correct the error. We decoupled yaw and forward motion as the low-level flight controller behaved badly when attempting to yaw and translate at the same time due to the complex vehicle and controller dynamics of a quadrotor.

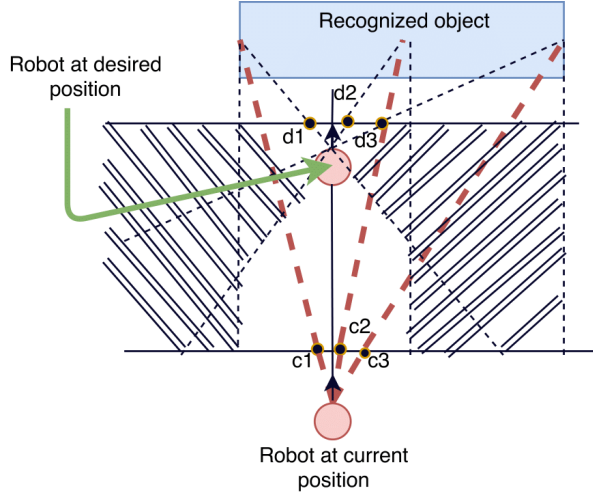


Figure 5. Schematic Funnel Controller

IV. EXPERIMENTS

We performed real-world experiments with a commodity Bebop 2 UAV. We are limited to a 10m x 6m experimental arena where we can independently measure UAV trajectories using a Vicon motion-capture system. The test environment is pictured in schematic in Figure 6. We limited the total number of recognizable objects in the environment to 18.

A. Training

In the training phase, the UAV is used as a passive camera. It is placed on a wheeled cart and pushed along the desired route, indicated by the trajectory marked with a camera Figure 6. The altitude is constant at 1.5m. The pose is recorded independently by motion capture. The video is fed to the CNN and descriptor generator and the sequence of memory of 290 key-frames is built.

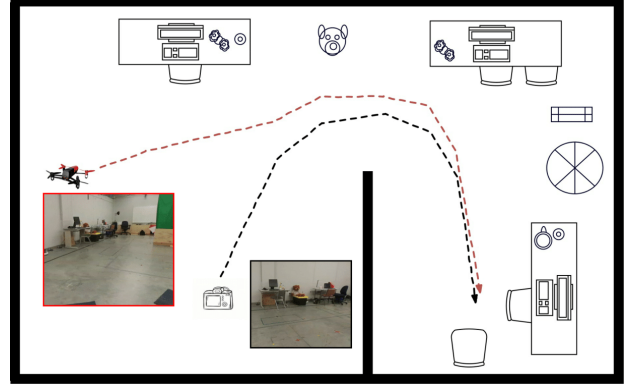


Figure 6. Test scenario: The trajectory of the camera is encoded as a series of object landmark locations in image space. Later, the UAV autonomously localizes itself in this space, then repeats the remaining part of the camera trajectory. A CNN detects objects such as desk, chair, mug, umbrella, backpack in keyframes. The UAV trajectory is generated directly from pairs of keyframes.

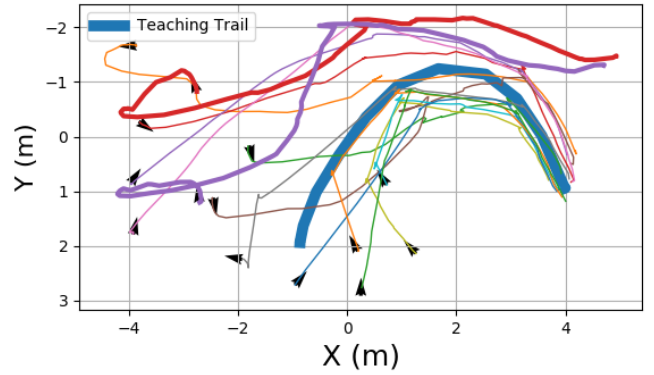


Figure 7. Trails in both teaching and repeating phases, arrows are indicating initial rotation of the robot.

B. Experiment I: Changing Viewpoint

A first experiment was performed to test the robustness of relocalization and trajectory repeating to changes in initial viewpoint. We run 12 trials with the the robot starting from (x, y, θ) poses chosen at random up to 5m from the original camera, and one of two fixed altitudes chosen at random. The robot flies autonomously, controlled by the VTR system. On startup, the robot rotates on spot until relocalization is achieved, then attempts to repeat the trajectory until matching the last keyframe.

1) *Results:* Out of 12 trials, 10 robots completed the learned trajectory, arriving less than 0.5m from the original end-point. Figure 7 reports the start positions and the paths of all robots. The blue bold line is the taught path. Note that the robot navigates around an opaque wall (Fig 6), so that the features detected in the second half of the trajectory are not visible in the first half: direct visual servo to the final target can not solve this task. In both the failed cases, the robots incorrectly detected the endpoint keyframe too

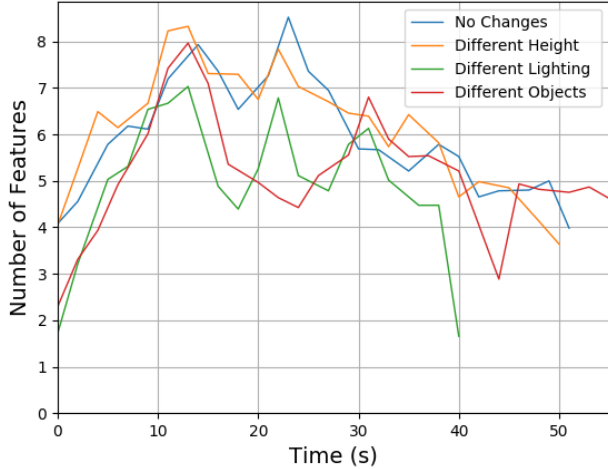


Figure 8. Number of features detected at nearest-corresponding locations during teaching and repeat phases.

early, stopping at the pair of chairs at the top right, and not the correct final pair of chairs at the bottom right. Both did correctly localize initially and completed 65% of the trial correctly.

C. Experiment II: Changing Environment

A second experiment investigates the ability to repeat a learned trajectory when the environment changes. The robot starts approximately at the same place as in teaching. But in one set of trials we change the lighting by turning off the ceiling lights and turn on two spot lamps. In a second set of trials we remove five objects that were noted by the object detector in training, and move eight others. The supplementary video shows us disturbing the objects to change their appearance. We repeat the trajectory and record the time taken to arrive at the final landmark.

1) *Results:* Examining robustness to appearance changes, we record the time taken to repeat the trajectory after (a) drastically changing the lighting and (b) removing objects and moving others to change their appearance. A histogram of flight times is shown in figure 9 with fitted Gaussians.

Removing and moving objects changes the execution time distribution, increasing it by an mean of 4sec (less than 10%).

But, perhaps surprisingly, changing the illumination reduces time taken for a repeat. The data show that the changed illumination slightly reduced the number of recognized objects, in particular the objects that were less repeatably detected in the teaching phase. Without objects disappearing and reappearing as frequently, the flight controller produced smoother behaviour and reached the goal faster. Perhaps the method could be improved by deliberately filtering such 'flickering' objects.

Figure 8 provides evidence of this effect, showing that the number of detected objects differs during the light-

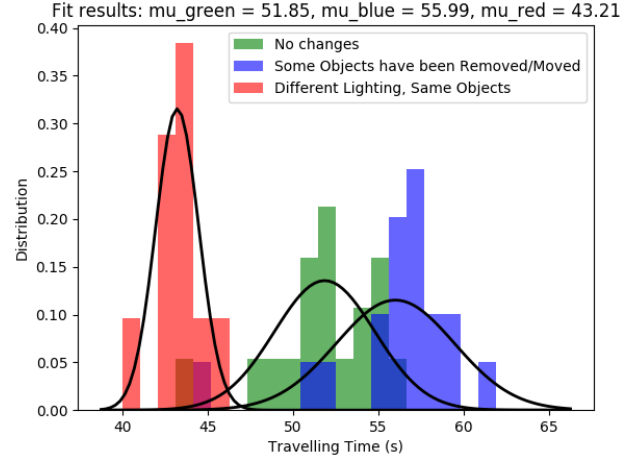


Figure 9. Repeat completion times for identical environment (color), light-changed environment (color) and object-moved environment (color).

changing and removing/moving objects experiments. When removing/moving objects, the number of detected features decreases, but the run time is longer because the moved objects degrade the controller behaviour, and the robot takes longer to line itself up along the trajectory. In the spot-lighting trials, the fewer object detections improve completion time as described above.

V. CONCLUSION, LIMITATIONS AND FUTURE WORK

In this paper we showed that a current deep-CNN object detector can be used to provide robust and repeatable features that are sufficient for monocular VTR. We demonstrated the method with real UAV experiments, but in an lab/office-like environment as a substitute for our eventual goal of large scale outdoor flights. The indoor environment allowed us to use a pre-trained object detector since a detector for UAV navigation tasks is not available, and allowed us to collect ground truth data from an installed motion capture system. Objects are detected with viewpoint and lighting invariance that compares favourably with other methods, and with interesting novel invariances to the object being completely turned around or replaced by another of the same category. We demonstrated that we can start the robot 5m away from the taught initial position, facing at a random heading, and the robot could relocalize itself and complete the trajectory using the object detections alone. Comparable experiments using SURF features have been shown to work at not more than 1m disturbance. We also successfully repeated a trajectory with the UAV's altitude changed by 40%, and also if several objects are removed and removed, or if the global lighting is changed dramatically.

We also contribute a novel two-window use of the funnel lane visual servo method that looks ahead in time to respond to upcoming objects.

Our method depends on a supply of distinct ambient

objects. It will not work in environments without discontinuous, recognizable object categories. Previous work on automatic generation of low-level feature dictionaries could suggest directions for future work. We aim to take this work outdoors over kilometer scales. For this, we need a CNN that reliably detects objects that exist in our target environment. We may have to train our own networks on the salient objects. In settings where distinct objects are rare, we could extend the method to use point features and/or inertial sensing to allow navigation in object-free areas.

One simple but interesting extension of the work would be for the robot to announce changes to the set of objects. If a previously-seen vase or painting is missing, perhaps it has been stolen. If a backpack has appeared, perhaps it was left accidentally. More substantially, perhaps a robot is performing an inspection trajectory over a new manufactured object such as an aeroplane wing. Is a rivet missing or has a bad weld appeared, compared to the teach phase recorded over a known-good example? If patrolling a construction site at night, a newly arrived truck or absent machine could be a security issue.

The recent performance improvement in some computer vision tasks due to CNNs and similar methods are giving us interesting new choices for robot vision. We aim to exploit the new robust vision methods towards more complete robot autonomy and new applications.

ACKNOWLEDGMENT

Supported by the NSERC Canadian Field Robotics Network.

REFERENCES

- [1] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The Proc. of the seventh IEEE Int. Conf.*, vol. 2. Ieee, 1999, pp. 1150–1157.
- [2] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer vision–ECCV 2006*, pp. 404–417, 2006.
- [3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE Int. Conf.* IEEE, 2011, pp. 2564–2571.
- [4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," *Computer Vision–ECCV 2010*, pp. 778–792, 2010.
- [5] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *Robotics and Automation (ICRA), 2012 IEEE Int. Conf.* IEEE, 2012, pp. 1643–1649.
- [6] Z. Chen and S. T. Birchfield, "Qualitative vision-based path following," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 749–754, 2009.
- [7] A. Pfrunder, A. P. Schoellig, and T. D. Barfoot, "A proof-of-concept demonstration of visual teach and repeat on a quadcopter using an altitude sensor and a monocular camera," in *Computer and Robot Vision (CRV), 2014 Canadian Conf.* IEEE, 2014, pp. 238–245.
- [8] T. Nguyen, G. K. Mann, R. G. Gosine, and A. Vardy, "Appearance-based visual-teach-and-repeat navigation technique for micro aerial vehicle," *Journal of Intelligent and Robotic Systems*, vol. 84, no. 1-4, pp. 217–240, 2016.
- [9] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.
- [10] J. Surber, L. Teixeira, and M. Chli, "Robust visual-inertial localization with weak gps priors for repetitive uav flights," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 6300–6306.
- [11] L. Clement, J. Kelly, and T. D. Barfoot, "Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery," *Journal of Field Robotics*, vol. 34, no. 1, pp. 74–97, 2017.
- [12] K. MacTavish, M. Paton, and T. D. Barfoot, "Visual triage: A bag-of-words experience selector for long-term visual route following," in *Robotics and Automation (ICRA), 2017 IEEE Int. Conf.* IEEE, 2017, pp. 2065–2072.
- [13] M. Paton, K. MacTavish, M. Warren, and T. D. Barfoot, "Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on.* IEEE, 2016, pp. 1918–1925.
- [14] T. Krajník, P. Cristóforis, K. Kusumam, P. Neubert, and T. Duckett, "Image features for visual teach-and-repeat navigation in changing environments," *Robotics and Autonomous Systems*, vol. 88, pp. 127–141, 2017.
- [15] J. G. Rogers, A. J. Trevor, C. Nieto-Granda, and H. I. Christensen, "Simultaneous localization and mapping with learned object recognition and semantic data association," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ Int. Conf.* IEEE, 2011, pp. 1264–1270.
- [16] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1352–1359.
- [17] N. Sünderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P. Corke, G. Wyeth, B. Upcroft, and M. Milford, "Place categorization and semantic mapping on a mobile robot," in *Robotics and Automation (ICRA), 2016 IEEE Int. Conf.* IEEE, 2016, pp. 5729–5736.
- [18] B. Mu, S.-Y. Liu, L. Paull, J. Leonard, and J. P. How, "Slam with objects using a nonparametric pose graph," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ Int. Conf.* IEEE, 2016, pp. 4602–4609.

- [19] T. Taisho and T. Kanji, "Mining dcnn landmarks for long-term visual slam," in *Robotics and Biomimetics (ROBIO), 2016 IEEE Int. Conf.* IEEE, 2016, pp. 570–576.
- [20] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, "Probabilistic data association for semantic slam," in *Robotics and Automation (ICRA), 2017 IEEE Int. Conf.* IEEE, 2017, pp. 1722–1729.
- [21] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.
- [22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.
- [23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*. Springer, 2016, pp. 21–37.