DROAN - Disparity-space Representation for Obstacle AvoidaNce: Enabling Wire Mapping & Avoidance

Geetesh Dubey[1], Ratnesh Madaan[1] and Sebastian Scherer[1]

***Abstract*— Wire detection, depth estimation and avoidance is one of the hardest challenges towards the ubiquitous presence of robust autonomous aerial vehicles. We present an approach and a system which tackles these three challenges along with generic obstacle avoidance as well. First, we perform monocular wire detection using a convolutional neural network under the semantic segmentation paradigm, and obtain a confidence map of wire pixels. Along with this, we also use a binocular stereo pair to detect other generic obstacles. We represent wires and generic obstacles using a disparity space representation and do a C-space expansion by using a non-linear sensor model we develop. Occupancy inference for collision checking is performed by maintaining a pose graph over multiple disparity images. For avoidance of wire and generic obstacles, we use a precomputed trajectory library, which is evaluated in an online fashion in accordance to a cost function over proximity to the goal. We follow this trajectory with a path tracking controller. Finally, we demonstrate the effectiveness of our proposed method in simulation for wire mapping, and on hardware by multiple runs for both wire and generic obstacle avoidance.**

I. Introduction

Thin obstacles such as wires, ropes, cables, and power lines are one of the toughest obstacles to detect for unmanned aerial vehicles. They can be especially hard to perceive in cases when the background is cluttered with similar looking edges, when the contrast is low, or when they are of barely visible thickness. Furthermore, using a horizontal binocular stereo pair does not help if they are right on the drone's path and parallel to the stereo baseline, thus evading detection in the disparity image. State of the art in drone obstacle avoidance, arguably Skydio [1], admit that their technology can not detect wires and power lines. Power line corridor inspection is another area of potentially widespread application of wire detection and avoidance capabilities, and leveraging UAVs for this task can save a lot of money, time, and help avoid dangerous manual labor done by linemen. For our purposes, we favour cameras because they are cheap, low weight, have long range sensing capabilities, and consume a fraction of power as compared to expensive and heavy sensors like lidar, radar, etc.

To the best of our knowledge, previous work does not demonstrate thin obstacle detection, depth estimation, and avoidance all together in a single package, nonetheless there are multiple works which attempt to solve parts thereof. [2] developed a method to avoid multiple strings in a cluttered, indoor environment by convexifying the free space and using mixed integer programming to generates trajectories consisting of polynomial segments. However, the assume that the location the of strings is provided to them as convex hulls, and hence, do not solve the perception challenge. [3] develop a generic, thin obstacle detection pipeline for both monocular and stereo cameras based on previous work edge-based visual odometry [4]. They also show results for thin obstacle detection on with a DJI Guidance stereo camera, but do not show avoidance. [5] does image based reconstruction of 3D wire art objects, but their method is not real time.

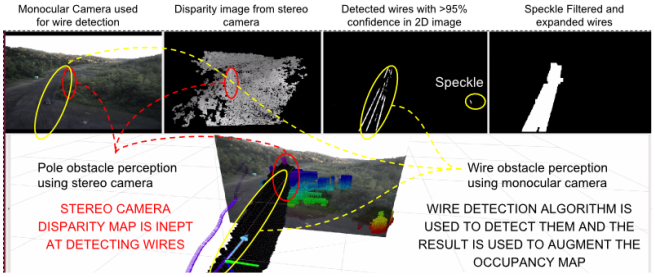


Fig. 1: Wire Mapping using Monocular Detection and Obstacle Perception using Disparity Expansion. Obstacles perceived using a stereo-camera are shown in red ellipses, and wire obstacles detected from monocular camera are shown with yellow ellipses. The bottom half shows the obstacle grid represented by our pose of graph of disparity images. Wires are shown as black boxes, and stereo camera perceived obstacles are colored by height.

In this paper we combine and extend our previous works, DROAN (Disparity-space Representation for Obstacle AvoidaNce) [6] and monocular wire detection using synthetic data and dilated convolutional networks [7], by stitching monocular and binocular stereo observations in a seamless way to enable detection and avoidance of generic obstacles and especially hard to perceive thin obstacles such as wires and power lines, in a novel fashion. For monocular wire detection, we use a semantic segmentation paradigm using fully convolutional neural networks [8] with multiple dilation layers [9] and no downsampling, trained on synthetically generated wires superimposed on real flight images as explained in [7]. For a complete obstacle avoidance system however, a monocular camera is not enough. Hence, we use a binocular stereo pair as a generic obstacle perception sensor. The standard paradigm for doing this involves using the point cloud obtained from the stereo images to maintain and update a 3D occupancy map, where obstacles are then expanded according to the robot's radius to do motion planning by treating the robot as a point-sized obstacle. However, such an occupancy map is expensive to maintain and store in practice. Hence, we use the inverse depth or disparity space as our configuration space [6], [10]. We model the

[1]The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA basti@cmu.edu.

Preprint submitted to 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 1, 2018.

sensor uncertainty in order to do a C-space expansion of the observed data associated with generic obstacles. For estimating the depth of wires, we first threshold and filter the confidence map obtained from the CNN, and make a synthetic disparity image for the wire obstacle with a conservative depth estimate. Then, a pose graph maintained over multiple disparity images is used to infer occupancy for both wire and generic obstacle in order to perform collision checking in the planning stage. For planning, we use a receding horizon planner deploying a precomputed trajectory library composed of dynamically feasible segments to a pre-specified goal location. The precomputed candidate trajectories are then evaluated online to assign a cost based on how close they get to the goal in position, heading and remain free from collisions. Finally a path tracking controller from [6] is used to follow the selected candidate trajectory. This process continues at the rate of the fastest observation sensor i.e. the disparity image frequency of 10 Hz. We test the effectiveness of wire depth estimation in simulation as shown in Figure 6. Consequently, we demonstrate our pipeline on multiple runs in both an outdoor environment with wire obstacles and in an indoor environment with generic obstacles.

II. Related Work

A. Monocular detection

One of the earliest works in wire detection is from Kasturi et al.[11], who extract an edgemap using Steger's algorithm [12], followed by a thresholded Hough transform. Candamo et al.[13] find edges using the Canny detector and then weigh them proportionally according to their estimated motion found using optical flow, followed by morphological filtering and the windowed Hough transform. Song and Li [14] proposed a sequential local-to-global power line detection algorithm which can detect both straight and curved wires. Fully Convolutional Networks [8] proposed learned upsampling and skip layers for the task of semantic segmentation. However, for thin wires, FCNs and similar approaches like SegNet [15] are intuitively suboptimal as crucial information is lost in pooling layers which becomes difficult to localize in the upsampling layers. Dilated or atrous kernels provide a framework to capture an exponentially increasing field of view as the network depth increases by increasing the dilation parameter as explained in [9]. We use the approach of [7] using dilated convnets which are pre-trained on synthetic data and fine-tuned on small amounts for real data for decent results on the test site in practice.

B. Depth or Disparity map based Obstacle Detection

Most approaches generate point clouds from disparity images and generate 3D evidence or occupancy grids to infer occupancy for collision checking [16], [17]. Gohl et. al [18] proposed to use a spherical coordinate based gridmap suitable for stereo sensors, but this requires that each disparity map has to be converted to a 3D point cloud before being injected into the 3D gridmap.

Working with 3D gridmaps is both memory intensive for large occupancy maps, and computationally expensive for registration and book keeping when scrolling or moving the grid along with the robot. OctoMaps [19] have recently become popular due to their efficient structure for occupancy mapping. However, due to excess noise in stereo sensor generated data at long ranges, often a smaller map is maintained and full stereo sensor data is not used. A pushbroom stereo scanning method is proposed in [20] for obstacle detection for MAVs flying at high speeds, however it is capable of only detecting a prefixed disparity which it accumulates as the robot moves in the environment, and therefore is limited to myopic sensing.

We base our work on [10] which proposed a Configuration-Space (C-Space) expansion step to apply an extra padding around disparities based on robot size, but it also fails to use complete sensor data as only the closer occurring obstacles are represented by their method without considering any sensor uncertainty.

III. Approach

We now explain each module in our pipeline.

A. Monocular Wire Detection

We treat wire detection as a semantic segmentation problem in monocular images and leverage our work in [7] for the same. Due to unavailability of a large dataset of synthetic wires, we generate a large number of synthetic wires using a ray-tracing engine [21], [22], and superimpose them on publicly available videos' frames, in order to make an ImageNet analogue for pre-training the network. We then do a grid search over multiple dilated convolutional networks we designed in [7], and pick the best in terms of accuracy and performance on the Nvidia TX-2, specifically the architecture k32-k32-k32-k32-d1-d2-d4-d8-d16-d1-k2 as explained in Table III of [7]. For good performance on our real world experiments, we fine-tuned the network weights on an assortment of a few manually labelled images with real wires we collected ourselves.

The output of the CNN is a confidence map, where each pixel maps to a confidence value $\in [0, 1]$ of that pixel belonging to the wire class. First, we threshold this map and remove pixels with confidence < 0.95. The generated mask at times has false positives in terms of speckles or disconnected regions similar to stereo-sensor generated disparity maps. Since, our current approach is very conservative about avoiding wire obstacles, such speckles are detrimental when planning for avoidance of obstacles, resulting in erratic robot motion, or in the worst case, complete failure to find a plan to avoid the wire. Hence, we run it through a speckle filter to get rid of small non-wire patches. The filtered mask is then treated as the wire-obstacle map as shown in Figure 1. In Section III-C.1, we discuss how we convert this map to a disparity space map for seamless integration into the occupancy inference pipeline.

Preprint submitted to 2018 IEEE/RSJ International Conference
on Intelligent Robots and Systems. Received March 1, 2018.

B. Characterizing Disparity Space Uncertainty

1) Modeling perception error: In order to detect generic obstacles, we use a binocular stereo camera, and in this section we explain the need for using a sensor model and how we develop the same. We use the disparity or inverse depth image for obstacle representation as it naturally captures spatial volume according to the sensor resolution [18], which means that implicitly nearby obstacles have a denser representation, whereas obstacles which are further away are represented with a sparser resolution.

In stereo vision, the depth z of a pixel (u, v) and the corresponding disparity value d are related as:

$$z = \frac{bf}{d} \tag{1}$$

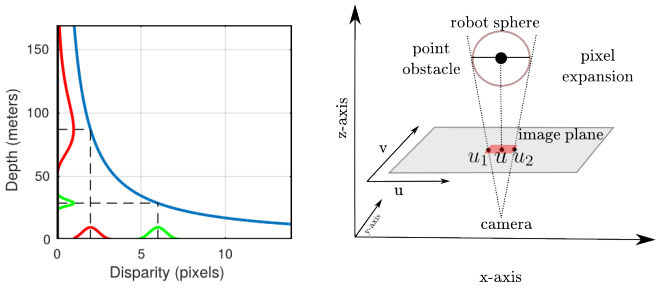
where b is stereo baseline and f is the focal length in pixels. The 3D coordinates of the corresponding point can be expressed as

$$P(x, y, z) = (uz/f, vz/f, z) \tag{2}$$

The accuracy of the stereo setup is drastically affected as the disparity decreases. The error in depth increases quadratically with depth, as can be seen by differentiating equation (1) w.r.t d and then back-substituting for z as explained in [6], i.e. $\partial z \sim z^2$.

It is evident that the sensor model is non-linear in the depth space and has a long tail distribution, as can be seen in Figure 2a. However, in the disparity space, it can be modelled as a Gaussian distribution, $\mathcal{N}(d, \sigma^2)$, where d is the observed disparity, and σ^2 is the covariance parameter of the sensor model which we determine empirically in the next section. The reasoning behind using a Gaussian in disparity space is that disparity error is primarily caused due to correspondence error while matching pixels along the epipolar line. Figure 2a shows that a Gaussian pdf in disparity space captures a difficult to model pdf in depth which has an elongated tail on one side and a compressed tail on the other. The blue curve shows how depth is related to disparity. We can see that a Gaussian pdf centred around a low disparity value (red on X-axis) maps to a long tail distribution (red on Y-axis) in the depth space. Similarly, a Gaussian around a larger disparity value (green on X-axis) maps to less uncertainty in depth (green on Y-axis). This fits well with the fact that disparity naturally captures spatial volume according to the sensor resolution, and establishes our motivation to use disparity image space domain directly for occupancy inference rather than resorting to depth or 3D spatial domain.

2) Finding the sensor model parameters empirically: In this section, we explain how we find the σ value from the previous section, given a disparity image and corresponding ground truth value of depth. To this end, we generate disparity using an FPGA based solution [23], and obtain the ground truth depth value by placing a known chequered board pattern in front of the sensor. Then, we generate a histogram of disparity errors by collecting several samples of disparity values corresponding to each corner pixel. Figure 3 shows our setup at a distance of 2.5 m. Data was sampled at



(a) (b)

Fig. 2: (a) Disparity and corresponding depth values are shown in blue. A Gaussian pdf centred around a low disparity value (red on X-axis) maps to a long tail distribution (red on Y-axis) in the depth space, whereas a Gaussian for larger disparity value maps to less uncertainty in depth space. This explains how we capture difficult to model distributions in the depth space with a Gaussian counterpoint in the disparity space. (b) Shows the pixel-wise area expansion of a point obstacle according to robot size in the image plane.

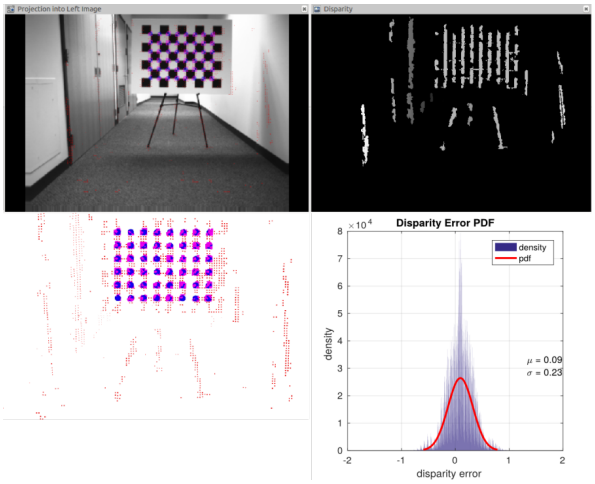


Fig. 3: Experimental setup to collect data for error modelling and the fitted model. Top left, clock wise: Projection of ground truth/known corners into the current image, disparity map, the fitted PDF for disparity error. The point cloud shows the ground truth points in blue and sampled points in pink. The small sized red point cloud is the disparity generated point cloud.

multiple distances to model the error, and we fit a Gaussian distribution to the obtained histogram to obtain a value of $\sigma = 0.23$, as depicted in Figure 3.

C. Configuration-Space Expansion in Disparity Space

Now that we have established a sensor model in disparity space, in this section we explain how we do configuration space expansion for collision checking. We capture the volume occupied by an obstacle using two virtual *limit surfaces*, one each for front and back by generating two corresponding disparity images via the sensor model we developed in the previous section. Each pixel in these two images effectively captures the range of disparity based on the robot size and the sensor error model as shown in the Figure 2a. We do the expansion in two steps : the first one expands disparities along the image XY axes as shown in Figure 2b, and the second step expands along the depth dimension (image Z axis) as shown in Figure 4.

The first step does area expansion of the obstacle pixel (u, v) in the disparity image, to occupy a set of

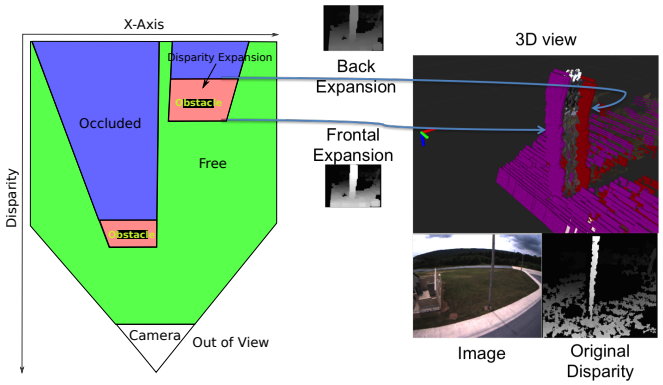


Fig. 4: Disparity expansion shown as point cloud. The pink and red point cloud represent the foreground and background disparity limits.

pixels ranging from $[u_1, u_2]$ to $[v_1, v_2]$ after inflation. This is similar to [10] which introduced a procedure to generate a look-up table (LUT) to obtain the expansion mappings across both image dimensions, $u \rightarrow [u_1, u_2]$ and $v \rightarrow [v_1, v_2]$, given disparity value d. However, we differ from [10] in that we also incorporate the sensor error modelled in the previous section. To ensure robot safety while flying, we introduce another parameter, λ which is a sigma multiplier in the expansion step depending on how far the obstacle is from the robot. The intuition here is that the nearby obstacles are expanded more, whereas the obstacles further away are expanded less, to enable long-range deliberate planning for exploration tasks. Thus, instead of looking up for the raw disparity value d from the LUT as done in [10], we rather look up for $(d + \lambda\sigma)$. By varying λ we ensure safe planning at short range and a more optimistic planning at long range.

The second step in C-space expansion expands disparities in the depth dimension to get values for front and back images using equation (3), as shown in Figure 4. These images represent the maximum and minimum disparities for every pixel respectively:

$$d_\text{f} = \frac{bf}{z - r_v} + \lambda\sigma \quad , \quad d_\text{b} = \frac{bf}{z + r_v} - \lambda\sigma \qquad (3)$$

where r_v is the expansion radius based on robot size, d_f and d_b are the computed front and back disparities which encompass the obstacle. As shown in illustration on left side of Figure 4, the red area around the original disparity of obstacle is the padding generated in the expansion step. This padding is based on the robot size and sensor error model. The reader is advised to refer to our previous work [6] for further details on the algorithm used for C-Space expansion.

1) C-Space expansion of detected wire pixels: As explained in Section III-A, we obtained a thresholded and filtered confidence map of detected wires from monocular images. First, we generate a synthetic disparity image using a prior depth. Given the synthetic disparity we apply the steps from the previous subsection to generate a frontal expansion. For back expansion however, we apply a fixed padding as shown in Figure 5. We test the effectiveness of depth estimation in simulation as shown in Figure 6.

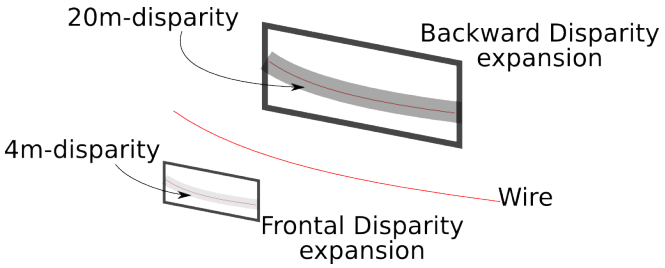


Fig. 5: Synthetic disparity generation and expansion of wire.

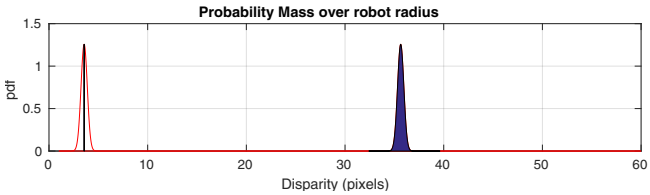
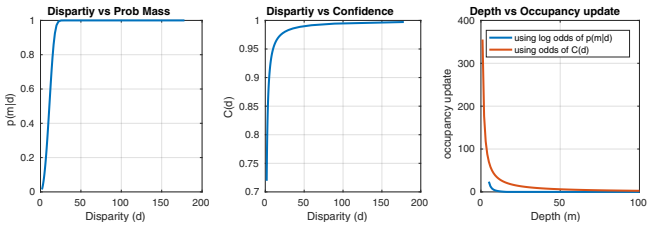


Fig. 7: Probability mass (shown in blue area) occupied by a robot of radius $0.5m$ at a distance of $50m(3.5px)$ and $5m(35.6px)$. As the distance increases or disparity decreases the probability mass occupied by the robot varies.

D. Probabalistic Occupancy Inference

Occupancy inference is the method to derive occupancy of a volume using all the observations or evidence we have. This is a widely studied topic and is often not a trivial problem. Chapter 9 of [24] explains this in detail. Evidence grids or occupancy maps are practical methods for fusion of different measurements taken over time. The proposed pose-graph, explained later in Section III-F, enables similar fusion of multiple observations. In [25] we showed how an inverse sensor model for stereo-cameras can be used for occupancy inference in a probabilistic manner, i.e. similar to log-odds used in evidence grids. We also compared it to a easy to compute *Confidence Function*. Following is a brief discussion, for details please refer [25].

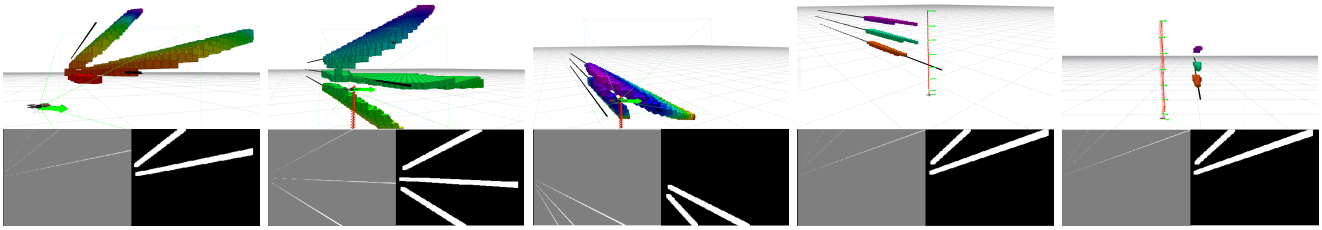
1) Confidence Function for inference in Disparity Space: Using the Gaussian sensor error model as explained earlier we make the following analysis. Figure 7 shows how the occupied volume changes in disparity space given a fixed robot size. Figure 8a shows the probability mass as a function



(a) (b) (c)

Fig. 8: (a) Probability Mass of occupancy, given a robot at disparity d pixels. This curve is used to compute the log-odds to infer occupancy. (b) Disparity vs Confidence plot obtained from equation (4). This is inexpensive to compute online compared to computation of probability mass which involves integration of inverse-sensor-model over robot radius. (c) Occupancy update comparison between log-odds and proposed confidence inference. Confidence based occupancy update is more conservative in nature and will lead to detection of an obstacle using fewer observations. Hence, this function is pessimistic about free-space and ensures an obstacle will be detected in all cases when log-odds also detects an obstacle.

Preprint submitted to 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 1, 2018.



(a) low (b) medium (c) high (d) oblique perspective (e) another perspective

Fig. 6: All snapshots at bottom show the camera view and detected + expanded wires. (a)-(c) show single observations at different heights using our representation. The wire ground truth is shown as black linear point cloud. It is assumed to be uniformly distributed in this space as depicted by colored voxels (by height). (d)-(e) show the final result of our fusion and occupancy inference from oblique and along the wire direction perspectives. Our approach is successful in mapping the three wires. The robot odometry is shown in red and the individual nodes of the pose-graph as green arrows.

TABLE I: Occupancy update

Check	Remark	occupancy cost $occ(d_s)$
$d_s > d_{\mathrm{f}}(u,v)$	safe	$-0.5\frac{C(d_s)}{1-C(d_s)}$
$d_s < d_{\mathrm{f}}(u,v)$ and $d_s > d_{\mathrm{b}}(u,v)$	obstacle	$\frac{C(d_s)}{1-C(d_s)}$
$d_s < d_{\mathrm{b}}(u,v)$	potentially safe	$-0.5\frac{C(d_s)}{1-C(d_s)}$

of inverse depth or disparity and shows that the same Gaussian distribution at different disparities the actual range of disparity that the robot would occupy falls drastically.

However, it is difficult to compute the probability mass online as it requires integration of inverse-sensor-model, hence we propose a new confidence function which is inexpensive to compute online. Given the standard deviation of correspondence error σ, we compute confidence of a disparity state d in the following manner.

$$C(d) = \frac{(d-\sigma)}{d} \tag{4}$$

Confidence measure from equation(4) gives us a measure of how much can we trust a given disparity for occupancy inference. Figure 8b shows how this measure relates to disparity.

We compare the occupancy update using the log-odds probabilistic inference and the proposed confidence inference method. Figure 8c shows the plot of occupancy update using the two methods. The confidence function is more conservative in nature when doing occupancy update at longer ranges. Hence, it is guaranteed that the proposed confidence function will mark a state as occupied if the probabilistic inference also evaluates to the same. We further discount measurements that mark an area safe or potentially safe(occluded) by 0.5 to be more conservative about clearing areas previously marked occupied.

E. Collision Checking

Collision checking is used to plan a new path and to validate if an existing path is safe to follow. Collision checking is performed by projecting the 3D world point $P(x,y,z)$ to image pixel $I(u.v)$ with disparity d_s using equation (1) and equation (2). The point P is projected in all the images that constitute the nodes of the pose-graph and checked for collision as follows.

A state is in collision if the total occupancy measure $\mathcal{M}$ as shown in equation (5) crosses a pre-defined threshold γ.

$$\mathcal{M} = max(\sum_{nodes} occ(d_s), 0) \tag{5}$$

$$Collision = \begin{cases} 1, & \text{if } \mathcal{M} \geq \gamma \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $occ(d_s)$ is computed according to Table I. If the occupancy for a state is below the threshold, we consider that state as not occupied by an obstacle. We also clamp $\mathcal{M}$ to be not negative to prevent over confidence for free volume.

Note: For wire occupancy inference, since the wires are assumed to be uniformly distributed over a region per observation (see Figure 5), we directly use the confidence function $C(d)$ with $\sigma = 0$ instead of its odds to compute the occupancy cost $occ(d)$. This simplifies to summation of observations with wire over a given volume.

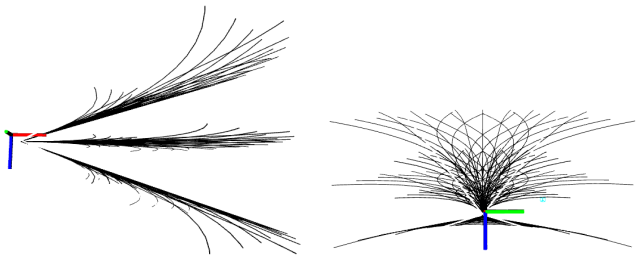
F. Pose Graph of Disparity Images

A single observation is often not enough to construct a reliable occupancy map, hence several observations are fused into a local map enabling local spatial memory. Moreover, stereo cameras only observe the environment in the overlapping field of view. Hence, a spatial memory is required to create a local map of the environment as the robot moves in it. We propose to use a pose-graph of our disparity image based representation to maintain a spatial memory. A pose-graph can further benefit from a simultaneous localization and mapping solution to correctly register the observations [26], and can be later used to generate a global occupancy map. By maintaining a pose graph of expanded disparity images, we can do fusion without building a spatial grid which are not suited for stereo data as discussed previously. Details are provided in [6]

IV. Planning & Motion Control

We implemented a receding horizon planner based on a trajectory library. Trajectory or manoeuvre libraries have been widely used in the robotics community to solve high dimensional control problems such as graph selection or for trajectory set generation for mobile robot navigation [27],[28],[29],[30],[31]. The motivation for using a prior set of libraries is that they effectively discretize a large

Preprint submitted to 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 1, 2018.



(a) (b)

Fig. 9: (a) Side view (b) back view of 3D Trajectory Library used on the robot. The Axes represent the robot pose, red is forward x-axis and blue is down z-axis.

control or planning space and enable good performance within possible computational limits. All candidates in the library are evaluated at runtime, and the one with least cost and free from collision is chosen and executed. However, the performance is hugely affected by the size and content quality or coverage of the library. Size refers to the number of candidates that can be evaluated during runtime and quality or coverage refers to dispersion of the candidates [32]. The main advantage of using such libraries is that they are guaranteed to be dynamically feasible and hence allow smooth motion, or manipulation of the original library.

Figure 9 shows a set of 3D trajectories used on the robot. To evaluate a trajectory, it is split into m waypoints and the following cost function is used to prune the original trajectory:

$$i = \underset{i}{\operatorname{argmin}}\, \alpha\Delta x_i + (1-\alpha)\Delta\theta_i \quad , \quad \texttt{where} \;\; i \in [1, m] \tag{7}$$

$$J(\tau) = \sum_{n=1}^{i} \alpha\Delta x_n + (1-\alpha)\Delta\theta_n \tag{8}$$

where Δx_i is the distance to goal and $\Delta\theta_i$ is the angle offset from goal for the waypoint τ_i. The parameter α is hand tuned to prefer proximity v/s heading offset to goal. The trajectories are checked for collision in ascending order of their traversal cost, by checking each waypoint on the trajectory using the method explained in Section III-E, until a valid trajectory is found. If no valid trajectory is found, the robot is commanded to brake and rotate in place as a recovery manoeuvre to find alternate paths.

V. SYSTEM & EXPERIMENTS

A. System

We developed two MAV systems to conduct the experiments. Both are similar in capabilities, but with different sizes.

For wire avoidance tests, we used a COTS (Commercial Off-The-Shelf) quadrotor (DJI Matrice m100), retrofitted with in-house developed sensor suite consisting of a stereo-camera, an FPGA stereo processor [23], a monocular color camera, an IMU, and a Nvidia TX2 ARM computer as shown in Figure 10. Figure 11 shows the system diagram with the algorithms that run onboard. The FPGA board computes the stereo disparity while the rest of computation is done on the

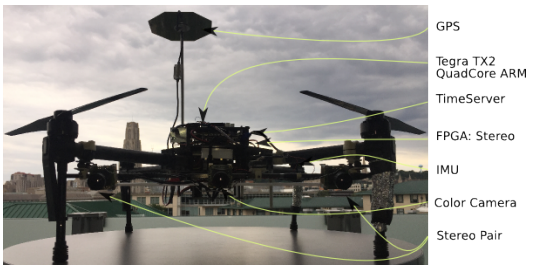


Fig. 10: Robot-1: Quadrotor platform used for experiments: equipped with stereo camera + color (middle) camera sensor suite and onboard ARM computer

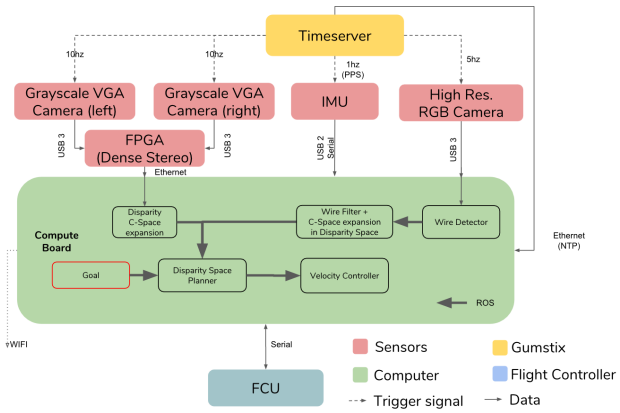


Fig. 11: System diagram of the robot. Shows the hardware and main software components and their data exchange. ROS is used to interchange data between various onboard software components and to provide goal from the ground station.

TX2 board. A ground station computer is used to set the goal or to provide a set of intermediate goal locations.

For regular obstacle avoidance using stereo-camera disparity we used a small COTS base platform (Autel X-Star quadrotor) retrofitted with in-house developed sensing and computing suite consisting of a stereo camera pair, an integrated GPS/INS unit, and an Nvidia TX1, as shown in Figure 12. The stereo camera pair provides 640×512 resolution image which is used to compute a disparity image at 10 fps on the embedded GPU. All computation for autonomous operation is performed onboard.

B. Experiments

We conducted separate experiments for wire and regular obstacle avoidance using trajectory libraries. Two different systems as explained in previous section were used.

We then tested our DROAN mapping approach on regular and wire obstacles combined as shown in Figure 1.

Fig. 12: Robot-2: A smaller quadrotor platform used for experiments: equipped with stereo camera sensor suite and onboard ARM computer

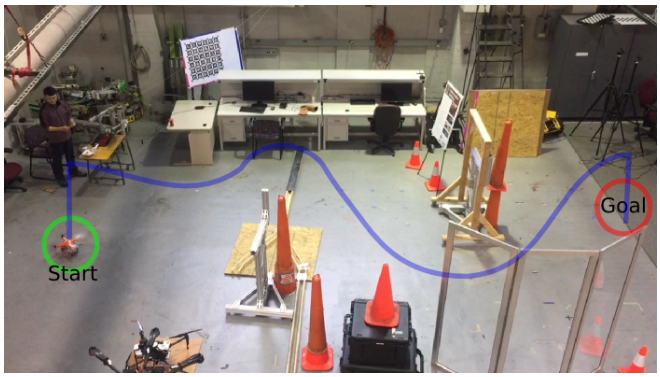


Fig. 13: Demo setup for Robotics Week, 2017 at RI-CMU. The Start is on left and Goal is 10 m away on right with several obstacles restricting a direct path, hence forming a curvy corridor (in blue) to follow. We did more than 100 runs with no failures at 2.5 m/s

1) Wire Avoidance: We suspended a metal wire between two ladders such that it lies directly between the start and goal locations. We first conducted some baseline experiments using COTS DJI-Phantom4 PRO quadrotor. It has a onboard obstacle detection system that uses a stereo-camera sensor.

To validate our approach for wire avoidance, we configured Robot-1 system to only detect and avoid wires. We conducted 10 runs where the goal was set roughly 10 m behind the wire with varying start distances from the wire. The robot was configured to fly at a maximum speed of 2 m/s.

2) Regular Obstacle Avoidance using Stereo-Camera Disparity: We conducted indoor and outdoor experiments using a set of trajectory library. Tests involved autonomous take off, navigate to pre-fixed well separated global waypoints and finally land. For state estimation the EKF based methods from [33] are used. In outdoor experiments, GPS was fused with IMU data to obtain robot state information while for indoor experiments stereo-camera based visual-inertial-odometry was used. Figure 13 shows the setup. The Start and Goal are separated by $10m$ with a curvy path obstructed with multiple obstacles.

VI. RESULTS

A. Simulation Results

To test our wire mapping approach we setup a simulation environment in Gazebo [34] with three horizontal wires stacked 3 m away as shown in Figure 6. The first three snapshots show single observations at different heights using our representation. The wire (ground truth shown as black linear point cloud) is assumed to be uniformly distributed in this space as depicted by colored voxels. The last two snapshots show the final result of our fusion and occupancy inference from oblique and along the wire direction perspectives. It is evident that our approach is successful in mapping the three wires. The voxels are only generated for visualization purpose by creating a 0.5 m resolution, ego-centric gridmap ($60 \times 40 \times 40$ m^3) with respect to the robot. Hence, there are no voxels beyond a certain point displayed in Figure 6. Real world experiments are explained in next section.

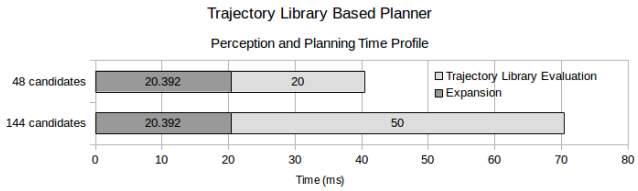


Fig. 15: Time Profile for trajectory library based planner approach on Jetson TX2.

B. Wire Avoidance Results

We first tested with a COTS DJI Phantom 4 Pro drone. It successfully detected regular obstacles but fails to detect wires as an obstacle. Figure 14 shows the process of wire avoidance in Run-8 using our approach. We show the black voxel collision map of wire obstacle only for visualization. For collision avoidance, we only need to perform a collision check for the states traversed by the trajectories. In Figure 14a the robot observed the wire and we show the current collision map. The goal is set approximately 15 m in front of the robot. Figure 14b shows the robot executing the upward trajectory to avoid the wire in its path. Figure 14c, after the robot has ascended enough to fly over the wire, the straight trajectory going towards the goal is selected, and the wire occupancy map gets updated.We obtained 90% success rate as summarized in Table II.

TABLE II: 10 Runs of Wire Obstacle avoidance

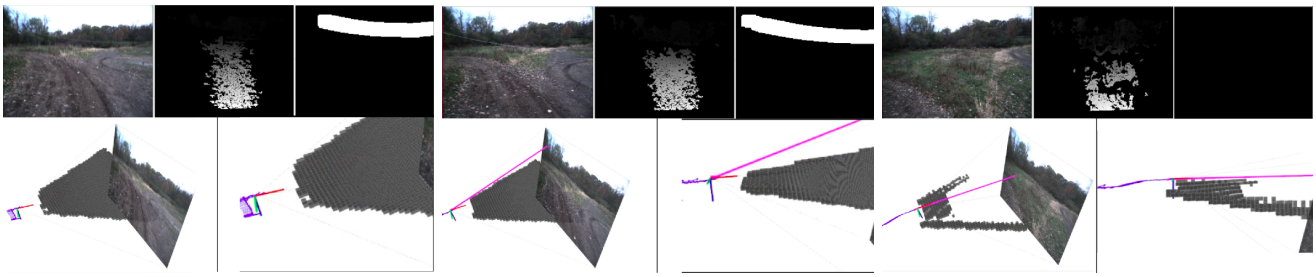
Run	Success	Remark
1	TRUE	Goal sent towards right. Robot tries to avoid wire and moves rightwards
2	TRUE	Pilot took over but the plan was successfully avoiding the wire
3	TRUE	Robot moved straight and up to avoid the wire
4	TRUE	Robot moved straight and up to avoid the wire
5	TRUE	Goal was set off to left direction. Robot flew towards left and over the wires.
6	TRUE	Robot tried to avoid wire by going left but operator took over as regular ladder obstacle was in the way.
7	TRUE	Avoided wire by going around it from left direction.
8	TRUE	Robot moved straight and up to avoid the wire
9	TRUE	Robot moved straight and up to avoid the wire
10	FALSE	Robot was too low and tried to avoid the wire by going straight and under it. Robot GPS antenna boom got stuck in the wire.

C. Disparity Map based Obstacle Avoidance Results

Figure 15 shows the effect on compute times on varying the size of candidates in the trajectory library. In either case we are able to do real-time planning i.e. within the sensor frame rate. Most runs were conducted between the speed of $2m/s$ to $3m/s$. During the Robotics Week, 2017 at RI-CMU more than 100 runs were conducted in front of public with no failures.

VII. CONCLUSIONS & FUTURE WORK

We have presented a novel method and a system which can detect wires from monocular images, estimate their depth and navigate around them. We proposed an efficient method to represent wires and generic obstacles in 2.5D image-space (disparity) suitable for cameras, while accounting for



(a) (b) (c)

Fig. 14: This figure shows three timestamps from run 8 of our real world wire avoidance experiment. Top row shows the input color image used for wire detection, disparity map, and (c-space) expanded wire obstacle map. Bottom row shows corresponding scenarios in different 3D perspectives (oblique and side). The black voxels are shown for visualization of wire collision map. (a) Robot observes the wire. (b) Executes upward trajectory. (c) After ascending upwards enough to avoid the wire, the robot executes a straight trajectory towards a goal 15 m ahead from the start point of the mission. Wire collision map is updated with new observations void of wire.

perception uncertainty, and demonstrated avoidance using a trajectory library. In the future, we aim to propose a mapping framework which accounts for both state estimate and perception uncertainty explicitly.

VIII. ACKNOWLEDGEMENT

This work was supported by Autel Robotics under award number A018532. We also thank Silvio Maeta and John Keller for their invaluable help in conducting the experiments.

REFERENCES

[1] "https://www.skydio.com/."
[2] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, "Aggressive quadrotor flight through cluttered environments using mixed integer programming," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1469–1475.
[3] C. Zhou, J. Yang, C. Zhao, and G. Hua, "Fast, accurate thin-structure obstacle detection for autonomous mobile robots," *arXiv preprint arXiv:1708.04006*, 2017.
[4] J. J. Tarrio and S. Pedre, "Realtime edge-based visual odometry for a monocular camera." in *ICCV*, 2015, pp. 702–710.
[5] L. Liu, D. Ceylan, C. Lin, W. Wang, and N. J. Mitra, "Image-based reconstruction of wire art," *ACM SIGGRAPH 2017*, 2017.
[6] G. Dubey, S. Arora, and S. Scherer, "Droan disparity-space representation for obstacle avoidance," 09 2017, pp. 1324–1330.
[7] R. Madaan, D. Maturana, and S. Scherer, "Wire detection using synthetic data and dilated convolutional networks for unmanned aerial vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2017.
[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.
[9] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.
[10] L. Matthies, R. Brockers, Y. Kuwata, and S. Weiss, "Stereo vision-based obstacle avoidance for micro air vehicles using disparity space," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3242–3249.
[11] R. Kasturi and O. I. Camps, "Wire detection algorithms for navigation," *NASA Technical Report*, 2002.
[12] C. Steger, "An unbiased detector of curvilinear structures," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 2, pp. 113–125, 1998.
[13] J. Candamo, R. Kasturi, D. Goldgof, and S. Sarkar, "Detection of thin lines using low-quality video from low-altitude aircraft in urban settings," *IEEE Transactions on aerospace and electronic systems*, vol. 45, no. 3, 2009.
[14] B. Song and X. Li, "Power line detection from optical images," *Neurocomputing*, vol. 129, pp. 350–361, 2014.
[15] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for scene segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
[16] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing," in *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE, 2011, pp. 2472–2477.
[17] F. Andert and F. Adolf, "Online world modeling and path planning for an unmanned helicopter," *Autonomous Robots*, vol. 27, no. 3, pp. 147–164, 2009.
[18] P. Gohl, D. Honegger, S. Omari, M. Achtelik, M. Pollefeys, and R. Siegwart, "Omnidirectional visual obstacle detection using embedded FPGA," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 3938–3943, 2015.
[19] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9321-0
[20] A. J. Barry and R. Tedrake, "Pushbroom stereo for high-speed navigation in cluttered environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3046–3052.
[21] "Persistence of vision raytracer (version 3.7)," 2004. [Online]. Available: http://www.povray.org/download/
[22] C. B., "Povray rope macro," 2009.
[23] K. Schauwecker, "Sp1: Stereo vision in real time."
[24] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.
[25] G. Dubey, "Droan: Disparity space representation for obstacle avoidance," Master's thesis, Carnegie Mellon University, Pittsburgh, PA, August 2017.
[26] P. R. Florence, J. Carter, J. Ware, and R. Tedrake, "Nanomap: Fast, uncertainty-aware proximity queries with lazy search over local 3d data."
[27] E. Frazzoli, M. A. Dahleh, and E. Feron, "Robust hybrid control for autonomous vehicle motion planning," in *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, vol. 1, 2000, pp. 821–826 vol.1.
[28] M. Stolle and C. Atkeson, "Policies based on trajectory libraries," in *In Proceedings of the International Conference on Robotics and Automation (ICRA*, 2006.
[29] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner, "Grasp planning in complex scenes."
[30] C. Goldfeder, M. Ciocarlie, J. Peretzman, H. Dang, and P. K. Allen, "Data-driven grasping with partial sensor data."
[31] D. Dey, T. Liu, B. Sofman, and J. Bagnell, "Efficient Optimization of Control Libraries." *Aaai*, pp. 1983–1989, 2012.
[32] C. J. Green and A. Kelly, "Toward optimal sampling in the space of paths," *Springer Tracts in Advanced Robotics*, vol. 66, no. STAR, pp. 281–292, 2010.
[33] Y. Song, S. Nuske, and S. Scherer, "A multi-sensor fusion mav state estimation from long-range stereo, imu, gps and barometric sensors," *Sensors*, vol. 17, 2017.
[34] "http://gazebosim.org/."