https://doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-015

A Sense and Avoid Algorithm using Surround Stereo Vision for Drones

G. M. Dilshan P. Godaliyadda [1,2]***, Do-Kyoung Kwon*** [2]***, Darnell Moore*** [2] ***and Victor Cheng*** [3]***;***
[1] ***School of ECE, Purdue University, West Lafayette, IN, USA;*** [2] ***Perception and Analytics Lab, Texas Instruments, Dallas, TX, USA;***[3] ***Automotive Processor BU, Texas Instruments, Dallas, TX, USA;***

Abstract

As applications of drone proliferate, it has become increasingly important to equip drones with automatic sense and avoid (SAA) algorithms to address safety and liability concerns. Sense and avoid algorithms can be based upon either active or passive sensing methods. Each of them has advantages when compared to the other but neither is sufficient by itself. Therefore, especially for application such as autonomous navigation where failure could be catastrophic, deploying both passive and active sensors simultaneously and utilizing inputs from them become critical to detect and avoid objects in a reliable way.

As part of the solution, in this paper, we present an efficient SAA algorithm based on input from multiple stereo cameras, which can be implemented on a low-cost and low-power embedded processor. In this algorithm, we construct an instantaneous 3D occupancy grid (OG) map at each time instance using the disparity information from the stereo cameras. Then, we filter noise using spacial information, and further filter noise using a probabilistic approach based on temporal information. Using this OG Map, we detect threats to the drone in order to determine the best trajectory for it to reach a destination.

Introduction

Sense and avoid algorithms for drones fall in to two primary categories — algorithms based on active sensing methods and algorithms based on passive sensing methods. Active sensing methods such as radar and ultrasonic sensors can provide very accurate range information from surrounding objects and structures, but are susceptible to interference clutter, and generally have a more limited field of view or poorer spatial discrimination. On the other hand, passive sensors such as stereo cameras have wider field of view and are not susceptible to interference from other sensing sources, but they are hampered by chaotic lighting and weather conditions and require ample computational power to process images. A solution to this problem is having the best of both worlds, i.e., combining both methods either as two redundant SAA algorithms or as an SAA algorithm that utilizes inputs from both types of sources.

In this paper, we focus on an efficient SAA algorithm for a drone based on stereo vision, which is a passive sensing method. In stereo vision, we image a point in space using two cameras that are aligned with each other to find the disparity between the same point in the two images. Then this disparity information together with the camera's intrinsic parameters allow us to triangulate the exact 3D position of said point [1]. Hence, using one or more stereo camera pairs we can construct disparity images, and from them, a 3D point cloud of objects in the surrounding space. There are numerous algorithms that allow us to compute a disparity map from stereo camera inputs. In this work, we use a local block matching method on census transformed inputs since it provides a relatively good disparity map without excessive uses of computation and memory.

There are SAA algorithms based on stereo vision, which are efficient both in memory and computation. One such algorithm is the algorithm presented by Oleynikova et al. in [2]. In this algorithm, the authors utilize a U-disparity map and a V-disparity map to identify objects and plan a path for the drone. Even though this algorithm is fast and memory efficient, the U and V disparity maps only provide the horizontal positions of objects. Another class of SAA algorithms based on stereo vision use occupancy grid (OG) maps. OG maps were first introduced in [3, 4]. As the name states, in this method, space surrounding the drone is divided into grids and each grid is labeled as occupied or as free. Then SAA algorithms plan a path along free space using this information. In most of these algorithms, a probabilistic global OG map is created with respect to the world frame and kept in memory. This OG map is then updated using new point cloud information that a drone gathers at each instance in time. Examples of such methods include [5, 6, 7, 8, 9]. To update this global OG map one needs to geo-reference each new point in the point cloud. Therefore we need to estimate with a high degree of precision the pose and position of a drone. However, it is very difficult to estimate them accurately due to the many degrees of freedom the drone has. Such errors in turn could result in erroneous OG map. On the other hand, since a global OG Map needs to be kept in memory the memory requirements are high as well [10] and furthermore since the point cloud needs to be geo-referenced another layer of computation will be added as well.

In this paper, we present a stereo vision based SAA algorithm in which we utilize an instantaneous non-uniform 3D OG map to identify threats to the drone and find a safe trajectory for it. The non-uniformity in the 3D OG map is introduced to reduce the memory needed to store the data and to reduce the computation complexity. We use an instantaneous OG map so that we can avoid having to geo-reference the point cloud. Also we can save memory since we do not need to save a large global OG map. To mitigate the effect of erroneous detection, we introduce a probabilistic object tracking algorithm that tracks the displacement of objects relative to a drone. In particular, we

find which objects in the present OG map are likely given the previous OG map by computing the probability of the relative displacements that could have resulted in said objects. The parameters of the relative displacement distribution are updated on the fly in order to ensure it describes the typical behavior of objects.

Once threats are identified in the *safety bubble*, which is the region around a drone where no object should exist, an avoid command is generated so that the drone can move to unoccupied space. When finding the best location for the drone to move to, we consider the locations of all detected objects in a region of interest (ROI) surrounding the drone as well as the direction of travel. Experimental results show that the proposed algorithm can detect real threats very accurately while tolerating noisy data. The proposed algorithm is also appropriate for implementation on an embedded processor because of efficient use of memory and computation.

Object Sensing Algorithm

In this section, we describe our algorithm for object sensing. We use stereo image pairs to compute a disparity map, which we in turn convert to a 3D point cloud.

To generate the disparity map, we use a local block matching method on rectified and census transformed left and right images. In our implementation we first apply the census transform [11] on both input images with a census window of size $n_c \times n_c$. Then, for every pixel in the base image (e.g. left image), we find the corresponding pixel on the same line of the reference image (e.g. right image) by identifying the pixel that results in lowest Hamming distance within an $n_h \times n_h$ window centered at the pixel of interest. The output disparity map is post-processed through left-right check and texture based cleaning, etc. to get higher fidelity disparity map. From the disparity map, d, the depth Z can be calculated for every pixel by

$$Z = \frac{f \cdot b}{d}, \tag{1}$$

where f is a focal length of cameras and b is the baseline distance between left and right camera centers. Then the X and Y positions in 3D space of each pixel (x, y) can be calculated by

$$X = \frac{1}{f} \cdot (x - d_x) \cdot Z \text{ and } Y = \frac{1}{f} \cdot (y - d_y) \cdot Z, \tag{2}$$

where (d_x, d_y) is a image distortion center. It should be noted that 3D positions are with respect to the base (e.g. left) camera center.

We use this 3D point cloud to generate a 3D OG map. Then using information from one or more OG maps from previous time instances we find which occupied grids in the OG map are in fact *probable* and which ones are *improbable*. Then we use this clean OG Map to determine if there are any objects in the safety bubble of the drone. In the proceeding sections we will explain in detail the steps of the sensing algorithm.

Constructing a 3D Non-Uniform Occupancy Grid

Since all computations need to be performed on an embedded processor mounted atop a drone, we have limited memory and computational power. Therefore, instead of a uniformly spaced OG map, we construct a non-uniform OG map in which the fineness of the grid spacing depends on the relative importance of the space. In this implementation we give more precision (finer grid spacing) to the immediate vicinity of the drone and relatively less precision (coarser grid spacing) to the space farther away from the drone. For the remainder of the paper, we will refer to the area with higher uniform precision where we perform tracking as the region of interest (ROI).

To construct the 3D OG Map, we place all the 3D points within the OG map and count the number of points in each grid. If the number of points within a given grid exceeds a user selected (or empirically determined) threshold, we consider that grid to be *occupied*. For the remainder of this paper we will refer to an occupied grid as an *object*.

The next step is to filter out noise in the OG map. In our implementation, we only focus on noise filtering within the ROI so as to minimize computation. We remove noise by using a connected component analysis based thresholding method. In particular, we first perform connected component analysis in the ROI and then filter out connected components that are smaller than a user selected threshold. For example, if the threshold is n, each connected component that has fewer grids than n is considered noise and hence removed. By performing this step, we can remove *isolated* objects that are deemed too small. It should be noted that n is determined based on the grid size.

Our next task is to identify which objects in this filtered 3D OG map are in fact probable given OG map of a previous frame. In essence we want to further clean up the OG map, but using temporal information as well.

Filtering Noise in Instantaneous 3D OG Map using Temporal Information

In this section we describe how we further clean up the OG Map using a probabilistic approach based on temporal information. For this purpose, we consider each occupied grid as an independent object and then use the instantaneous OG map constructed in the previous time instance to identify if each object is probable. The reason we consider each occupied grid as independent objects instead of considering clustered grids as objects is because the number of grids an object occupies can change depending on its position in space. For example, an object that occupied one grid at a certain time instance can move to the boundary of 4 grids and can result in 4 occupied grids in the next time instance. Now we explain in detail the steps involved in the temporal filtering process.

Let us start by denoting the present time as n. Assume that we have constructed a 3D OG map as described previously for time n. Also assume there are N_n different objects $\{O_{1,n}, O_{2,n}, \ldots, O_{N_n,n}\}$ in the ROI of this 3D OG map, i.e. N_n grids are classified as occupied. Out task is to identify the *probable* objects among these N_n objects.

For this purpose, for an object $O_{i,n}$, where $i \in \{1, 2, \ldots, N_{n-1}\}$, we look at the previous time instance $(n-1)$ to identify if any

of the objects in the previous OG map could have been object $O_{i,n}$. That is, we find if any object in $(n-1)$ could have moved to where $O_{i,n}$ is in n.

Suppose that the object which could have resulted in $O_{i,n}$ is $O_{i_j,n-1}$, where $O_{i_j,n-1}$ is the object in the previous frame that is closest to $O_{i,n}$ when the OG map at n is overlaid on the OG map at $(n-1)$. Therefore, $O_{i_j,n-1}$ was what resulted in object $O_{i,n}$ if $d\left(O_{i,n},O_{i_j,n-1}\right) \leq d\left(O_{i,n},O_{k,n-1}\right) \forall\, k \in \{1,2,\ldots,N_{n-1}\}$, where $d(\cdot,\cdot)$ is the displacement of an object relative to the drone from one frame to the next. Figure 1 shows a 2D example illustrating how we find corresponding pairs of objects across two frames.

Once we make the correspondence between $O_{i,n}$ and $O_{i_j,n-1}$ we compute the likelihood of that event. In this algorithm, we compute the probability that $O_{i_j,n-1}$ was displaced relative to the drone by at least $d\left(O_{i,n},O_{i_j,n-1}\right)$, i.e.,

$$p\left(d \geq |d\left(O_{i,n},O_{i_j,n-1}|\right)\right). \tag{3}$$

Then we threshold this likelihood to distinguish between probable and improbable objects, i.e., the object $O_{i,n}$ is deemed probable if

$$p\left(d \geq |d\left(O_{i,n},O_{i_j,n-1}\right)|\right) \geq T. \tag{4}$$

Otherwise, it is deemed improbable.

It is important to mention that when we consider objects from the previous frame to determine which ones could have resulted in the objects in the current frame, we consider improable objects in the previous frame as well, i.e. objects in $(n-1)$ that were classified as improbable based on the objects in $(n-2)$. The reason is because an object that was classified improbable might actually be probable, and by including them for the next instance we can correct our mistake.

Modeling the Relative Displacement Distribution

In order to compute the probability in Equation 4, we need to model the distribution of the relative displacement d. We model d as a Gaussian random variable with zero mean and σ^2 variance, which will in effect distinguish between probable and improbable objects. Therefore, we recompute σ at every time instance so that it reflects the *typical relative displacement* of objects.

To compute σ for time n, we first consider a time instance $(n-m)$ in the past. For each object in frame $(n-m)$, i.e., for $O_{1,n-m},O_{2,n-m},\ldots,$ *and* $O_{N_{n-m},n-m}$, we find the objects in time $(n-m-1)$ that are closest in the sense of relative displacement. In other words, we find for each object $O_{i,n-m}$ in frame $(n-m)$ object $O_{i_j,n-m-1}$ in frame $(n-m-1)$ that corresponds to the minimum relative displacement. Then from this set we find the maximum value d_{n-m}^{max}, i.e.,

$$d_{n-m}^{max} = \max_{i\in\{1,2,\ldots N_{n-m}\}} \left\{d\left(O_{i,n-m},O_{i_j,n-m-1}\right)\right\}. \tag{5}$$

The value of σ for n is computed by

$$\sigma = \frac{1}{L}\sum_{m=1}^{L} max\left\{d_{n-m}^{max},D_{grid}\right\}, \tag{6}$$

where D_{grid} is the grid spacing in the uniformly spaced ROI.

Now we can identify if there are objects in the safety bubble that threaten the drone. Our next task is to find how the drone can best avoid these objects while staying on-course towards a destination.

Object Avoidance Algorithm

The primary objective of our work is to preserve the integrity of the safety bubble surrounding the drone. Therefore, out of the locations that have sufficient free space to fit a safety bubble, we find the location that is closest to a destination or to a known way-point toward a destination.

To formulate the problem, suppose that in the present frame we have objects $O_{i,n}$ for $i \in \{1,2,\ldots,N_n\}$. Note that all coordinates in this section are with respect to the drone center. Then the problem of finding locations that can fit a safety bubble without hitting any objects can be written as finding a location $\hat{V}_{n+1}$ for the drone to move to so that

$$D\left(\hat{V}_{n+1},O_{i,n}\right) > s^2, \quad \forall i \in \{1,2,\ldots,N_n\}, \tag{7}$$

where s is the radius of the safety bubble and $D(\cdot,\cdot)$ is an operator that computes the euclidean distance between two points in space. It is important to note that $D(\cdot,\cdot)$ is not the same as $d(\cdot,\cdot)$.

This problem might have no solution or it might have more than one solution. So we need to design two solutions for the two different cases. It is also important to note that we do not consider the possible displacements of objects when finding $\hat{V}_{n+1}$. However, given enough computing capacity, one can extend this algorithm to include possible displacements of objects.

Case 1: More than one candidate location to move to

In the case that there is more than one solution to the above equation, we select the one which is closest to the way-point W_n the drone is trying to reach. Figure 2(c) shows an example for a 2D OG map.

With the way-point in mind, the solution we seek is the position $\hat{V}_{n+1}$ that satisfies

$$\hat{V}_{n+1} = \arg\min_{(V_{n+1}\in S)} \{D(W_n,V_{n+1})\}$$
$$s.t.\;\; D\left(\hat{V}_{n+1},O_{i,n}\right) > s^2, \quad \forall i \in \{1,\ldots,N_n\},$$

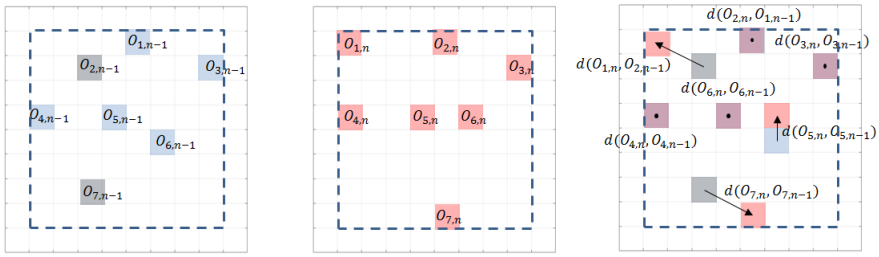
where S is the search space for $\hat{V}_{n+1}$.

Case 2: No candidate location to move to

In the case that there is no such a location, we temporarily reduce the radius of the safety bubble to the distance between the drone and the closest object i.e.

$$\bar{s} \leftarrow \min_{i\in\{1,2,\ldots,N_n\}} \left\{D\left(V_n,O_{i,n}\right)\right\} \tag{8}$$

where V_n is the current location of the drone. Then we repeat the process described in Case 1 to find the best solution. So in essence we reduce the size of the safety bubble temporarily so that we can



(a) – OG map at (n-1): improbable objects (black) and probable objects (blue)

(b) – OG map at n

(c) – OG map at n overlaid with OG map at (n-1)

Figure 1: Illustrations that show how we compute the relative displacement that corresponds to each object in the present 3D OG map, i.e. for each $O_{i,n}$ how we compute $d\left(O_{i,n}, O_{i_j,n-1}\right)$, where $d\left(O_{i,n}, O_{i_j,n-1}\right) \leq d\left(O_{i,n}, O_{k,n-1}\right)\ \forall\ k \in \{1,2,\ldots,N_{n-1}\}$.

find a location for the drone to move to. The reduction can be done in steps as well, i.e. halve the size of the safety bubble in each step.

Search Space for Candidate Location

Now the next question is what is the appropriate search space S for $\hat{V}_{n+1}$? In our implementation, we search for solutions within the safety bubble only. Figure 2(a) shows the candidate search locations for a 2D example. Another problem we might encounter is that when traveling from V_n to $\hat{V}_{n+1}$ on a straight line, the drone might come across an object. However, by limiting the search radius to s, we automatically avoid this problem.

Pseudo-Code of Sense and Avoid Algorithm

The whole sense and avoid algorithm that we have proposed with multiple stereo camera pairs is described in Algorithm 1. We used up to 3 stereo camera pairs for surrounding front, left and right views.

Experimental Results

To evaluate our algorithm, we conducted experiments using a DJI Matrice 100 drone. Specifically we gathered streams of frames from the stereo cameras already mounted on the drone and used those image pairs to experiment with. Input frame resolution was QVGA, i.e. 360 × 240, and we set census window and hamming distance window to 9 ×9 and 11× 11, respectively, to compute disparity map. Once the stereo pairs were acquired, we performed simulations on a PC.

The primary purpose of these experiments were to determine if the proposed algorithm was able to distinguish between erroneous objects and actual objects. The errors could result from time synchronization problems between stereo cameras, from sensor noise or from errors that resulted from the stereo vision algorithm. Secondly, the experiments were also used to evaluate our avoid algorithm. From the experiments that were preformed, we show some selected illustrative examples in this paper. For clarity, in the first two experiments, we only show the input from the front facing stereo cameras. Then, in the final experiment, we show an example when 3 stereo camera pairs are used as inputs.

Algorithm 1 Proposed Sense and Avoid Algorithm

Inputs:
· OG_{n-1}: OG Map from time $(n-1)$
· σ^2: variance of relative displacement
· $I_{n,(j,k)}$: i^{th} image from j^{th} stereo pair from present time
· $d_{n-1}^{max}, d_{n-2}^{max} \ldots d_{n-L-1}^{max}$: max displacement from previous $L-1$ instances
· s: radius of safety bubble

Outputs:
· OG_n: OG Map from time n
· σ^2: variance of relative displacement
· d_n^{max}: max displacement from time n

Compute 3D point cloud using $I_{n,(1,1)}, I_{n,(1,2)}, \ldots, I_{n,(N,2)}$
Compute OG_n and filter noise using connected components based thresholding (optional)

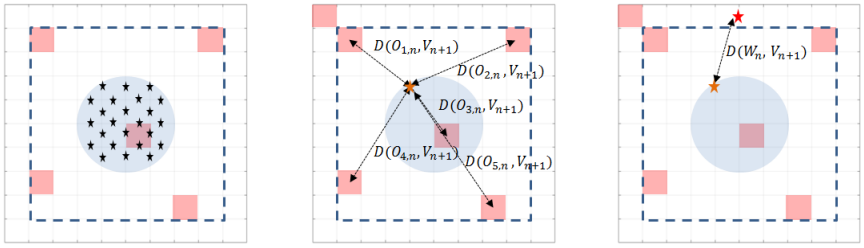
for $(i = 1, i{+}{+}, i \leq N_n)$ **do**
    For $O_{i,n}$, find corresponding $O_{i_j,n-1}$ such that $d\left(O_{i,n}, O_{i_j,n-1}\right) \leq d\left(O_{i,n}, O_{k,n-1}\right)\ \forall\ k \in \{1,2,\ldots,N_{n-1}\}$.
    if $p\left(d \geq |d\left(O_{i,n}, O_{i_j,n-1}\right)|\right) \geq T$ **then**
        $O_{i,n}$ is probable
    else
        $O_{i,n}$ is improbable
Create $\tilde{O}_{i,n}$: OG Map (with probable objects only)

if Probable object in Safety Bubble (SB) **then**

$$\hat{V}_{n+1} = \arg \min_{(V_{n+1} \in S)} \{D(W_n, V_{n+1})\}$$
$$s.t.\ \ D\left(\hat{V}_{n+1}, O_{i,n}\right) > s^2, \quad \forall i \in \{1,\ldots,N_n.\}$$
$$and\ r \in S\ \ if\ \ D(V_n, r) \leq s^2$$

    if Solution exists **then**
        Move drone towards $\hat{V}_{n+1}$
    else
        $\tilde{s} \leftarrow \min_{i \in \{1,2,\ldots,N_n\}} \left\{D\left(V_n, O_{i,n}\right)\right\}$.
        Repeat **Step** 11 with $\tilde{s}$ instead of s
    Move towards $\hat{V}_{n+1}$
else
    Continue on previous trajectory

Compute $d_n^{max} = \max_{i \in \{1,2,\ldots,N_n\}} \left\{d\left(O_{i,t-n}, O_{i_j,t-n-1}\right)\right\}$.
Compute $\sigma = \frac{1}{L} \sum_{m=1}^{L} max\left\{d_{n-m}^{max}, D_{grid}\right\}$.



(a) – Candidate locations are shown as stars

(b) – Distances to objects in the ROI from one candidate location (orange)

(c) – Distance to desired location (red) from candidate location (orange)

Figure 2: 2D example of the candidate search locations, the distance from a candidate location to all objects in the ROI and the distance from a candidate location to the desired location. Notice that for this illustration we use $D(\cdot,\cdot)$ instead of $d(\cdot,\cdot)$ since we are now computing the absolute distances.

Example 1

In Figure 3, we show three consecutive frames as the drone approaches a tree. In frame 340 we can see that there is a significant amount of noise when compared to the other frames. As can be seen in the bottom figure in frame 340, just by thresholding the 3D point cloud (by 20) and using the connected components based noise filtering we remove a significant chunk of the noise. Then, the temporal object tracking part of the algorithm is able to tag the remaining noise as improbable objects (black). In this experiment, we set the Threshold T in Equation 4 to 0.01 and D_{grid}, the grid spacing in the ROI, to 50cm.

Example 2

In Figure 4, we show another three consecutive frames for the case when a box is thrown in to the safety bubble. We can see that in frame 1011 there is some noise, but this time inside the safety bubble. The algorithm is able to recognize which objects are improbable ones. Furthermore, in this experiment, since there are objects in the safety bubble we see that a location is prescribed for the drone to move to in the direction of the way-point, which we set for experiments at 10 meter away from the current drone's location. It is important to state that the drone is not controlled according to detected objects. We merely show instantaneous decisions at each frame. Also, we do not show the location to move to if an object is not found in the safety bubble since we do not want to alter the direction of the drone in that case.

Example 3

We show in Figure 5 an illustrative example of when 3 stereo camera pairs were used. We show one frame in which there is an object inside the safety bubble in one of the three views.

Conclusions and Future Work

In this work, we presented an efficient sense and avoid algorithm for a drone. From the results we have shown that our algorithm is able to detect and remove noise quite well and then prescribe a new location to move to in order to avoid objects that may threaten the drone. As future work, we plan to integrate point cloud uncertainty as well as spacial information when computing the probability of an object and also the probable movements of objects when finding the new trajectory for the drone.

References

[1] Richard I Hartley and Peter Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[2] Helen Oleynikova, Dominik Honegger, and Marc Pollefeys. Reactive avoidance using embedded stereo vision for mav flight. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 50–56. IEEE, 2015.

[3] Alberto Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation*, 3(3):249–265, 1987.

[4] Alberto Elfes. Occupancy grids: A probabilistic framework for robot perception and navigation. 1989.

[5] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4557–4564. IEEE, 2012.

[6] Hernán Badino, Uwe Franke, and Rudolf Mester. Free space computation using stochastic occupancy grids and dynamic programming. In *Workshop on Dynamical Vision, ICCV, Rio de Janeiro, Brazil*, volume 20, 2007.

[7] Maia Rosiery Souza, Anderson A. S. and Luiz M. G. Gonalves. 3d probabilistic occupancy grid to robotic mapping with stereo vision. *INTECH Science, Technology and Medicine open access publisher*.

[8] Stefan Hrabar. 3d path planning and stereo-based obstacle avoidance for rotorcraft uavs. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 807–814. IEEE, 2008.

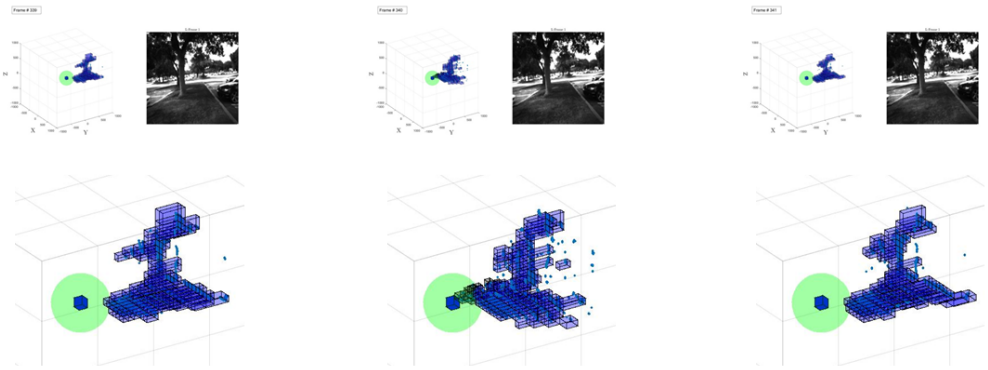
[9] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[10] Yuval Roth-Tabak and Ramesh Jain. Building an environment model using depth information. *Computer*, 22(6):85–90, 1989.

[11] Ramin Zabih and John Woodfill. A non-parametric approach to visual correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996.

Author Biography

G.M. Dilshan P. Godaliyadda is a PhD candidate from the Electrical and Computer Engineering (ECE) Department at Purdue University. Originally from Sri Lanka, Dilshan completed his Bachelor's degree in Electrical Engineering from the University of Maryland with Summa Cum Laude honors and his Master's degree from Purdue University. During the time this work was done he was employed as an Intern at the Perception and Analytics Lab at Texas Instruments.

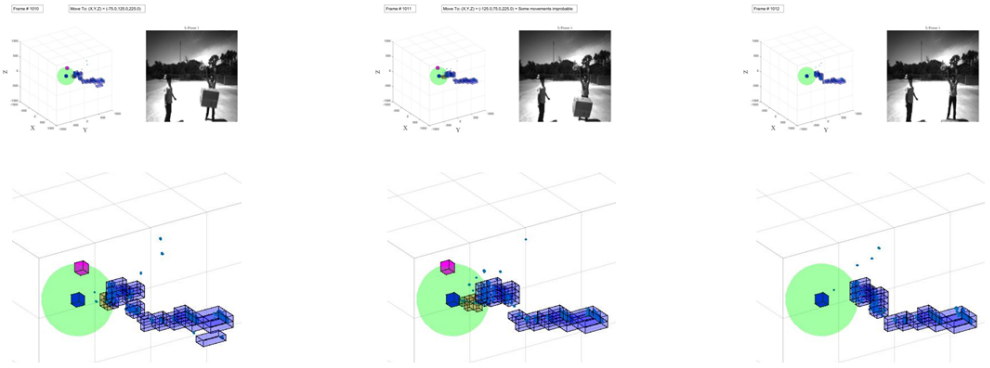


(a) – Frame 339

(b) – Frame 340: has improbable objects marked as black cubes

(c) – Frame 341

Figure 3: 3 consecutive 3D OG maps constructed using our algorithm to illustrate how well it removes noise.



(a) – Frame 1010: objects in safety bubble. Pink cube shows position to move to.

(b) – Frame 1011: has improbable objects marked as black cubes and probable objects in safety bubble (red). Pink cube shows position to move to.

(c) – Frame 1012

Figure 4: 3 consecutive 3D OG maps constructed using our algorithm to illustrate how well it removes noise and also to illustrate how our avoid algorithm works when an object is found in the safety bubble.

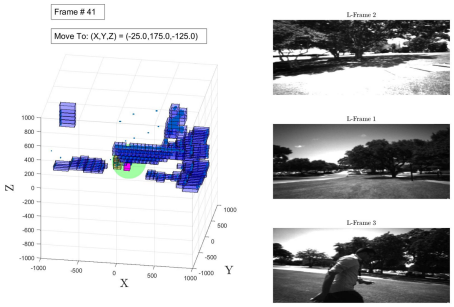


Figure 5: An example when 3 pairs of stereo camera inputs are used for sense and avoid.

Do-Kyoung Kwon received the Ph.D degree from the University of Southern California in Electrical Engineering in 2006. Since 2007, he has been worked in various fields at Texas Instruments, such as video compressing algorithms development on TI embedded processors, HEVC video standardization, medical image denosing, automotive vision, etc.

Darnell Moore is a Senior Member of Technical Staff and Manager of Texas Instruments Perception and Analytics Laboratory (PAL). PAL develops autonomous capabilities for vehicles, drones, and robots using advanced sensor fusion and environmental modeling. Darnell has led the development of stereo camera technology for self-driving cars, unsupervised robots, pilotless drones, and surveillance cameras. He was the software architect for the worlds first integrated smart-camera chip used by surveillance cameras to automatically count people and cars. Darnell joined TI after completing a Masters and Doctorate from Georgia Tech and a Bachelors from Northwestern University, all in electrical engineering. He is one of the co-chairs of Electronic Imagings Autonomous Vehicles and Machines.

Victor Cheng graduated from Texas A& M in 2000 with a MS in Electrical Engineering. Since then, he has been working at Texas Instruments as a software engineer. Throughout his career, he has implemented various imaging and vision algorithms on TI embedded processors for consumers and for automotive industries.