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

Geetesh Dubey1, Ratnesh Madaan1 and Sebastian Scherer

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.

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
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.

We now explain each module in our pipeline.

### III. Approach

#### 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-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 $v \in [0,1]$ 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.
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

\[
P(x, y, z) = \left(\frac{uz}{f}, \frac{vz}{f}, z\right) \tag{2}
\]

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 \(\sigma^2\) 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 \(\sigma\) 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 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
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, \(\lambda\) 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 \(\lambda\) 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:

\[
d_f = \frac{bf}{z - r_v} + \lambda \sigma, \quad d_b = \frac{bf}{z + r_v} - \lambda \sigma
\]

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.

**Note:** For the experiments in this paper we set \(\lambda = 1\).

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. However, we have no estimate of depth to generate a disparity map, so we assume the wire to be present between 4 m and 20 m distance. Using the 4 m assumption we apply this prior to all wire-pixels and obtain a synthetic disparity image. Using this synthetic disparity we apply the steps from the previous subsection to generate a frontal expansion. For back expansion however, we apply a fixed padding using second assumption of wire being as far as 20 m as shown in Figure 5. These assumptions can be attributed to generation of fixed size solid planes in 3D space and by using multiple such observations we can triangulate the real location of wire within those ranges. The motivation of using these close proximity depths is to avoid wires as early as possible or move wire obstacles out of robot view by changing course of motion. We test the effectiveness of depth estimation in simulation as shown in Figure 6.

D. Probabilistic 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. 3D Evidence grids or occupancy grid 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 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
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 $M$ as shown in equation (5) crosses a pre-defined threshold $\gamma$.

$$M = \max_{\text{nodes}} \sum \text{occ}(d_s, 0)$$  

(5)

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

(6)

where $\text{occ}(d_s)$ 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 $M$ 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 $\text{occ}(d)$. 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.
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 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:

\[
i = \text{argmin}_i \alpha \Delta x_i + (1 - \alpha) \Delta \theta_i \quad \text{, where} \; i \in [1, m] \quad (7)
\]

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

where \( \Delta x_i \) is the distance to goal and \( \Delta \theta_i \) is the angle offset from goal for the waypoint \( \tau_i \). The parameter \( \alpha \) is hand tuned to prefer proximity vs 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 as explained below.

Robot-1: 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 TX2 board. A ground station computer is used to set the goal or to provide a set of intermediate goal locations.

Robot-2: 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 \(640 \times 512\) 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.

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-Phantom-4 PRO quadrotor. It has an 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 \(10\) m 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.

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

<table>
<thead>
<tr>
<th>Run</th>
<th>Success</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TRUE</td>
<td>Goal sent towards right. Avoids wire and moves rightwards.</td>
</tr>
<tr>
<td>2</td>
<td>TRUE</td>
<td>Pilot took over but the plan was successfully avoiding the wire.</td>
</tr>
<tr>
<td>3</td>
<td>TRUE</td>
<td>Robot moved straight and up to avoid the wire.</td>
</tr>
<tr>
<td>4</td>
<td>TRUE</td>
<td>Goal sent towards left. Robot flew towards left and over the wires.</td>
</tr>
<tr>
<td>5</td>
<td>TRUE</td>
<td>Avoids wire by going left. Ladder obstacle was in the way.</td>
</tr>
<tr>
<td>6</td>
<td>TRUE</td>
<td>Avoided wire by going around it from left direction.</td>
</tr>
<tr>
<td>7</td>
<td>TRUE</td>
<td>Robot moved straight and up to avoid the wire.</td>
</tr>
<tr>
<td>8</td>
<td>TRUE</td>
<td>Robot moved straight and up to avoid the wire.</td>
</tr>
<tr>
<td>9</td>
<td>TRUE</td>
<td>Avoid the wire by going under it. GPS boom got stuck in the wire.</td>
</tr>
</tbody>
</table>

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 \(2\) m/s to \(3\) m/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 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 thank Silvio Maeta and John Keller for their invaluable help in conducting the experiments.

REFERENCES

[34] “http://gazebosim.org/”.