Stereo and LIDAR Fusion based Detection of Humans and Other Obstacles in Farming Scenarios

Stefan-Daniel Suvei, Frederik Haarslev, Leon Bodenhagen and Norbert Krüger
SDU Robotics, University of Southern Denmark, Campusvej 55, 5230 Odense M, Denmark,

Keywords: Agricultural Robot, Sensor Fusion, Neural Network.

Abstract: In this paper we propose a fusion method which uses the depth information acquired from a LIDAR sensor to guide a block matching stereo algorithm. The resulting fused point clouds are then used for obstacle detection, either by processing the raw data and clustering the protruding objects in the scene, or by applying a Convolutional Neural Network on the 3D points and labeling them into classes. The performance of the proposed method is evaluated by carrying out a series of experiments on different data sets obtained from the SAFE robotic platform. The results show that the fusion algorithm significantly improves the F1 detection score of the trained networks.

1 INTRODUCTION

In recent years, a variety of agricultural robotic solutions have been proposed with the purpose of improving farm productivity. The prototypes are generally targeted towards applications such as fertilizing and/or application of pesticides (Sharma and Borse, 2016), fruit picking (Song et al., 2016) or autonomous weeding (Nakamura et al., 2016), thus allowing the farmers to reduce the environmental impact and increase the efficiency and precision of operations.

Farming requires a large variety of processes to be carried out in order to obtain the required result. In many of these operations an important element is accurately traversing the field or the crop rows. As such, the field robot has to not only drive and reach the correct area of the field, but do so while avoiding the humans or animals that could be present and also any obstacle that might damage the system, such as trees or fences.

Hence guidance of the agricultural robot is an important task. A common way of doing this is to guide the vehicle along a pre-defined path based on input from the global positioning system (GPS). Alternatively the vehicles can be operated relative to the crop lines, using machine vision (English et al., 2014). While these options address some of the problems outlined above, they are not necessarily the best solutions in terms of precision and safety of the platform. Another option is to fully automate the robot by using various sensors in parallel and thus improving the system's representation of the surrounding environment, which can then be used for navigation, detection of humans and obstacle avoidance.

Figure 1: The SAFE robotic platform, with the Sensor Kit mounted on the blade of the tractor for a better visibility.

This paper proposes a vision based 3D scene reconstruction method to generate dense point clouds, which are processed and used to detect the protruding objects that are detected in front of the robot. The advantage of our method is that it fuses the data cues from both the LIDAR and the stereo camera on a low level, thus profiting from the high reliability of the LIDAR and the high density of the stereo data and leading up to denser, more accurate point clouds. Additionally, the resulting point clouds are used to detect and label obstacles in the scene, by using the PointNet model.

Suvei, S-D., Haarslev, F., Bodenhagen, L. and Krüger, N.
Stereo and LIDAR Fusion based Detection of Humans and Other Obstacles in Farming Scenarios.
DOI: 10.5220/0006619801660173
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 4: VISAPP*, pages 166-173
ISBN: 978-989-758-290-5

The main contribution of this paper is generating dense and accurate fused point clouds, using a sped up version of a LIDAR-Stereo fusion algorithm, which are then used by a PointNet based neural network 3D recognition system, in order to label obstacles in outdoor scenarios. The results show that by using the fused point clouds, better detection and labeling accuracy is achieved.

The remainder of the paper is organized as follows: In Section 2, we describe related methods of fusing different vision sensors for scene reconstruction and neural network training. Section 3 describes the experimental platform used for our tests. The fusion algorithm is described in Sections 4, while Section 5 contains the Obstacle Detection methodology. Section 6 shows the results of the methods, while the concluding remarks are presented in Section 7.

2 RELATED WORK

In order to allow an agricultural robot to autonomously perform tasks in an outdoor environment, an accurate 3D scene reconstruction and interpretation must first be ensured. As discussed above, 3D scene reconstruction can be achieved by fusing the data streams of multiple vision sensors. This is a topic that has been investigated before in literature and two main fusion techniques can be identified: *a posteriori* and *a priori* fusion.

In the *a posteriori* fusion method, the final result of the vision system is improved by combining the disparity map of the passive sensor (e.g. a stereo camera) with the data from an active sensor (e.g. LIDAR, 2D scanner, etc.). The fusion can be done either directly at the object level (Zhang et al., 2014) or by constructing a cost map (Romero et al., 2016), where the two types of sensors have different weights according to the level of trust in each specific region. Another option is to build an elevation model by evaluating the consensus of the stereo and laser signals, as shown in (Aeschimann and Borges, 2015).

In the case of the *a priori* technique, the stereo matching process is directly guided by the active sensor's depth information. In (Badino et al., 2011), LIDAR depth data is used to improve the stereo computation process by limiting the disparity search space of the stereo matching algorithm. Similarly, in (Somanath et al., 2013) the Kinect depth information is used to determine the data and smoothness costs of the energy minimization function used for the global stereo matching algorithm.

Accurate obstacle detection is often the main goal of doing scene reconstruction and machine learning. For 2D data this can be done using a Convolutional Neural Network (CNN) by predicting bounding boxes (Redmon and Farhadi, 2016) or by semantically segmenting the image to produce pixel labels (Long et al., 2014; Teichmann et al., 2016). In (Qi et al., 2016) they extend this idea to 3D. Their PointNet model is able to semantically segment unordered point clouds to produce point-wise class labels.

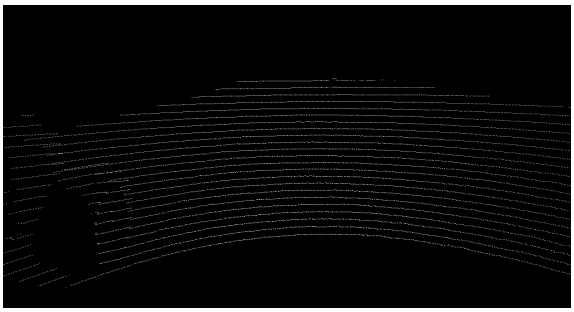
In this paper, we apply an *a priori* method which improves the stereo matching process by limiting the disparity search range around the depth value obtained from a LIDAR sensor. This method is a refinement of the algorithm in (Suvei et al., 2016). In contrast to (Somanath et al., 2013) and (Badino et al., 2011), our method focuses on the Block Matching local stereo matching algorithm, due to the advantage of scaling well for large images and for requiring less memory and computation time. In this way the LIDAR point cloud is used to guide the Block Matching process, leading to a better matching quality and a denser disparity map and point cloud. The resulting point cloud is then segmented using PointNet, and the output is used to detect and label obstacles (e.g. humans, trees) in the scenes. Compared to (Qi et al., 2016), our method uses a smaller feature vector and is applied on outdoor data.

3 EXPERIMENTAL SETUP

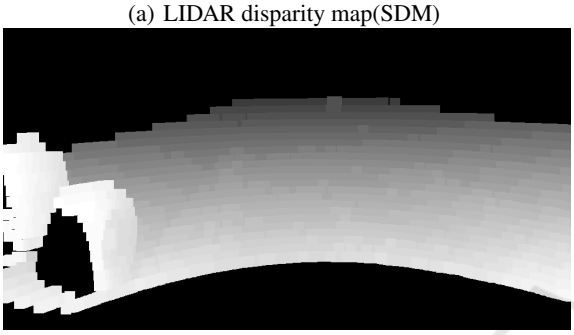
The data recordings have been done using the SAFE Platform and the Sensor Kit (Kragh et al., 2016) as shown in Figure 1. This is a multi-sensor platform, consisting of a Multisense S21 stereo camera, a Velodyne HDL-32E LiDAR sensor, a Flir A65 thermal camera, an RGB camera and a radar. The purpose of the Sensor Kit is to be used in outdoor environments on autonomous tractors to ensure safety of the humans in the field and that of the system itself. For our specific algorithm, only the stereo camera and LIDAR inputs are used. The LIDAR sensor rotates at a frequency of 10 Hz and it can record 700.000 points/second, using 32 horizontal scan beams and a horizontal field of view of 360°. The stereo system uses two S7 stereo ranging sensors, which operate at 15 Hz and output images with a 1024x544 pixels resolution.

4 SENSOR DATA FUSION

The sensor fusion algorithm used in this paper is our Guided Block Matching (GuBM), presented in (Suvei et al., 2016). The algorithm has been updated to run



(a) LIDAR disparity map(SDM)



(b) Dilated LIDAR disparity map

Figure 2: Visualization of the LIDAR disparity map - with and without dilation.

on ROS (Quigley et al., 2009) and improved by using the OpenMP (Dagum and Menon, 1998) library to parallelize some of the processes, which leads to a 76.9% increase in run time. With the purpose of offering more context to the overall work, in the following paragraphs we will restate details regarding the GuBM method.

As mentioned, due to its low computation time and good resolution scaling, one of the most popular local stereo algorithms is Block Matching (BM), where the depth information is computed by determining the pixel-distances of similar pixel-regions in the stereo images pair. In essence, the disparity of a pixel is computed by defining a reference block of neighboring pixels in the left image and then searching for the most similar block in the right image, which will reference the corresponding matching pixel. Because the images are rectified beforehand, pixel-features in the left image will be in the same pixel rows than in the right image. This restricts the search to only horizontal lines and transforms the correspondence problem into a 1D search problem which guarantees to find the solution (i.e. best-matching block). However, because the similarity check is done for all the pixels, the computation load can be high and it increases with the image resolution.

Fusion Algorithm: Choosing the disparity search interval D is crucial, because it directly influences the computation of the stereo matching process - a small interval will make the algorithm skip some matching pixels, while a too large interval will increase the computation time and increase the risk of having wrong matches. Our method proposes the use of the depth information from the LIDAR sensor to limit the disparity search interval around an expected value for each pixel. An important step in doing this is to efficiently transform the depth data from the LIDAR into a dense disparity range. Using the external calibration of the two sensors, each point $\mathbf{p} \in \mathbb{R}^3$ from the active sensor's point cloud is mapped to the left camera's coordinate system. The depth ranges are transformed into disparities by using the corresponding image displacements available from the stereo camera's calibration information file as follows:

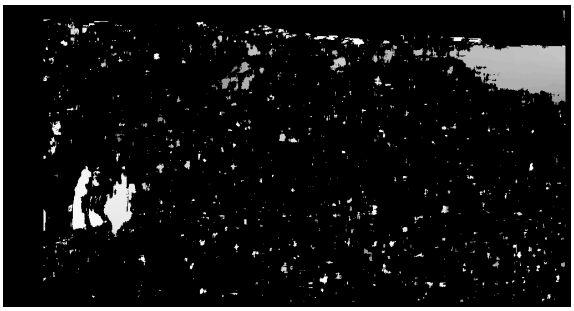
$$d_L = Bf/Z \tag{1}$$

where d_L is a LIDAR obtained disparity value, B is the baseline, f is the focal length and Z is the depth value. The resulting Sensor Disparity Map (SDM) can be observed in Figure 2(a).

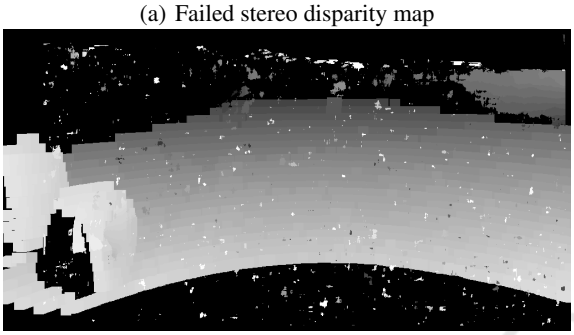
Because the depth measurements of the LIDAR sensor are of low resolution, the obtained disparity map can be sparse. To compensate for this, a morphological dilation operator is applied. A rectangular structuring element of size 3x3 pixels is used, with a number of 11 iterations. These values were chosen such that the dilated regions around the original points would barely overlap (see Figure 2(b)).

Using the dilated SDM and the d_L values, we can now perform so called "Guided Block Matching" (GuBM). First, D becomes $D = [d_L - R, d_L + R]$, where R is a constant fixed at the beginning of the algorithm with the purpose of limiting the disparity range around d_L. In the areas where there are no available d_L values, the full disparity interval D is used. Missing data caused by occlusions, textureless areas or even light conditions will still generate empty regions in the Stereo Disparity Map (StDM). These situations can be accounted for by applying Guided Block Matching with gap Filling (GuBM-F). In GuBM-F, if GuBM fails to compute a disparity value for a pixel, then the final stereo disparity value d_S is assigned from the computed dilated LIDAR disparity map, if the depth information is available for that specific pixel. A failed StDM and the resulting GuBM-F dense disparity map is shown in Figure 3(a).

The evaluation of the algorithm shows that, aside from leading to a much denser disparity map and point cloud, a secondary advantage of the GuBM-F method is that the LIDAR sensor acts as a double-checker for the stereo camera. This means that when stereo completely fails to do matching (e.g. due to

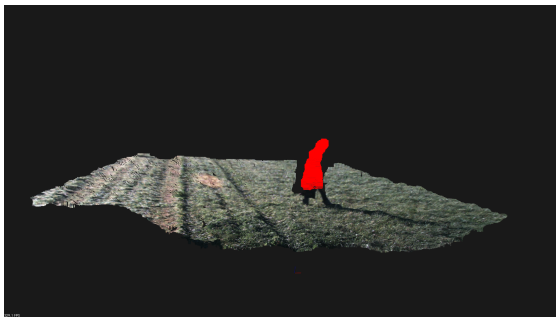


(a) Failed stereo disparity map

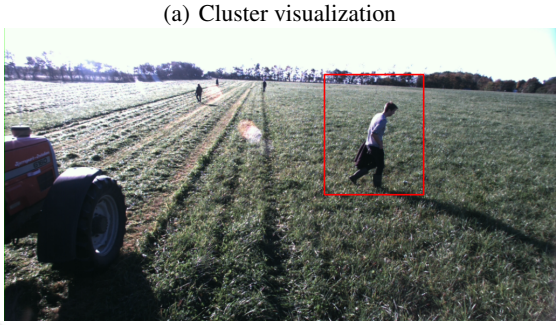


(b) GuBM-F disparity map

Figure 3: Result of GuBM-F, when stereo matching fails due to movement of the camera.



(a) Cluster visualization



(b) 2D label of cluster

Figure 4: Result of 2D label generation process.

light conditions or movement of the platform), the LIDAR data can still be used to generate a point cloud of the scene (see Figure 3(b)) and as a result obstacle detection and labeling can still be carried out.

5 OBSTACLE DETECTION

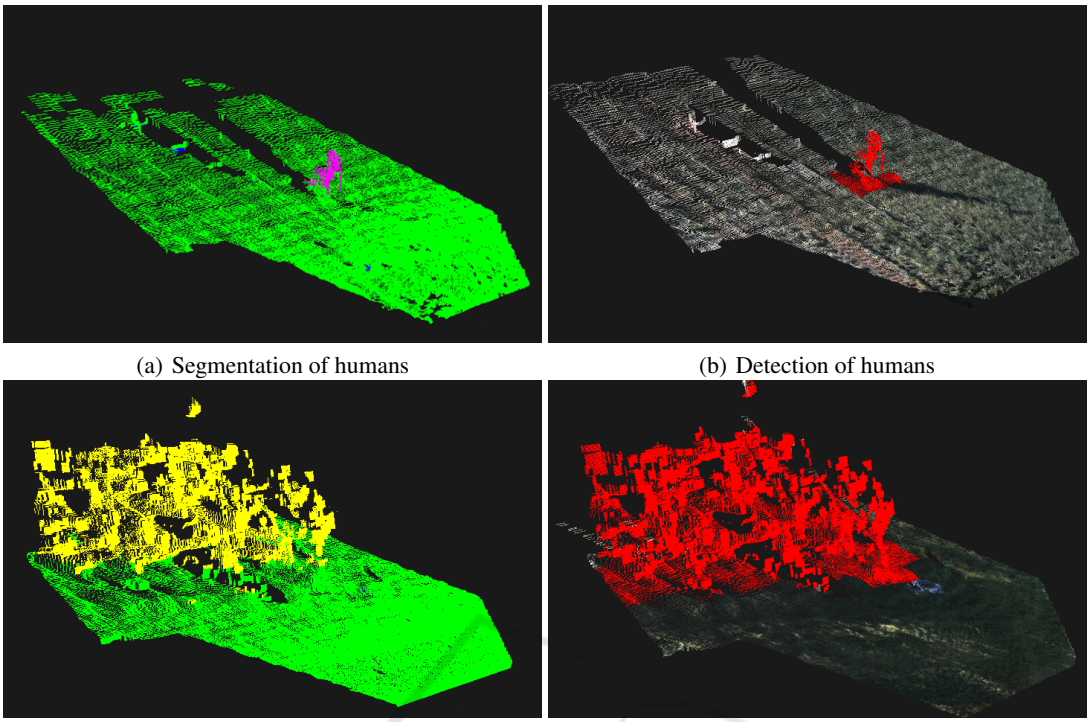
3D scene reconstruction is a vital component in agricultural robotics, because it enables the robot to take the appropriate actions accordingly to what is happening around it. To that extent, the purpose of obtaining a dense point cloud via the GuBM-F method is to enhance the result of the obstacle detection process. The result of the obstacle detection methods can be further used by the decision layer of the robot, which could issue a warning or potentially compute the best next action.

In this paper we tackle the problem of obstacle detection in two ways: via Data Clustering and by using a CNN. The following subsections describe these two methods in greater detail.

5.1 Data Clustering

The data clustering is done by processing the GuBM-F resulted point cloud using different Point Cloud Library (PCL) (Rusu and Cousins, 2011) methods, with the intent of detecting any protruding object in the scene (humans, trees, etc.) and marking them on the 2D image by attaching a bounding box around the obstacle. The first step in generating the boxes is removing the tractor points from the disparity map because the detection of the tractor is not important for the end result. After generating the point cloud, Plane Model Segmentation is applied with a plane fitting tolerance of 0.35 m. The tolerance value was chosen to account for the fact that the grass can have patches which are bigger and therefore it will never be a perfect plane. This will remove all the ground points that are within the specified tolerance. The remaining points represent the elements in the scene that are protruding from the ground. Using the PCL Euclidean Cluster Extraction algorithm, they are clustered into obstacles, with a tolerance of 0.3 m. The minimum accepted number of points for a cluster is 600, while the maximum is 12.000. The clusters are then reprojected into 2D onto the left image plane, where bounding boxes are attached around them (see Figure 4(b)). The clusters can also be visualized in the original point cloud, as shown in Figure 4(a). The method is not aware which type of obstacle is there, only that an obstacle is present.

The advantage of this method is that it can be used to improve the computation time of 2D based CNN labeling methods, such as YOLO (Redmon and Farhadi, 2016), by applying them directly onto the bounded regions, instead of on the entire image. The



(a) Segmentation of humans (b) Detection of humans

(c) Segmentation of trees (d) Detection of trees

Figure 5: The segmentation and the detection of obstacles. Here green points corresponds to grass, purple points to human, yellow points to tree and blue points to clutter.

disadvantage is that it is dependent on the system being able to produce a good-quality point cloud.

5.2 PointNet

As an alternative to the 2D labeling method, the point clouds are also processed with the PointNet (Qi et al., 2016) neural network. PointNet can semantically segment unordered point clouds by passing them through the network, producing a class label for each point.

Training: The network is trained using a dataset created with the Sensor Kit. 200 scene clouds have been annotated into the classes *grass*, *human*, *tree* and *clutter*. The clutter class is used for the tractor if it has not been completely removed, as well as artifacts from matching errors. The scene is cropped to 24×12 meters and Plane Model Segmentation is then used to remove as many grass points as possible. The left-over points are clustered using the before mentioned data clustering and each cluster is assigned one of the classes.

The input size of the network is set to n points. To accommodate this each scene is split into $b \times b$ blocks with a stride of s – resulting in overlap between the blocks if $b > s$. Blocks containing more than n points are subsampled at random, while blocks containing less have random points duplicated. If b is large there will be a lot more than n points in a block, resulting in a significant amount of information being thrown away. Therefore, when b is large each block is resampled c times, creating c different blocks per position. If there are less than 100 points in a block it is discarded. The data sets have an over representation of *grass* points in the scenes and are thereby not balanced. Therefore, blocks containing only *grass* are discarded with a probability of 80%.

The authors of PointNet use a 9-dimensional feature vector per point: *xyz* values of the point relative to the block center, *rgb* values from 0-1, as well as *xyz* relative to the whole scene, where $\{0,0,0\}$ is down in the bottom left corner, and $\{1,1,1\}$ is up in the far right corner. This works well for indoor scenes as this allows the network to learn that e.g. blackboards usually are located on the edge of rooms. This is irrelevant in our domain as all classes can be located throughout the whole scene. Therefore the feature vector is shortened to only consist of the *xyz* values relative to the block and the *rgb* values.

All networks were trained using adam optimizer with a batch size of 24 and momentum of 0.9, and an initial learning rate of 0.001 which decays every 300.000 steps by a rate of 0.5.

Detection: When a scene is segmented it is used for obstacle detection. This is done by first extract-

Table 1: Results of the data clustering method, using Block Matching and GuBM-F generated point clouds. The Processed Frames contains the total number of stereo pairs that where processed. The Human frames column shows how many of the total processed frames contain a human in the scene. The Labeled Frames column shows for how many of the scenes data clustering could be applied. The Detected humans column contains the total number of correctly labeled human frames, while the Accuracy column gives the percentage of correctly detected humans.

BM	Processed frames	Human frames	Labeled frames	Detected humans	**Accuracy**
Set 1	82	25	23	22	88%
Set 2	48	23	19	18	78%
Set 3	75	11	9	5	45%
GuBM-F	Processed frames	Human frames	Labeled frames	Detected humans	**Accuracy**
Set 1	82	25	28	24	96%
Set 2	48	23	20	18	78%
Set 3	75	11	29	8	72%

ing all the human and tree points into separate clouds. The separated clouds are then projected to the XZ plane into 1×1 meter grids creating an occupancy map. If a grid cell contains 800 points or more, it is marked as containing an obstacle of the given class. Figure 5 shows the results of semantic segmentation and obstacle detection in scenes containing a human or trees.

6 RESULTS

The proposed GuBM-F method has been implemented using and modifying the OpenCV (Bradski, 2000) specific Block Matching method. The stereo cameras and the LIDAR sensor have been appropriately calibrated beforehand.

To measure the time required for our method, the algorithm is tested on 15 consecutive scene frames, with a resolution of 1024x544 pixels, on an Intel Core i5-4210U, 1.7GHz machine. On average, the standard Block Matching algorithm has a runtime of 0.176 s. The introduction of the fusion component leads to an increase of the stereo matching time and as a result, the average runtime of GuBM-F is 0.292 s per frame.

To verify the overall performance and to quantify the impact of the fusion method, both the Data Clustering and the PointNet detection were tested on the point clouds generated by the GuBM-F method and the standard OpenCV Block Matching function.

6.1 Data Clustering Results

For the Data Clustering algorithm, three different data sets (i.e. rosbags) were used for testing:

Set 1: Scenes with a human traversing the field, in front of the robot.

Set 2: Scenes with two humans walking in the field.

Set 3: Scenes with 5 humans walking in the field.

Since the scenes contain no other obstacles than the mentioned humans, the results for detection on humans is identical with obstacle detection.

On average, the Data Clustering algorithm has a runtime of 0.097 s. The results of the tests are shown in Table 1. As mentioned before, there are cases in which the stereo matching process completely fails to generate a point cloud, due to the movement of the robot or the direct sun light. This, in return, heavily affects the accuracy of the labeling process, which relies on detecting clusters in the scene. This problem is particularly obvious when looking at the results of Set 3, where the GuBM-F method labeled 20 more frames, with an increase in accuracy of 27%. Similarly, in Set 1, we see a difference of 5 labeled frames between the two methods. On the other hand, when the Block Matching process does not fail, the result between the two methods is comparable, as seen for Set 2. It is worth mentioning that the reason why there is a difference between the number of processed frames and that of the labeled frames is because some of the frames contain only grass scenes.

A practical issue that could be addressed in order to improve the results is the usage of the thermal information from the Sensor Kit's thermal camera. This would allow for the detection of humans, even when they would lie below the plane segmentation threshold used by the data clustering method, thus improving the overall detection accuracy of the system.

6.2 PointNet Results

To further improve performance and to associate more meaningful labels to the clusters extracted, we apply the 3D deep neural network approach described in section 5.2.

Table 2: Ground truth parameters and test results of all trained networks. n is the number of points per block, b is the block size, s is the stride when seperating the scene into blocks, and c is the sample count per block position.

	n	b [m]	s [m]	c	$F1_{human}$	$F1_{tree}$	Time [s]
GuBM-F	4096	1	0.5	1	**0.931**	**0.950**	3.67
BM	4096	1	0.5	1	0.839	0.702	3.06
GuBM-F (NF)	4096	1	0.5	1	**0.959**	**0.986**	3.75
BM (NF)	4096	1	0.5	1	0.895	0.879	3.80
GuBM-F (B6)	4096	6	3	10	**0.902**	0.900	0.139
GuBM-F (B12)	4096	12	3	20	0.843	0.917	**0.033**
GuBM-F (8kB12)	8192	12	3	10	0.863	**0.919**	0.062

PointNet is used to test the performance of obstacle detection in clouds created with the proposed GuBM-F method vs. the regular Block Matching generated point clouds. To do this, a ground truth has been generated from the 200 scenes for each of the two methods. The scenes contain five different people in various poses, as well as a mannequin and some trees. 40 of the 200 scenes are selected as the validation set, by finding all 28 scenes which contain person A and 12 random scenes containing trees. The same scenes are selected for the GuBM-F and Block Matching (BM) sets. Two networks were then trained using the remaining 160 scenes, one on the GuBM-F data and one on BM data. The parameters for the ground truth creation can be seen Table 2. As PointNet requires a GPU to run in real time, the tests in this section was executed on an Intel Core i7-6700k, 4.0GHz workstation with a GeForce GTX 1080 TI.

The trained networks are used to detect humans and trees in the validation set, as per section 5.2. Table 2 shows that the GuBM-F network scores significantly higher F1 scores for both human and tree detection than the BM network which again documents the advantage of using fused information where the more reliable sensor (LIDAR) guides the less reliable but high resolution sensor.

A reason for the big difference might be that the BM some times completely fails, making it impossible to detect any objects in the scene. In these cases, the GuBM-F network is still able to detect obstacles, as the GuBM-F algorithm fills the disparity map. This is tested by making new data sets with all the scenes where the BM fails removed. This leaves 150 scenes in the training set and 32 scenes in the validation set. Two networks are then trained on the resulting sets – seen as GuBM-F (NF) and BM (NF) in Table 2. F1 scores for both networks increases, while the BM network has a significant increase in the F1 score for tree detection. Again, the detection results on the fused data where the LIDAR information guides the stereo are significantly better than for the pure stereo data.

It can be seen in Table 2 that it takes over 3 seconds for the networks to segment a scene, which is not suitable for a real time obstacle detection system. This can be remedied by increasing the size of the blocks fed to the network. Since a scene is 24×12 meters it can be separated into 288 1x1 meter blocks, requiring the network to run 288 times in order to segment the whole scene. If instead it is separated into 6×6 meter blocks it only has to run eight times, and only twice for 12×12 meter blocks.

Three new networks are trained with an increased block size – GuBM-F (B6), (B12) and (8kB12) in Table 2. c is adjusted to retain approximately the same amount of training points for each network. All networks show a significant improvement in processing speed, while still retaining good F1 scores – although a little worse than the slower counterparts. All three networks are able to detect the trees reliably, while the B12 networks struggles with humans to some degree. This makes sense as small objects are represented with a minimal amount of points at that block size. This introduces a balance between how fast the segmentation should be, versus how small objects it should be able to detect. By doubling the number of point per block (8kB12) we increase the F1 score slightly versus (B12), but the segmentation time is also doubled as the network has to process twice as many points.

Using the GuBM-F (B12) network, the whole pibeline – including block matching and point cloud segmentation – takes at average $0.245s$. However, the block matching and segmentation can be run in parallel, as one is computed on the CPU while the other is on the GPU. In practice this means that the system runs at 10Hz with a $0.245s$ delay.

7 CONCLUSIONS

In this paper, we have presented an *a priori* fusion method which improves a local stereo matching al-

gorithm by using the depth information provided by a LIDAR sensor to guide the stereo correspondence process by limiting the disparity search interval. The advantage of the method is that it generates fused and dense point clouds which can then be used for a more accurate detection of protruding obstacles in the scene. The principle of fusion is that the more reliable LIDAR sensor which however has a low resolution guides the less reliable but higher resolution sensor. The obstacle detection is performed in two ways on that data: by data clustering and by using a CNN.

The results presented in Subsection 6.1 show that the fused point clouds lead to a better 2D clustering result, where the accuracy in detection can be up to 27% better than in the case where the Block Matching point clouds are used. The simple clustering is not able to detect which type of obstacle is there, only that it is present.

The CNN is able to detect different classes. Our results show that it is able to accurately detect humans and trees in the fused point clouds while distinguishing between them. The guidance through the LIDAR information significantly improves the detection F1 score of the trained networks.

ACKNOWLEDGEMENTS

This work was supported by the project SAFE Perception, funded by the Danish Innovation Fund.

REFERENCES

Aeschimann, R. and Borges, P. V. K. (2015). Ground or obstacles? detecting clear paths in vehicle navigation. In *2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3927–3934.

Badino, H., Huber, D., and Kanade, T. (2011). Integrating lidar into stereo for fast and improved disparity computation. In *3D Imaging, Modeling, Processing, Visualization and Transmission, Int. Conf. on*, pages 405–412.

Bradski, G. (2000). The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123.

Dagum, L. and Menon, R. (1998). Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55.

English, A., Ross, P., Ball, D., and Corke, P. (2014). Vision based guidance for robot navigation in agriculture. In *2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1693–1698.

Kragh, M. F., Christiansen, P., Laursen, M. S., Larsen, M., Steen, K. A., Green, O., Karstoft, H., and Jørgensen, R. N. (2016). Fieldsafe: Dataset for obstacle detection in agriculture. *CoRR*.

Long, J., Shelhamer, E., and Darrell, T. (2014). Fully convolutional networks for semantic segmentation. *CoRR*.

Nakamura, K., Kimura, M., Anazawa, T., Takahashi, T., and Naruse, K. (2016). Investigation of weeding ability and plant damage for rice field weeding robots. In *2016 IEEE/SICE International Symposium on System Integration (SII)*, pages 899–905.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, page 5. Kobe.

Redmon, J. and Farhadi, A. (2016). Yolo9000: Better, faster, stronger. *CoRR*.

Romero, A. R., Borges, P. V. K., Elfes, A., and Pfrunder, A. (2016). Environment-aware sensor fusion for obstacle detection. In *2016 IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 114–121.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point cloud library (PCL). In *Robotics and automation (ICRA), 2011 IEEE Int. Conf. on*, pages 1–4. IEEE.

Sharma, S. and Borse, R. (2016). Automatic agriculture spraying robot with smart decision making. In Corchado Rodriguez, Juan Manueland Mitra, S. and Thampi, Sabu M.and El-Alfy, E.-S., editors, *Intelligent Systems Technologies and Applications 2016*, pages 43–758. Springer International Publishing.

Somanath, G., Cohen, S., Price, B., and Kambhamettu, C. (2013). Stereo+kinect for high resolution stereo correspondences. In *3D Vision, Int. Conf. on*, pages 9–16.

Song, X., Shan, H., Liu, H., and Guo, J. (2016). An underactuated end-effector design for fruit picking. In *2016 23rd Int. Conf. on Mechatronics and Machine Vision in Practice (M2VIP)*, pages 1–5. IEEE.

Suvei, S.-D., Bodenhagen, L., Kiforenko, L., Christiansen, P., Jørgensen, R. N., Buch, A. G., and Krüger, N. (2016). Stereo and active-sensor data fusion for improved stereo block matching. In *Int. Conf. Image Analysis and Recognition*, pages 451–461. Springer.

Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., and Urtasun, R. (2016). Multinet: Real-time joint semantic reasoning for autonomous driving. *CoRR*.

Zhang, F., Clarke, D., and Knoll, A. (2014). Vehicle detection based on lidar and camera fusion. In *IEEE 17th Int. Conf. on Intelligent Transportation Systems*, pages 1620–1625.