DROAN - Disparity-space Representation for Obstacle AvoidaNce

Geetesh Dubey[1], Sankalp Arora[1] and Sebastian Scherer[1]

***Abstract*— Agile MAVs are required to operate in cluttered, unstructured environments at high speeds and low altitudes for efficient data gathering. Given the payload constraints and long range sensing requirements, cameras are the preferred sensing modality for MAVs. The computation burden of using cameras for obstacle sensing has forced the state of the art methods to construct world representations on a per frame basis, leading to myopic decision making. In this paper we propose a long range perception and planning approach using cameras. By utilizing FPGA hardware for disparity calculation and image space to represent obstacles, our approach and system design allows for construction of long term world representation whilst accounting for highly non-linear noise models in real time. We demonstrate these obstacle avoidance capabilities on a quadrotor flying through dense foliage at speeds of up to $4\ m/s$ for a total of 1.6 hours of autonomous flights. The presented approach enables high speed navigation at low altitudes for MAVs for terrestrial scouting.**

I. INTRODUCTION

Micro-aerial vehicles have long promised to be the agile sensing platforms of the future. MAV applications like stealth reconnaissance, search and rescue and cargo delivery etc. need fast aerial vehicles moving autonomously in cluttered environments, at low altitudes [1], [2]. Hence, fast and safe obstacle avoidance has remained an active research area. Achieving safe, autonomous, fast flight through cluttered environments on MAVs, presents two main challenges. *The need for a large sensing horizon to allow for adequate time to detect and avoid obstacles and fast and accurate world representation update for minimal latency in reacting to newly discovered obstacles, (Figure 1).*

Both of these challenges need to be addressed while keeping the sensing and computational payload to a minimum, to allow for maximizing MAV's flight time and agility. However, current state of the art systems that have demonstrated reliable autonomous flights in cluttered environments have either done so through active sensors like lidars [1], sacrificing agility and range of sensing, or have relied on monocular cameras and data driven techniques to provide proof of concept implementation in a specific environment [3]. Both schools of thoughts have led to pioneering demonstrations of obstacle avoidance capabilities of MAV's albeit at low speeds, either due to restricted sensing range or slow world/robot state updates.

Use of a stereo camera pair offers low weight, long range sensing at the cost of increased computation. Barry et. al [4] and Mathies et. al [5] have demonstrated obstacle avoidance at high speeds. While [5] perform motion planning on data observed in a single frame, [4] generate a relatively sparse map leading to a reactive behavior which is likely to be stuck in local minima in complex environments.

Fig. 1: Low altitude high speed navigation for MAVs. The robot makes a turn to go to the goal point and avoids the immediately seen obstacle.

Integrating multiple disparity images, generated by a stereo pair is hindered by its highly non-linear noise model with noise monotonically increasing with distance. We propose an approach to integrate multiple disparity images in a computationally efficient manner to allow for long range, non-myopic obstacle avoidance using stereo data. Our main contributions are -

- Improvement of disparity expansion as suggested by Mathies et. al [5] through incorporating a sensor error model and explicitly using two disparity images to create a front and back mask for obstacles.
- An approach to use multiple disparity images for occupancy inference.
- System design and demonstration of high speed ($4\ m/s$) obstacle avoidance in dense foliage, at heights as low as $1\ m$ AGL (above ground level).

The rest of the paper is structured as follows, Section II presents a short summary of the related work. Section III describes the overview of our planning algorithm. Section IV describes various parts of the obstacle perception and planning algorithm in detail while Section VI describes the MAV system on which the algorithm was tested with

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

experimental results.

II. Related Work

Recently many research groups have ventured into the vision only obstacle detection and avoidance. Most approaches generate point clouds from disparity images and fuse with point clouds from other sensors such as lasers. Appropriate fusion of multimodal sensory data is still a work of active research. Most prevalent approach has been to generate evidence grids or occupancy grids to determine occupancy and do collision checks [6], [7]. Working with 3D gridmaps is both memory intensive for large occupancy maps and require more computation for registration of data and book keeping when scrolling or moving the grid along with the robot. Trade-off between high resolution gridmap vs grid size is another reason why occupancy grids usually cannot be used to map a large volume with higher accuracy. OctoMaps [8] 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.

Gohl et. al [9] propose to use a spherical coordinate based gridmap for stereo sensors but it also suffers from the problem of computationally expensive step of map warping or scrolling as the robot moves.

A pushbroom stereo scanning method is proposed in [4] for obstacle detection for MAVs flying at high speeds. As the robot moves, disparity measurements equal to a fixed value are collected to generate a map of the environment. Since the collected disparity measurements are at a fixed distance, usually not very far to obtain reliable measurements from stereo camera, it is only suitable for short distance planning.

We base our work on [5] which proposed a C-space expansion step to apply an extra padding around disparities based on robot size. The method in [5] works when planning in spaces where the stereo system disparity is not very noisy i.e. in close range making the planning system myopic in nature and prevents long planning horizon.

Although cameras provide rich information about the environment, the aforementioned approaches fall short of utilizing this rich information.

III. Planning Pipeline

Mathies et. al [5] presented an approach to use disparity images generated by a stereo pair for obstacle avoidance. In this approach the occupied pixels in the disparity image obtained from the stereo pair are expanded to account for robot's size. The expanded disparity images are used as a spatial representation to plan collision free paths.

We maintain a similar pipeline to process the disparity images but improve the expansion step through the inclusion of the observation noise model in the disparity expansion. Furthermore, we compute two image expansions; frontal and back to probabilistically capture the occupancy region. Figure 2 and Figure 4 show how the two images capture the pole obstacle. The frontal expansion is shown in pink point cloud and the back expansion in red. We also improve the path planning by using multiple disparity images to infer occupied volumes. The use of multiple disparity images allows the planner to reason about long range obstacles. The improved expansion algorithm and multi-image occupancy inference are presented in section IV-F. Furthermore, all the planned paths end at hover position with zero velocities ensuring safety of the vehicle. Figure 2 briefly shows the planning pipeline.

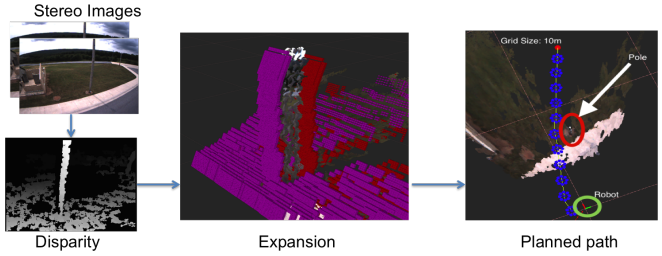

Fig. 2: Planning pipeline based on inverse depth obstacle perception. The frontal expansion and back expansion are shown in pink and red point cloud around the original point cloud of pole. Planned path around the pole is also shown with the current robot position circled in green.

IV. Local Perception & Planning

We use disparity image or inverse depth image for obstacle representation as it naturally captures spatial volume according to the sensor resolution [9]. This representation is befitting for noisy stereo data as explained in Section IV-A. We employ C-space expansion where the original disparity image is expanded, allowing us to treat the robot as a point when doing collision checks during planning [5].

Our method incorporates a stereo sensor error model and allows us to reason about space behind obstacles. We use an additional padding in disparity both in front and behind obstacles. This padding varies from 3σ for close obstacles to 1σ for far obstacles, where σ is the standard deviation of disparity error and the multiplier is represented by λ in later sections. By varying λ we ensure safe planning at short range and a more optimistic planning at long range. This enables the deliberative planning required for exploration tasks.

A. Disparity error and its effects

Disparity is a measure of the proximity of an obstacle. We can derive how close the obstacle is in depth using triangulation in stereo vision as follows.

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

Where, z is the depth of a pixel(u, v) with disparity d, b is baseline and f is the focal length in pixels.

The actual 3D point can be derived 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 shown in equation (5). Differentiating

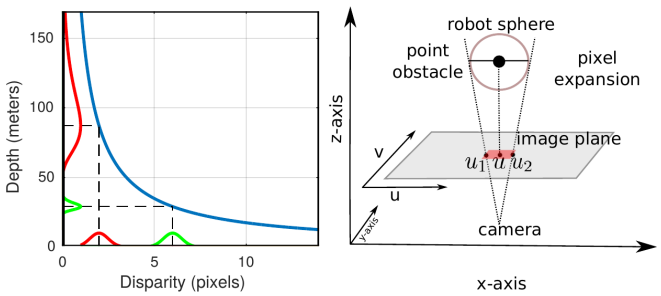

Fig. 3: Left: Disparity vs Depth (blue) and probability distributions are shown in red and green. Red and Green PDF in disparity are same and easy to model but their corresponding Red and Green PDF in range vary and difficult to model. Hence we use inverse depth space to represent obstacles. Also, disparity i.e. inverse range captures space at multi-resolution suitable for registration of stereo sensor data. Right: Shows the pixel-wise expansion of a point obstacle according to robot size.

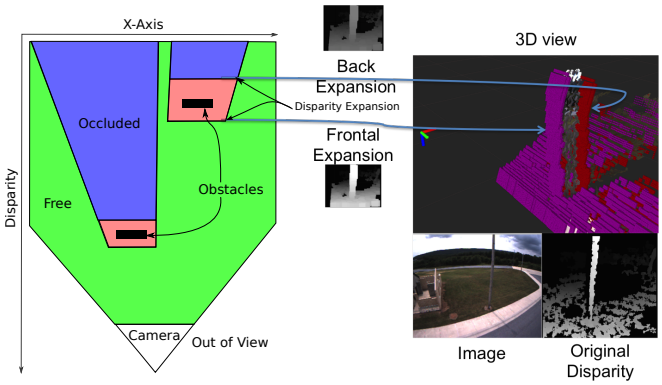

Fig. 4: Disparity expansion shown as point cloud. The pink and red point cloud represent the foreground and background disparity limits.

equation (1) wrt d

$$\frac{\partial z}{\partial d} = -\frac{bf}{d^2} \tag{3}$$

$$\partial z = -\frac{z^2}{bf}\partial d \tag{4}$$

$$\partial z \sim z^2 \tag{5}$$

Disparity error is primarily caused due to correspondence error while matching pixels along the epipolar line. It can be modelled using a Gaussian pdf. Assuming correspondence error during disparity computation has a std deviation $\sigma = 0.5\ pixels$, we define the Gaussian pdf $\mathcal{N}(d, \sigma^2)$. Figure 3 shows how this Gaussian pdf in disparity results in a difficult to model pdf for error in depth with an elongated tail on one side and a compressed tail on the other. This motivates to use disparity image space domain directly for occupancy inference rather than resorting to depth or 3D domain.

B. Configuration-Space Expansion

C-Space expansion is required to represent obstacles such that a single point state query can be used for collision checks [5]. Occupancy grids have been the default methods for registration of sensor data and C-Space expansion for occupancy inference. Usually point clouds are used to populate occupancy grids but point cloud generated using disparity images are highly uncertain at greater depths Figure 9(c) and hence occupancy grid based representation is infeasible. Moreover, 3D occupancy grids require a huge amount of memory to capture the planning workspace and hence fail to incorporate long range measurements available from stereo sensors. To overcome this limitation we use disparity images and apply disparity expansion step explained in section IV-C.

C. Disparity Expansion

In this section we explain the step of C-Space expansion as applied to disparity images. This step allows us to capture the volume occupied by an obstacle using two surfaces represented by two disparity images. These images represent front and back surface limits of the reported disparity. Each pixel in these two images effectively captures the range of disparity based on robot size and the sensor error model as shown in the Figure 3. This process can be divided into two steps.

The first step expands disparities along the image XY axis Figure 3 (right) i.e. an obstacle at some pixel (u, v) after inflation occupies a manifold of pixels from $[u_1, u_2]$ and $[v_1, v_2]$. This is achieved by traversing through the image row-wise first and then column-wise. This is similar to [5] but we also incorporate sensor error. We omit the steps required to generate the look-up-table (LUT) to map $u \rightarrow [u_1, u_2]$ given disparity d and $v \rightarrow [v_1, v_2]$ given disparity d. Reader is advised to refer [5] for generation of the LUT, but unlike looking up for the raw disparity value d from table we look up for $(d + \lambda\sigma)$, where λ is the sigma multiplier dependent on the range as discussed previously in Section IV.

The second step expands disparities to get new values for front and back images using equation (6). These images represent the maximum and minimum disparities for every pixel respectively.

$$\begin{aligned} z &= \frac{bf}{d} \\ d_{\mathrm{f}} &= \frac{bf}{z - r_v} + \lambda\sigma \\ d_{\mathrm{b}} &= \frac{bf}{z + r_v} - \lambda\sigma \end{aligned} \tag{6}$$

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.

Our approach uses the LUT as shown in Algorithm(1) which takes the original disparity image D as input and processes it to generate the expanded frontal and back disparity images D_f and D_g respectively. Note that this algorithm is run twice, once row-wise and subsequently column-wise. Hence, to prevent double expansion in depth, the function $expand(d)$ implements equation (6) with $\lambda = 0$ and $r_v = 0$ for row-wise operation. The function

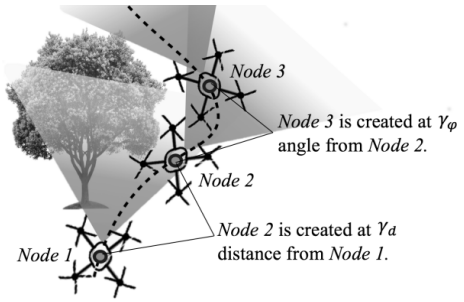

Fig. 5: Pose Graph of expanded disparity images. Dashed path shows robot motion and stored nodes in the graph are shown as triangles. Nodes are stored at intervals of distance and orientation.

$connectedComponent()$ searches for minimum disparity connected to the maximum disparity over steps of provided $range$ (set to a multiple of robot radius). This helps to find an obstacle bounding volume. We do not want to use the minimum disparity in a window as that can be located very far with no connection to the actual obstacle and hence the $connectedComponent()$ step is required.

Algorithm 1: Disparity Expansion Algorithm

Input: Disparity image D
Output: Expanded disparity images: D_f, D_b

for $v = 1 : Height(D)$ **do**
  for $u = 1 : Width(D)$ **do**
    $\hat{d} = ceil(D(u,v) + \lambda\sigma)$
    $[u_1, u_2] = LUT(u, \hat{d})$
    $V = D(u_1 : u_2, v)$
    $d_f = expand(max(V))$
    $d_b = expand(connectedComponent(d_f, range))$
    for $i = u_1 : u_2$ **do**
      $D_f(i,v) = \max(d_f, D_f(i,v))$
      $D_b(i,v) = \min(d_b, D_b(i,v))$
    end
  end
end

Next column-wise expansion is applied with λ and r_v set to default values as specified in Table II when using $expand()$ function to get correct final expansion.The expanded disparity images constitute a single snapshot volumes occupied by obstacles. To maintain a spatial memory we create a pose graph consisting of multiple expanded disparity images as described in the following section.

D. Pose Graph of Disparity Images

The motivation to maintain spatial memory of the previously seen environment as the vehicle is moving using a pose graph is because of the following reasons:

1) Previously seen obstacles might not be visible in the current image.
   a) The stereo sensor has a minimum range dependent on maximum perceivable disparity.
   b) Obstacles get occluded in different views.
   c) The field of view is limited.
2) Maintain a pose graph of disparity images (measurements) with nodes at regular intervals of distances and angles as shown in Figure 5.
3) Allows occupancy inference using multiple measurements.

Algorithm(2) shows how we construct this graph. Each

Algorithm 2: Pose Graph Algorithm

Input: $D_f, D_g, Pose, N_{graph}, \gamma_d, \gamma_\psi$
Output: Pose Graph of Expanded disparity images: $Graph$

$T_s^w \leftarrow Pose$
$Node = createNode(T_s^w, D_f, D_b)$
if $Graph.size() == 0$ **then**
  $Graph.push\_front(Node)$
  $Graph.push\_back(Node)$
end
$PrevNode = Graph.begin()$
$pos\_err = distance(Pose, PrevNode)$
$ang\_err = angle(Pose, PrevNode)$
if $pos\_err >= \gamma_d \| ang\_err >= \gamma_\psi$ **then**
  if $Graph.size() == N_{graph}$ **then**
    $Graph.pop\_back()$
  end
  $Graph.push\_front(Node)$
end
$Graph.pop\_back()$
$Graph.push\_back(Node)$

node in the graph is comprised of the following:

1) D_f
2) D_b
3) T_s^w which is the transform between the processed sensor measurement(D_f, D_b) and world frame.

The algorithm takes as input the current robot position $Pose$, processed disparity images D_f, D_b, maximum number of nodes N_{graph} and two tolerance parameters γ_d, γ_ψ for position and angular displacement respectively. The constructed graph is used to project a given world point into all node images and do occupancy inference. Occupancy inference using the set of disparity images in the graph is explained in subsequent section.

E. Occupancy Inference

Evidence grids or occupancy maps are methods to allow fusion of different measurements taken over time. By maintaining a pose graph of expanded disparity images, we can also take advantage of similar fusion without building an occupancy grid which are not suited for stereo data as discussed previously. We devised an occupancy inference method by fusing information from all the images in the graph using the stereo sensor error model. Given the standard deviation of correspondence error σ, we compute confidence of a disparity state in the following manner.

$$C(d) = \frac{(d - \sigma)}{d} \tag{7}$$

Confidence measure from equation (7) gives us a measure of how much can we trust a given disparity for occupancy inference. Thus, long range or low disparity, uncertain measurements have low confidence and update the occupancy with lower values. In the experiments we further discount measurements that mark an area safe or potentially safe(occluded) by a value of 0.5 which we picked empirically, to be more conservative about clearing areas previously marked occupied. It should be noted that the potentially safe

areas are behind obstacles and have lower disparity state, hence their contribution to occupancy clearance is less due to lower confidence value. The final occupancy measure is obtained by projecting a world point P using equation (8) and equation (2) in disparity images of all nodes in the graph and accumulating the occupancy cost according to Table I:

TABLE I: Occupancy update

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

F. 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 using the following mapping of a 3D world point P to image pixel I with disparity d_s:

$$P(x,y,z) \leftrightarrow I(u,v,d_s) \tag{8}$$

A state is in collision if the total occupancy measure $\mathcal{M}$ as shown in equation (9) crosses a pre-defined threshold γ.

$$\mathcal{M} = max(\sum_{nodes} occ(d_s), 0) \tag{9}$$

$$Collision = \begin{cases} 1, & \text{if } \mathcal{M} \geq \gamma \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

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.

G. Planning

We use a sampling based planner, BIT* [10] to draw samples in 3D space which are checked for collision as described in IV-F. The output is a collision free path connecting start to goal state.

In our experiments we found that disparity images fluctuate around obstacle edges leading to unwanted replanning due to the current plan being in collision. To remedy this we used two threshold values. A lower value γ_{low} is used during planning to find a path i.e. obstacles are observed sooner even at long distances and hence a more conservative path is obtained. A higher threshold value γ_{high} is used to check the current plan for collision and do replanning in case of collision. The advantage of using two threshold values is that an initial plan is found using a more conservative occupancy map while the replanning is done using a more reliable occupancy map. The reliable occupancy map is not affected by fluctuations in the disparity maps. The thresholds are chosen such that collisions at close range are always detected but have great advantage to not force replanning due to less reliable and fluctuating observations at long range when planning paths to longer distances. In our experiments we have planned paths at distances longer than $100m$ (Figure 9). Figure 2 shows a planned path that avoids a pole obstacle. This path is sent to the motion controller of the vehicle.

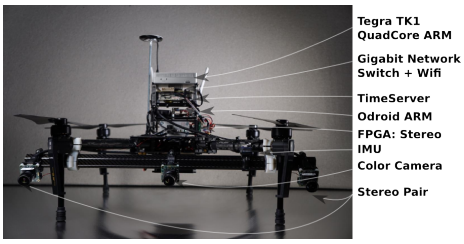

Fig. 6: Quadrotor platform used for experiments: equipped with stereo camera sensor suite and onboard ARM computer

H. Motion Control

We developed a path tracker similar to [11]. It takes the current trajectory and uses feed-forward velocities specified in the trajectory and generates final velocity and heading rates for the low level velocity controller. The low level velocity controller runs on the quadrotor's flight control unit. For more details reader is advised to read [11] Section IV-B and Section V-A.

V. SYSTEM & EXPERIMENTS

We test our algorithms on-board an autonomous UAV system, see Figure 6. The base platform is an off-the-shelf DJI Matrice m100 quadrotor vehicle retrofitted with in-house developed sensing and computing suite designed for semantic exploration. The sensor suite consists of a monochrome stereo camera pair, a monocular color camera, an integrated GPS/INS unit and a barometer. The stereo camera pair provides 640×480 resolution disparity image at 10 fps for the obstacle avoidance and 3D mapping systems. The central camera is operated at a lower frame rate, to provide high resolution color imagery for the semantic perception system. All cameras are forward-facing, tilted downwards at 15°, an orientation well suited for low-altitude ($< 40\ m$) operation. The GPS/INS system and the barometer are used for state estimation.

All computation for autonomous operation is performed onboard. To this end we equip the MAV with two embedded ARM computers; one of them is devoted primarily to planning tasks, while the other is devoted to perceptual tasks. In addition, we use a specialized FPGA processor [12] for stereo depth computation. The computers are networked through high-speed ethernet.

We conducted most of the experiments in the highlighted area shown in Figure 7. Some of the features of region were narrow trails, dense foliage and varying height tree line, all of which made for challenging and interesting obstacles. Tests involved manual take-off and sending a random goal point as shown in Figure 7 or a list of sparse global waypoints to the obstacle avoidance system with the desired velocity. Sparse global waypoints were selected to force navigation through obstacle populated areas. Table II lists the values we used for

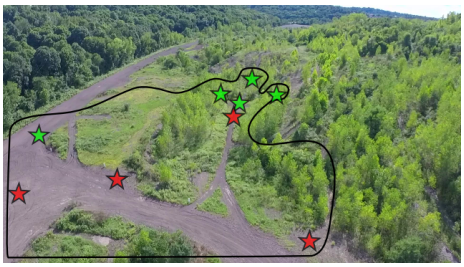

Fig. 7: Marked area of the location where experiments were carried out. Various start and goal locations are shown in green and red star markers respectively. Runs were conducted between random pairs of the start-goal locations and sometimes by creating a sparse waypoint list using the marked goals.

conducting the experiments. For field experiments we fixed $\lambda = 1$ instead of setting it to a function of depth. Sensor error σ was determined empirically by analysing sensor error.

TABLE II: Parameters Used

Parameter	Value
Baseline: b	0.35m
Focal length: f	514.17 pixels
Correspondence error: σ	0.5
Sigma multiplier: λ	1.0
Connected component range: $range$	$2r_v$
Robot radius: r_v	1.5m
Lenient Occupancy Threshold: γ_{high}	1.8
Strict Occupancy Threshold: γ_{low}	0.9
No. of nodes in Pose graph: N_{graph}	10
Displacement between nodes: γ_d	1.5m
Angle between nodes: γ_ψ	30°

VI. RESULTS

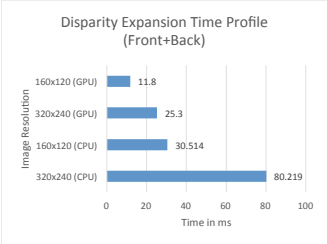

Fig. 8: Time profile of expansion step at different resolutions on Jetson TK1 arm computer. On recent Jetson TX2 computer it takes 20 ms on CPU for 320×240 resolution.

Figure 8 shows the time taken to process a single disparity image to compute the frontal and back expansions using Algorithm(1) on the onboard ARM computer. In our experiments we used CPU version at 320×240 resolution because the GPU was used for semantic classification algorithm as concurrent part of the experiments. A pose graph using Algorithm(2) was created and used for collision checks using equation (9). Using our approach a single occupancy inference and collision check takes on average 0.01 ms. Given 100 ms between each frame we can do about 2000 collision checks which was usually sufficient for the BIT* planning algorithm.

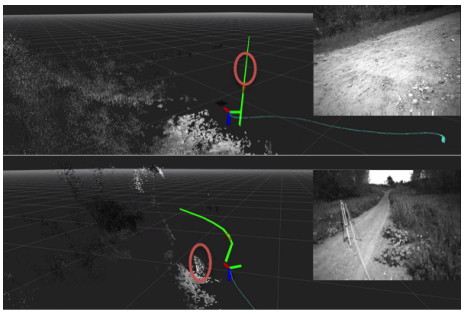

Fig. 10: Reactive Planning at 4 m/s: Top image shows the robot has planned to go right with unseen obstacle marked in red ellipse. Bottom image: after banking right an obstacle obstructs the previous plan and a new plan avoiding it is generated.

Figure 9(a) Shows planned path going through two low height trees. The top left is the disparity image with left camera image shown on top right. The point cloud is only for visualization purpose and the trees are marked in ellipses. Although the trees are not completely visible in the current disparity image, they are still a part of obstacles as they were seen at previous robot positions and hence stored in the pose graph. Without the pose graph these trees would have been invisible to the robot. Thus the pose graph helps in keeping memory of obstacles which were seen previously but can't be observed as they exceed the limit of maximum possible disparity after robot motion.

Figure 9(b) shows the previous path was replanned and pushed up as more observations of the bushes/trees are made at long range are marked as obstacles at approximately 30 m distance from the robot. This was possible due to fusion of occupancy using several disparity images in the pose graph.

Figure 9(c) emphasises the advantage of planning in disparity space at long distances. At greater distances the point cloud is very noisy but we are able to get some information about occupancy by using all the sensor data. While occupancy grids would have huge impact, both memory wise and computationally to use all this data, our approach is able to incorporate all the information using minimalistic image space representation and do better occupancy inference.

Figure 10 shows the reactive nature of our approach. For this experiment the robot was allowed to find a plan outside the sensor's field of view and was given a goal point in right direction. As the robot follows the plan and turns right, an obstacle obstructing its path is detected and a new plan avoiding it is generated. This happened at a speed of 4 m/s hence implying our approach quickly reacts to newly seen obstacles. Figure 1 shows the third person view of the same run.

Using the parameters specified in Table II, if the robot moves 15 m maintaining 10 nodes and assuming a maximum of only 100 m depth (1.79 $pixel$ disparity) per image our approach uses approximately 38% of the memory required by a gridmap of cell size 1 m^3 covering the same volume. This is the case when using a gridmap of large cell size meaning a very coarse resolution. For a better resolution gridmap will require even more memory.

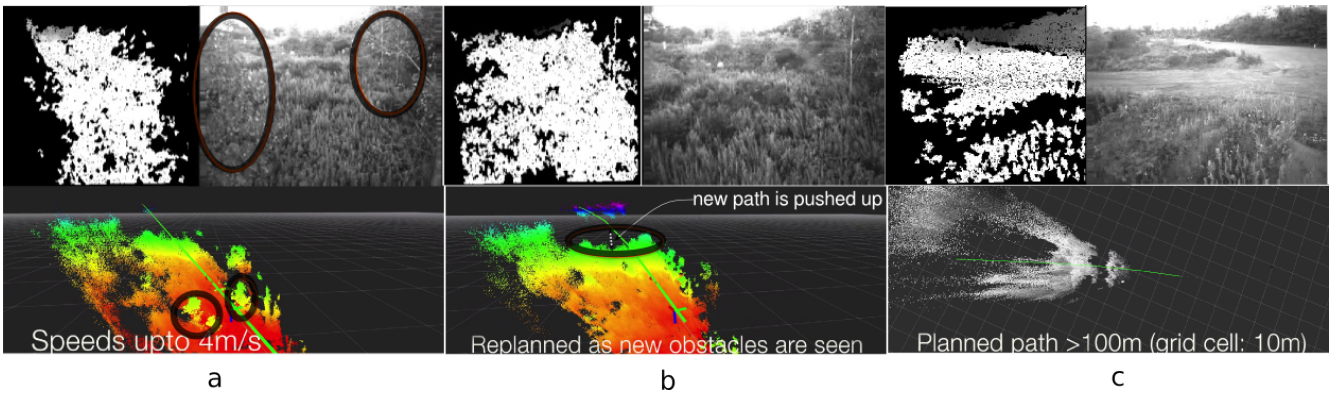

Fig. 9: All figures: Top Left (disparity image), Top Right (left camera image), Bottom (Point cloud for reference). Point cloud is colored by height in (a) & (b) and by actual intensity in (c). (a) Planned path(green) between low trees highlighted in ellipses (b) Replanned(green path) as more observations are made, marked in ellipse, (c) Long range planning horizon. The point cloud shows the noisy measurement but even noisy information allows to infer occupancy at long distances.

More than 100 successful runs were executed with approximately 1.6 hours in autonomous mode, covering a cumulative distance of approximately 1.5 km. The maximum speed was capped at 4 m/s. Our approach allowed us to plan to distances greater than 100 m as shown in Figure 9(c). Average distance to goal was 36 m. The standard deviation of length of planned paths from straight line paths was on average of 1.38 m with a maximum of 30 m. This shows that in most cases planned paths were close to a straight path but with slight deviation to avoid obstacles.

VII. CONCLUSIONS & FUTURE WORK

We have presented an approach and a system design that allows high speed, non-myopic obstacle avoidance. We demonstrated the system flying at 4 m/s in dense foliage while relying on stereo image data for modeling the world. To our knowledge it is the first one to do so.

The key factor that enables our system to perform safely at high speeds in highly cluttered environments is integrating multiple stereo sensor frames in real time while reasoning about the related highly non-linear noise model to generate a world representation in inverse depth space. Using FPGA hardware for disparity calculation, combined with fast disparity expansion allowed us to limit our computational burden. This allowed the system to share computation with classification and state estimation tasks.

The current disparity space representation lacks an explicit model of unknown space, rendering the system vulnerable to collision whilst operating in environments with complex geometries. We are currently working on this issue by performing disparity space expansion over multiple layers whilst guaranteeing vehicle safety using emergency maneuver libraries [13] in conjunction with active control of heading along the lines of the sensor planning approach suggested in [14].

REFERENCES

[1] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: state estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 33, no. 1-2, pp. 189–214, 2012.

[2] S. Choudhury, S. Arora, and S. Scherer, "The planner ensemble and trajectory executive: A high performance motion planning system with guaranteed safety," in *AHS 70th Annual Forum, Montre al, Que bec, Canada*, Pittsburgh, PA, May 2014.

[3] D. Dey, K. S. Shankar, S. Zeng, R. Mehta, M. T. Agcayazi, C. Eriksen, S. Daftry, M. Hebert, and J. A. Bagnell, "Vision and learning for deliberative monocular cluttered flight," in *Field and Service Robotics*. Springer, 2016, pp. 391–409.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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

[9] 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.

[10] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Bit*: Batch informed trees for optimal sampling-based planning via dynamic programming on implicit random geometric graphs," *CoRR*, vol. abs/1405.5848, 2014. [Online]. Available: http://arxiv.org/abs/1405.5848

[11] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, 2008, pp. 1–14.

[12] K. Schauwecker, "Sp1: Stereo vision in real time."

[13] S. Arora, S. Chowdhury, D. Althoff, and S. Scherer, "Emergency maneuver library - ensuring safe navigation in partially known environments," in *IEEE, International Conference on Robotics and Automation*, 2015.

[14] S. Arora and S. Scherer, "Pasp: Policy based approach for sensor planning," in *IEEE, International Conference on Robotics and Automatio*, 2015.