Stereo and Mono Depth Estimation Fusion for an Improved and Fault Tolerant 3D Reconstruction

Mircea Paul Muresan, Marchis Raul, Sergiu Nedevschi, Radu Danescu
Technical University of Cluj-Napoca
Computer Science Department, Cluj-Napoca Romania
{Mircea.Muresan, Raul.Marchis, Sergiu.Nedevschi, Radu.Danescu} at cs.utcluj.ro

***Abstract*—Depth estimation approaches are crucial for environment perception in applications like autonomous driving or driving assistance systems. Solutions using cameras have always been preferred to other depth estimation methods, due to low sensor prices and their ability to extract rich semantic information from the scene. Monocular depth estimation algorithms using CNNs may fail to reconstruct due to unknown geometric properties of certain objects or scenes, which may not be present during the training stage. Furthermore, stereo reconstruction methods, may also fail to reconstruct some regions for various other reasons, like repetitive surfaces, untextured areas or solar flares to name a few. To mitigate the reconstruction issues that may appear, in this paper we propose two refinement approaches that eliminate regions which are not correctly reconstructed. Moreover, we propose an original architecture for combining the mono and stereo results in order to obtain improved disparity maps. The proposed solution is designed to be fault tolerant such that if an image is not correctly acquired or is corrupted, the system is still able to reconstruct the environment. The proposed approach has been tested on the KITTI dataset in order to illustrate its performance.**

Keywords— fault tolerance, monocular depth estimation, stereo reconstruction, CNN, fusion, automated driving.

I. INTRODUCTION

The future of transportation is driven by the development of autonomous vehicles and advanced driver assistance systems, in order to reduce driving failures and enable a safer and more comfortable way of transportation.

The general pipeline of an autonomous system is built of four components: the sensing field, the perception module [1], the path planning module [2] and the control module [3]. Environment perception is one of the most fundamental and challenging problems of autonomous driving. Using this module, the autonomous system has to reliably detect all the traffic participants in various weather conditions like snow, rain, fog etc. and predict their future states. In order to get a 3D representation of the surrounding environments, accurate depth measurement is an essential task. In the case of autonomous systems, for gathering 3D data, a complex network of multiple sensors is needed for the perception of the environment to fully replace the human driver, such as LiDARs [4], RADARs [5], stereo-cameras [6] or mono-cameras [7]. To accurately represent the environment, in autonomous cars, multiple sensors are fused [8] to combine the redundant and complementary information from all sensors. Even so, for ensuring high quality results, the fusion system should not be centred around a single sensor and the output coming from each sensor should be as good as possible. Not centering the whole system around a single sensor will ensure robustness against individual sensor failure. 3D LiDARs provide very accurate geometric information of the driving scene, which can be useful to support a close object detection [4] or a SLAM [9] system. However, because of the sparsity of the acquired point clouds, LiDARs can offer a poor representation of the scene topology, and can have difficulties in detecting objects at far distances. Furthermore, the costs of LiDARs are very high and they have accuracy problems in adverse weather conditions. RADARs have an advantage over other technologies due to their ability to measure objects that are not in their direct line of sight, as a result of the ability of electromagnetic waves to bounce off hard surfaces [10]. Moreover, RADAR technology can be used to measure the speed of objects and they can work in adverse weather conditions. A downside of RADARs is that they discard objects in order to avoid over-reporting. Such a feature may be useful when omitting the road surface, however it can be very risky for object detection applications because it may fail to detect static objects. In addition, RADARs have a small vertical field of view and are unable to reconstruct the whole scene in detail.

Stereo vision-based 3D reconstruction is a much cheaper method of representing the environment. Such solutions have attracted a lot of interest from the research community due to the fact that cameras can also extract semantic information from a scene. Multiple types of methods have been developed for various applications. For instance, applications which require low resource consumption and real time execution, use local stereo algorithms [4]. For improving the quality of the 3D reconstruction, semi-global methods are used [11]. These solutions usually perform a semi-global matching aggregation on the cost volume in order to produce better quality results. Semi-global approaches can run in real time if they are tailored to run on a GPU or if CPU hardware acceleration methods are used. Global stereo reconstruction approaches produce high quality depth maps, however they are not suitable for real time automotive applications because of the high power and resource consumption when they are hardware accelerated [12].

Monocular depth estimation (MDE) uses a single camera to infer a depth map of the environment [7]. The papers published in the literature in the past 50 years regarding monocular depth perception can be split into three categories: structure from motion-based methods [13], handcrafted features methods [14] and state of the art deep-learning based methods [15]. The recent progresses in neural network-based solutions have made monocular depth estimation more reliable and desirable for advanced driver assistance systems and autonomous systems applications. There are many advantages of MDE solutions with respect to the stereo camera-based approaches. First of all, the costs of acquisition are smaller since there is a need only for one camera instead of two. Secondly, there is no need for complex cross calibration and temporal alignment approaches for obtaining accurate results. Even though humans appear good at judging depth from a single image, because of their common sense, life experience and exploitation of certain features like perspective or scale of known objects, it is impossible for a

978-1-6654-0976-6/21/$31.00 ©2021 IEEE

computer vision system to learn all prior knowledge about the geometric structures of all the objects in a scene. Certain structures from the environment may be incorrectly reconstructed and some may be unreconstructed at all. This issue makes current monocular depth perception solutions less reliable for critical applications. In this paper we propose an original fusion approach between monocular and stereo depth estimation in order to obtain better quality disparity maps. Furthermore, two refinement solutions are proposed for improving the disparity maps obtained using the two reconstruction methods. The main contributions of this work are enumerated below:

- An original pipeline for combining monocular depth estimation with local stereo algorithms in order to obtain a more reliable and fault tolerant approach for depth estimation.
- An original algorithm for eliminating speckles of wrong disparities.
- An adaptation to monocular depth estimation of a refinement algorithm, previously applied in stereo.
- A fusion scheme that combines disparity values from monocular and stereo depth estimation using semantic segmentation, resulting in an improved disparity map.

The rest of the paper is structured as follows: in section II we will review the state of the art in the field of monocular depth perception. In section III the proposed solution will be presented and in section IV we discuss the results obtained with our refinement approach. Finally, we conclude the paper in section V.

II. Related Work

A. Monocular Depth Estimation

Inferring depth from single images is not a straightforward task because of the loss of the 3D information in the process of capturing images with a single camera. Early monocular depth estimation algorithms mainly relied on texture variations, occlusion boundaries, defocus, surface layout and size of known objects as indices for predicting the dense depth maps. In the pioneering work of Horn et. al. [16] the color gradients of an acquired image are exploited to estimate depth information of objects. Following the feature engineering approach Kong and Black [17] formulate depth estimation as an intrinsic image estimation problem. The intrinsic images correspond to physical properties of the scene such as shadows, surface shape or reflectance. The authors train a contour detector to predict surface boundaries from shading. The predicted contour is used for enhancing the quality of depth maps.

The author in [18] proposes a depth estimation from mono approach that uses a learning mechanism which incorporates the size of the known objects in the image. The absolute scene depth is derived from the image structure represented as a set of features from Fourier and wavelet transforms. As the recognition of objects in unconstrained conditions is prone to errors, the authors in [19] combine texture features, geometric context and motion boundary based monocular features with co-planarity spatio-temporal constraints to infer depth from monocular videos. In [20] the authors categorize the objects in four types: sky, ground, cubic and plane. Based on the type of object and using the perspective information, a relative depth value is assigned to each object model.

Compared to the feature engineering approaches the deep learning solutions have achieved much greater reconstruction accuracy in the field of dense depth estimation. In the work of Laina et. al. [21] the authors propose a fully convolutional network, having an encoder decoder architecture, for the task of depth estimation. The encoder consists of a modified ResNet-50 [22] architecture after removing the FC layers and the last pooling layer. The decoder guides the network to efficiently upscale the results via a series of convolutional layers. By having the depth of the network increase, the accuracy of the model also increases because the large receptive fields capture more context information. By using this idea CNNs with more than 100 layers have been applied for the task of MDE [23, 24]. To further improve the quality of depth maps, Mancini et al. [25] has concatenated optical flow and the current RGB image. The stacked images are fed to an encoder-decoder network to learn depth from fused information. Alhashim and Wonka build upon the state of the art of MDE and design a connected encoder-decoder architecture [26]. The decoder used consists of a bilinear up-sampling and two convolutional layers. With the deep network architecture and elaborate training strategies the designed model obtains more accurate results on international datasets. Yin et al. [27] proposes the use of a geometric constraint in the 3D space, by designing a loss term. The designed loss function fuses the pixel-wise depth supervisions with geometric information which enables the depth estimation network to generate an accurate depth map.

Another semi-supervised depth estimation framework is presented in [28]. The proposed architecture consists of two networks: the first one is a pose estimation network and the second is a symmetric depth estimation network. The model is fed the RGB and semantic segmentation image to produce two results, an initial depth map and a semantic weight map. The two maps are combined to generate the final refined depth map.

B. Stereo Vision Depth Estimation and Fault Tolerance

Feature engineered based stereo matching methods can lead to ambiguous associations among the patches from the left and right images, as wrong matches can easily have lower association cost than the correct ones because of occlusions, reflections, noise etc. To overcome this issue, many cost-aggregation approaches have been developed to achieve more accurate depth estimations [29, 30].

Even though, recently, stereo reconstruction solutions which occupy the top positions in international benchmarks use deep learning methods, such approaches need vast amounts of training data to produce high quality results [31, 32]. Moreover, a GPU is needed to run deep learning models in real time. Furthermore, when a network is presented a scenario which was not available in the dataset, the method might not reconstruct the scene accurately. Feature engineering-based approaches like the ones presented in [29] and [30] can run in real time on the CPU without hardware acceleration methods, which makes them suitable applications on low-cost single board computers.

The most common descriptors used for stereo matching are intensity based (SAD, NCC) [33], binary (census, daisy) [29], non-parametric (Rank Transform) or other custom ones [6]. An aggregation step improves the robustness against noise or surfaces which are not fronto-parallel [29]. An additional

semi-global [34] or global [35] optimization step is used for refining the results and finally a smoothing is performed using a median or bilateral filter.

One common issue when dealing with stereo methods is that a frame may not be acquired correctly or the image may be corrupted. One of the reference approaches from the literature that deals with error mitigation in the context of stereo vision is presented in [36]. The key steps for recovering the missing stereoscopic video frames are temporal change detection from two successive left images, disparity estimation and frame difference projection. This proposed disparity-based frame difference projection solution, manages to offer good results in case there is a frame drop issue. In his work, Chung et al. [37], exploits the motion vectors of each color image for concealing the right color frame. Furthermore, the authors use a 3D image warping technique to determine matching pixels between inter-views, and use the motion vectors and the intensity differences of matching pixels to reconstruct the image. Miclea et. al. [38] identifies three points of failure in the stereo reconstruction pipeline and uses a neural network to resolve the failure. The proposed network is modelled as a regression and uses the previous correctly acquired RGB image, the previous disparity image and the semantic segmentation image to output the final disparity map.

III. Proposed Solution

The proposed solution section is split in four sub-sections. In the first three sections, the contributions regarding the disparity refinement approaches are displayed. The fourth section is dedicated to the fusion of monocular and stereo disparity maps with the aim of creating an improved result. The system is built such that it can also be fault tolerant in case of individual camera failure. In Figure 1, a block diagram of the proposed solution is displayed. Once the left and right frames are captured, they are verified to see if they are valid (if the frame contains information, and is not just noise). If both frames are validated, they are used in the stereo and monocular reconstruction modules and after the refinement stages, the results are fused to obtain a higher quality depth map. In case only one frame is correctly captured, that frame is used by the MDE approach, regardless which frame it is. Finally, if no frame is captured correctly a warning message is displayed.

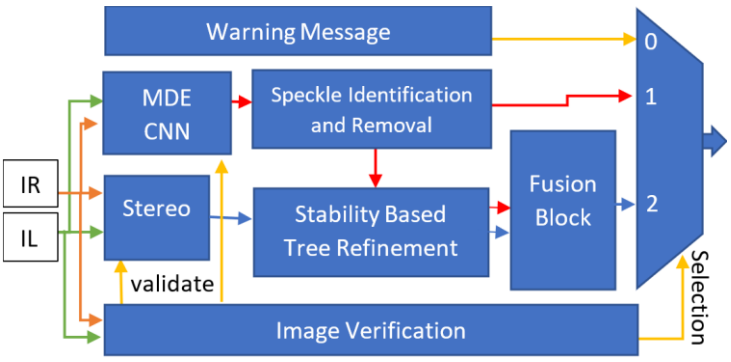

Figure 1. Pipeline of the fault tolerant approach that combines the refined stereo and mono disparity maps.

A. Monocular Depth Estimation

The monocular depth estimation approach used for generating the disparity map is based on the Convolutional Neural Network(CNN) presented in [39].

The neural network was built on an encoder decoder architecture having a total of 14 layers. The base architecture of the neural network and the distribution of layers is given by a slightly modified version of the VGG-16. At the same time, the whole architecture can be seen as an encoder-decoder type architecture, where the encoder part represents the down-sampling of the feature maps, and the decoder part represents the reverse up-sampling operation. The encoder contains 7 blocks, and in each of them the application of convolutional filters is first performed using stride 1 and the necessary padding in keep the image size because their rapid decrease would lead to a considerable loss of image information. Following this operation, we perform a strided convolution using a stride of 2 in order to halve the height and width of the image. The encoder contains 7 convolutional layers having ELU activation. The architectural structure of the encoder is the following: we first use a convolutional layer having 7x7 kernel and 32 filters, the second convolutional layer has a kernel of 5x5 and 64 filters, the third one has 128 filters and a kernel size of 3x3, the fourth has 256 filters and the convolution kernel size of 3x3, the fifth, sixth and seventh convolutional layers all use 512 filters and kernel size of 3x3.

The CNN's decoder also consists of 7 blocks, and each block consists of two parts: the operation of multiplying the dimensions of the feature maps received as input by a factor equal to 2, together with the unification of the new feature map obtained in the previous step with the corresponding block (having the same dimensions) in the encoder and applying a new convolution using stride 1 to obtain even more features or to recover information lost during dimensional reduction. Unification is performed using skip-connections with the encoders blocks. The reconstruction can be performed by a bilinear sampler using in turn the original left images together with the right disparity maps, and then the original right images together with the left disparity maps. For accurately computing the disparity map we store the original image as well as 3 other images obtained by dividing the original to 3 different scaling factors (2,4 and 8). The final disparity image is obtained by uniting the individual disparity images obtained for the four scales mentioned above with the help of a cost function. The whole cost function is composed of: the cost that compares the reconstructed images with the original images (both for the left image and for the right image), the cost that forces the disparities to be smooth locally and the cost that forces the obtained disparity maps (left and right) to be equal to each other. As for the cost function that forces the disparity maps to be locally smooth, L1 norm will be applied on the disparity gradients, and as discontinuities often occur at color changes in the image, the comparison terms applied on the disparity gradients and the image gradients.

The model has been implemented in TensorFlow and the architecture used has approximately 30 million parameters, which are modified during the learning process. The CNN was trained using the Adam optimizer, for 50 epochs applying different learning rates. The first 30 epochs have been trained using a learning rate equal to 1e-4. The learning rate was halved for the following 10 epochs, and then halved again for the last 10 epochs. The input data was augmented using the following transformations: horizontal flip, random color changes and gamma change with a random value

between 0.8 and 1.2, the color intensities in each channel (RGB) were also be changed with a random value, after which the values of all pixels were normalized. It is worth noting the fact that we did not create the architecture described above. However, we have mentioned the most relevant details of its design for the manuscript completeness and because we use it in our main processing pipeline.

B. Tree based refinement

The monocular and stereo disparity maps obtained are refined in order to obtain better quality results and eliminate regions which are not correctly reconstructed. The refinement algorithm has three steps: the creation of minimum spanning trees based on disparity values, the process of linking the trees and a final aggregation step. The tree-based refinement is an adaptation of the algorithm presented for stereo in [40], for the monocular depth estimation problem. The disparity map resulted by taking the left image as reference can be thought of as an undirected graph, in which each pixel will form a node in the graph, and possible edges exist in 4 directions (up, down, left and right). Each edge will have associated to it a weight, which is calculated according to (1) and denotes the difference in intensity of the pixels that form the edge in the original left image together with the difference in disparity of the same pixels, from the left disparity map.

$$W_{e_{pq}} = \begin{cases} \max\limits_{c \in \{r,g,b\}} | I_c(p) - I_c(q) | * \alpha + (1 - \alpha) * \Delta d, \Delta d < \tau \\ C, \qquad \Delta d \geq \tau \end{cases} \quad (1)$$

The nodes of the neighboring pixels p and q, are linked using an edge denoted by e_{pq}. The intensity value of a pixel in the original image is represented by I_c and Δd is the difference in disparity while the constants used have the following values α = 0.8, τ = 8 and C = 256.

It should be noted that only the edges which have a label less than constant C will be considered for the step of the generation of minimum spanning trees. The constant value C will be assigned to an edge when the difference in disparity of the pixels forming the edge is greater than a predetermined constant value, τ, because a higher value denotes the dissimilarity between the two pixels.

As all possible minimum spanning trees were built following the application of Kruskal's algorithm, it should be noted that some of them will be considered as support regions (stable trees) for areas with reconstruction errors. The purpose of the second major step of the algorithm (binding) is to find a support region for each erroneous (unstable) tree. The classification of trees into the two categories is achieved by taking into account both the size (number of existing nodes in the tree) and the ratio of the tree error, as presented in (2) and (3). If the number of nodes in the tree is less than a predetermined threshold, it indicates that the grouping represents a small hole. As the tree error ratio is higher than another set threshold, an area was identified which was present in only one of the two pictures when the network was run. In both cases, it will not be considered a support region, therefore it will be labelled as unstable.

$$\delta = \begin{cases} 0, \ r > \gamma_1 \ sau \ n < \gamma_2 \\ 1, \ r \leq \gamma_1 \ si \ n \geq \gamma_2 \end{cases} \quad (2)$$

$$r = \frac{number\ of\ error\ nodes}{total\ number\ of\ nodes} \quad (3)$$

In (2) n represents the number of nodes in a tree, r is the error ratio of the tree and γ_1 and γ_2 are two constants having the values γ_1 = 0.2 and γ_2 = 100. In order to calculate the error ratio, each tree must know its number of erroneous nodes. A node (pixel) will be considered erroneous when the difference between the disparity value at the position of that pixel in the left disparity map and the value at the position of the same pixel shifted to the left by the previously obtained disparity value in the right disparity map is greater than or equal to a threshold of 1, which indicates an inconsistency at that position in the two disparity maps. Determining the correctness of each node is already done when initializing the trees.

The first step in finding a support region for an unstable tree is to identify all neighboring stable trees which have an error rate lower than that of the unstable tree. Starting from the root of the unstable tree, all neighboring trees will be searched in 8 directions and if they comply with the condition aforementioned regarding the error ratio will be considered to be target regions to support the tree in question.

Having established the possible support regions for each unstable tree, the next step is to find the most suitable of them. If no stable tree was found in any of the neighboring region, the whole procedure will be reiterated after all unstable trees have been evaluated. The choice of the most favorable support region will be made according to a new term, called suitability that will take into account the difference in intensity and the difference in disparity of the last pixel in the unstable tree and the first pixel encountered of the target tree. The formulas used for computing the value of suitability can be observed in (4), (5), (6).

$$Z = \lambda_R * R + \lambda_I * I + \lambda_D * D \quad (4)$$

$$R = -r, I = -\max|I(p) - I(q)| \quad (5)$$

$$D = \frac{card|tree.D_t \cap tree.D_T|}{card|tree.D_t \cup tree.D_T|}, D_t = \cup d_p, p \in \mathrm{T_p} \quad (6)$$

The meaning of the terms in the equations above are the following: r is the error ratio of the target tree, the variable I represents the difference in intensity between pixels p and q in the left original image, by $card|M|$ we denote the total number of elements in the collection M, D_t is the collection of disparity values of the stable nodes p in the unstable tree t, D_T represents the collection of disparity values of the stable nodes p in the target tree $\mathrm{T_p}$, and finally the values of the used constant weights are $\lambda_R = 0.1$, λ_I = 0.8, λ_D = 0.1. The support tree that will be selected will be chosen according to the highest Z value. The linking between an unstable and a stable tree will be achieved using a new edge between the root node of the unstable tree and one of the nodes of the neighboring stable tree. Since the result of the linking will be a new tree, it is necessary to recalculate the total number of nodes, the error ratio of the resulting tree and possibly the new disparity interval corresponding to the tree. Following the union, it is necessary to recalculate the total number of nodes, the error ratio of the resulting tree and possibly the new disparity interval corresponding to the tree. Both the number of nodes and the error ratio will be calculated by summing these values already calculated for each of the two trees, before the binding step. In order to recalculate the range of disparity values by combining the range of disparity values corresponding to the unstable tree with that corresponding to the target tree, one of the following three conditions should be met. The first condition is that the total number of nodes of the target tree is less than a threshold equal to 100 in our case or the error ratio of the tree is greater than another

threshold empirically set to 0.6. The second condition refers to the range of disparity values calculated for the target tree has identical consecutive values, which indicates the presence of a hole. The third and final condition refers to the difference in the error ratio between the most suitable tree and the unstable tree which should be below a threshold equal to 0.05. If none of these conditions is met, the new disparity interval of the newly created tree will be identical to the unmodified one of the most suitable tree found. In order to be able to perform the last step of the algorithm, the aggregation of the resulting trees, it is necessary to create a cost volume based on the left image disparity map, as described by (7).

$$C_d(p) = \begin{cases} |d - D(p)|, p\ is\ stable\ and\ D(p) > 0 \\ 0, \qquad otherwise \end{cases} \quad (7)$$

The value d refers to one of the disparity interval values (which is maximum 128 in our case) and $D(p)$ disparity value of stable pixel p in left disparity map. The aggregation step aims to change the disparity value of each node according to all the other nodes contained in the tree in which it is located. In this step the disparity values of the stable nodes are propagated to the erroneous ones. The level of contribution of each node depends on the distance between the two nodes, which will be calculated by summing all the labels of the edges that connect the two nodes in the tree, following the shortest path between them. The greater the distance between two nodes, the less important the contribution will be. In each tree, starting from the leaves, the contributions of each node will be propagated to the root thus calculating an aggregated cost volume based on the cost volume constructed using (8). In the second step the algorithm will start from the root of the tree and the contributions will propagate to the last nodes in the tree, calculating a final cost volume based on the one built in the previous step. The equations can be seen in (8). The term $C_d(v)$ is the value from the cost volume resulted by aggregating the values in a bottom-up manner, S is a function that computes the similarity of two values, P is the parent node of node v.

$$C_d^A(v) = S(P(v), v)C_d^A(P(v)) + [1 - S^2(v, P(v))]\, C_d(v) \quad (8)$$

Therefore, by inspecting the lists of possible disparity values corresponding to each pixel, the lowest values encountered in each interval will be retained, but they will not represent the final values. The last step in the refining technique using stability-based trees is the sub-pixel interpolation, which aims to influence each minimum value of disparity previously found by the two neighbors between which it is located. We apply this interpolation in order to get more accurate results. This refinement method has been implemented on the GPU.

C. Speckle Filtering

Speckles of erroneous disparities may appear on the resulted disparity map due to repetitive patterns on some regions or untextured surfaces. For removing these speckles and replacing them with a more suitable disparity values, obtained from the region neighboring values, the speckle removal approach was created which uses two traversals of the original image in order to identify and remove the unwanted values.

In the first traversal we create clusters of similar disparities and assign a label(or ID) to each of them. The first labels will be designed for the values of disparities encountered, with respect to a four-neighbour vicinity (left, top left, top, top right). However, only the disparities that meet the condition from (9) will be considered to be part of the same cluster.

$$\left|D_{ij} - first(D_{ij}^{neighbour})\right| < T \quad (9)$$

In equation (9) the term D_{ij} refers to the disparity value of the unlabeled pixel, $first(D_{ij}^{neighbour})$ is the first disparity value found for a certain cluster which has been labeled, T is a threshold which was chosen empirically and has the value 15 and finally | x | refers to the absolute value of x. If no neighboring pixel complies with (9) a new ID will be assigned to the unlabeled pixel in question. In case a pixel has a neighbor, which was previously identified, and the pixel respects (9) the unlabeled pixel will receive the same ID as its neighbor. When there are two or more neighboring pixels that have been visited and they all respect (9), the current pixel will be labeled using the minimum cluster ID and all the clusters that neighbor that specific pixel will be considered equivalent. After creating the equivalence classes between the disparity clusters, in the next step we apply a breadth first search (BFS) algorithm to identify all the connected component clusters and reassign to each one a new label. The new label received by the new clusters resulted after the BFS operation is selected as the minimal label ID of the clusters that have been combined. After re-labelling we also have the size in pixels of each disparity cluster as a biproduct. We make a second pass through the image and nullify all disparity values belonging to clusters having an area smaller than 30x30 pixels.

For filling the zeroed regions, we are using the semantically segmented and instance level segmentation images. Going from left to right and from top to bottom in the entire disparity map, for each incorrectly reconstructed value, we search for the closest neighbouring pixel that is correctly reconstructed provided that both pixels are part of the same instance or in case the instance is not available (for example for the road, or vegetation scenarios) the search will be validated using the semantic image. The process will be repeated over the entire disparity image each time a new pixel with an incorrectly reconstructed disparity value has been identified. The final result will be filtered using a 3x3 median filter.

D. Mono and Stereo Disparity Map Fusion

The stereo and monocular fusion is useful because in case one of the cameras from the stereo rig fails to capture images or breaks for any reason, the system should still be able to reconstruct the scene, even though the quality might not be as good as when both cameras are functioning. Furthermore, the stereo vision system may fail to reconstruct certain regions due to issues like repetitive patterns, untextured areas, occlusions and so on. For such scenarios the monocular depth estimation approach might provide useful information, and help fill in the unreconstructed regions.

For creating a solution that does not consume many resources and is therefore portable on embedded devices, we have used a stereo solution based on local stereo reconstruction algorithms with a configuration similar to the one presented in [29]. In the first step, we are checking if the image received from the camera is correct. For this the histogram of the grey level image is computed in a region of

interest in the lower part of the image. If more than 70% of all the pixels from the histogram are below the threshold value 10 or above 230, the image is labelled incorrect and it is not used in the reconstruction process. In the case only one image is acquired correctly, the scene is reconstructed using the monocular depth estimation algorithm. If both images are correctly acquired the block matching approach is applied on both images, as well as the monocular depth estimation method on each individual image. It is worth noting the fact that if both images are acquired correctly the left image is taken as reference and the fusion is performed using the left disparity image from the monocular depth estimation procedure. The next step in our algorithm is verifying if in the disparity image obtained via stereo reconstruction there are any unreconstructed regions. If such regions are found they are filled with the information from the monocular depth estimation map. Following this step, we perform the fusion of the disparity maps obtained using the two reconstruction methods (monocular depth estimation and stereo), and use the semantic segmentation image of the left frame to aid the fusion process. The fusion is performed on the consideration that the monocular depth estimation method is able to provide a good reconstruction for some surfaces, like the road or walls, or object that are close to the ego vehicle, however as the disparity values get smaller, i.e. the objects from the scene get farther from the camera the disparity image is getting more blurry and the objects are not reconstructed as well as in the stereo vision case. Before combining the disparity values, we first check if the absolute value of the difference between the disparity value from the stereo and mono are below an error threshold (which has been empirically set to 10 in our case). Only if the condition holds true the two values are combined. Furthermore, we only use the disparity values from the monocular depth estimation which are above 60, because smaller values are unreliable. The fusion is performed using a set of weights for each semantic class. The combination between the values is computed using equation (10), where $disp$ represents the disparity taken from the disparity map obtained using stereo reconstruction, w_{class} is a weight selected experimentally for each class of interest, maxDisp represents the maximum number of disparities. The semantic classes weights for some of the classes of interest are the following w_{road} = 0.8, $w_{building}$ = 0.3, $w_{car} = 0.4$, $w_{pavement}$ = 0.6, $w_{sign} = 0.3$, $w_{bicicle} = 0.1$, $w_{pedestrian} = 0.1$. The other classes that appear in the KITTI semantic segmentation are not assigned any weights, their values remaining the same ones from the stereo reconstruction since the disparity values from the monocular depth estimation are unreliable for these classes. Such classes include grass, sky, fence, tree or discard class.

$$w_m = w_{class} * \frac{disp}{maxDisp} \quad (10)$$

The final result for a disparity value of coordinates i and j is computed as presented in (11), where for $\Delta_{i,j}$ is the final disparity value, $d\alpha_{i,j}$ is the disparity value obtained using the local stereo algorithm and $d\beta_{i,j}$ is the disparity value obtained using the monocular depth estimation.

$$\Delta_{i,j} = \left(1 - w_{m_{i,j}}\right) d\alpha_{i,j} + w_{m_{i,j}} d\beta_{i,j} \quad (11)$$

IV. Evaluation and Experimental Results

For evaluating the proposed solution, we have used the KITTI benchmark [41]. This dataset contains images of different driving scenarios, and it also provides ground truth files for depth estimation applications, taken using LIDAR sensors. Furthermore, the semantic and instance-based segmentation images are also provided for some of the traffic scenarios. The system on which we implemented our method contains an Intel(R) Core (TM) i5-7300HQ CPU, having 8GB of DDR RAM memory and an NVIDIA GeForce GTX 1050 4GB GDDR5 GPU. The neural network was trained using the GPU and the inference was also done on the GPU. Open MP has been used to parallelize some part of the code that runs on the CPU. The programming languages used are C++, Python and the frameworks used are OpenCV, TensorFlow and Point Cloud Library. The error threshold used in the error maps of our evaluation is 2 pixels. We evaluate the disparity images using the Out-All metric, which means that all pixels for which ground truth information exists will be evaluated even if they are occluded. In Figure 2 we can observe the results of the stereo algorithm, the MDE approach used and the proposed fusion solution, on a traffic scene from the KITTI dataset. We can also observe the error maps for each disparity image. The white pixels represent disparity values for which the difference with the ground truth data was greater than 2 pixels. In Table I the performance of the proposed solution is illustrated with respect to the used stereo and monocular depth estimation approaches. The methods are evaluated on the KITTI 2012 training set and the results are shown for error thresholds of 2,3 and 4 pixels.

TABLE I. Comparison of the Proposed fusion with each individual algorithm used

Method	Disparity Evaluation Error			
	2px	***3px***	***4px***	***Density***
Mono	18.202	13.940	9.649	100%
Stereo	7.324	6.168	4.888	100%
Proposed Fusion	5.756	4.446	3.000	100%

It is worth mentioning that the majority of errors encountered were at object borders and for semantic classes such as vegetation (especially at bushes) and fences. We can observe that the fusion of the information from the two disparity maps obtained using different approaches leads to better quality results incorporating the advantages of both methods. In Table II we have listed the comparison of the proposed solution with other state of the art methods. The evaluation is done using the Out-All error metric with an error threshold of 2 pixels. The running time of our solution is 100 ms, with the algorithms running on the CPU and GPU. The stereo reconstruction approach runs exclusively on the CPU, and the monocular depth estimation is running on the GPU.

TABLE II. Comparison with state of the art on KITTI

Position	Method	Density	Out-All Error
129	FD-Fusion [42]	100%	5.73 %
130	Proposed Fusion	100%	5.75%
131	OSF [43]	99.98 %	5.79 %
132	CoR [44]	100%	5.88%
133	SPS-St [45]	100%	6.28%

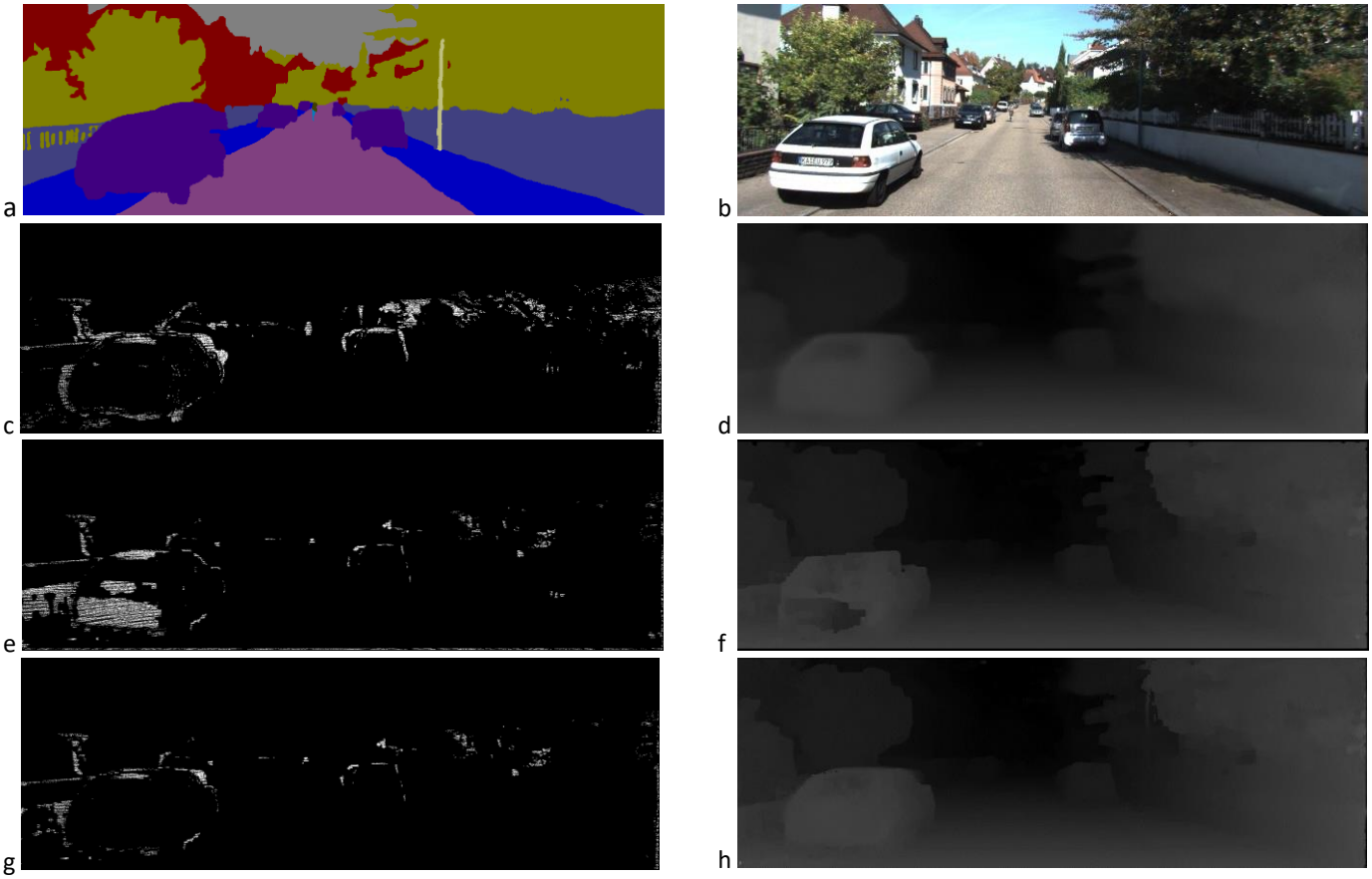

Figure 2. Comparison between the mono, stereo and proposed solution disparity maps. Figure a, presents the semantic segmentation, b is the original color image, c is the error map obtained for mono algorithm, d is the disparity map of the monocular depth estimation approach, f is the disparity map and e represents the error map of the local stereo algorithm, h represents the fused disparity map and g is the error map.

Other disparity images and the corresponding error maps obtained using the proposed fusion solution are displayed in Figure 3 and Figure 4. For visual analysis the Figure 3 corresponds to RGB image 38 and Figure 4 corresponds to RGB image 3 from the KITTI training dataset.

V. Conclusions and Future Work

In this paper we have presented an original pipeline that combines monocular with stereo depth estimation in order to produce high quality disparity maps. We have applied a tree-based refinement solution, which was originally used for stereo reconstruction, to the problem of monocular depth estimation in order to improve the resulted disparity map and proposed a speckle removal approach to eliminate small inconsistent values from the disparity maps. The proposed system is fault tolerant, meaning that if the stereo cameras fail to capture one image, the system will still provide a good quality disparity map using the monocular depth estimation method. Finally, if no image is acquired correctly a warning message is displayed. The proposed solution has been validated on the KITTI dataset and we have shown that the fusion approach is able to reconstruct scenes better than either the stereo or monocular depth estimation algorithms.

In future work we will investigate and create more efficient methods of mono and local stereo reconstruction that have a low resource consumption. Furthermore, we will also focus on improving the fusion approach using automatically

Figure 3. Disparity map of the proposed fusion and its error map using a 2-pixel threshold of a KITTI image

Figure 4. Disparity and error map of the proposed method applied on an image illustrating a parking lot.

learned weights and try to solve the issue of 3D reconstruction at the border of the objects.

ACKNOWLEDGMENT

This work was supported by the Romanian Ministry of Education and Research, through the CNCS UEFISCDI grant, Integrated Semantic Visual Perception and Control for Autonomous Systems (SEPCA), code PN-III-P4-ID-PCCF-2016-0180, grant no. 9/2018 and CNCS-UEFISCDI, project number PN-III-P4-ID-PCE-2020-1700, within PNCDI III.

REFERENCES

[1] M. P. Muresan and S. Nedevschi, "Multi-Object Tracking of 3D Cuboids Using Aggregated Features," 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing *(ICCP)*, 2019, pp. 11-18

[2] H. Kim, J. Cho, D. Kim, and K. Huh, “Intervention minimized semi-autonomous control using decoupled model predictive control,” in Proc. IV, Jun. 2017, pp. 618–623

[3] A. Arıkan, A. Kayaduman, S. Polat, Y. S¸ims¸ek, ˙I. C. Dikmen, H. G. Bakır, T. Karadag, and T. Abbasov, “Control method simulation and ˘ application for autonomous vehicles,” in Proc. IDAP, Sep. 2018, pp. 1–4

[4] M. Sualeh and G. -W. Kim, "Visual-LiDAR Based 3D Object Detection and Tracking for Embedded Systems," in IEEE Access, vol. 8, pp. 156285-156298, 2020

[5] A. Danzer, T. Griebel, M. Bach, and K. Dietmayer, "2d car detection in radar data with pointnets," in 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019, pp. 61-66: IEEE

[6] M. P. Muresan, M. Negru and S. Nedevschi, "Improving local stereo algorithms using binary shifted windows, fusion and smoothness constraint," *2015* IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 2015, pp. 179-185

[7] A. Bhoi, “Monocular depth estimation: a survey,” arXiv preprint arXiv:1901.09402, 2019

[8] M. P. Muresan, I. Giosan, and S. Nedevschi, “Stabilization and validation of 3d object position using multimodal sensor fusion and semantic segmentation,” Sensors, vol. 20, no. 4, p. 1110, 2020

[9] D. V. Nam and K. Gon-Woo, "Solid-State LiDAR based-SLAM: A Concise Review and Application," 2021 IEEE International Conference on Big Data and Smart Computing (BigComp), 2021, pp. 302-305

[10] S. Crowe. (2019, Apr.). Researchers back Tesla’s non-LiDAR approach to self-driving cars.

[11] R. Spangenberg, T. Langner, S. Adfeldt and R. Rojas, "Large scale Semi-Global Matching on the CPU," 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014, pp. 195-201

[12] Hailong Pan, Tao Guan, Keyang Luo, Yawei Luo, Junqing Yu, A visibility-based surface reconstruction method on the GPU, Computer Aided Geometric Design, Volume 84, 2021, 101956, ISSN 0167-8396

[13] H. Javidnia and P. Corcoran, “Accurate depth map estimation from small motions,” in ICCV, 2017, pp. 2453–2461

[14] S. H. Raza, O. Javed, A. Das, H. Sawhney, H. Cheng, and I. Essa, “Depth extraction from videos using geometric context and occlusion boundaries,” arXiv preprint arXiv:1510.07317, 2015

[15] V. Prasad and B. Bhowmick, “SfMLearner++: Learning monocular depth & ego-motion using meaningful geometric constraints,” in WACV, 2019, pp. 2087–2096

[16] B. K. Horn, “Shape from shading: A method for obtaining the shape of a smooth opaque object from one view,” 1970

[17] N. Kong and M. J. Black, “Intrinsic depth: Improving depth transfer with intrinsic images,” in ICCV, 2015, pp. 3514–3522

[18] A. Torralba and A. Oliva, “Depth estimation from image structure,” IEEE TPAMI, vol. 24, no. 9, pp. 1226–1238, 2002.

[19] S. H. Raza, O. Javed, A. Das, H. Sawhney, H. Cheng, and I. Essa, “Depth extraction from videos using geometric context and occlusion boundaries,” arXiv preprint arXiv:1510.07317, 2015.

[20] J.-I. Jung and Y.-S. Ho, “Depth map estimation from single-view image using object classification based on bayesian learning,” in 3DTV Conference, 2010, pp. 1–4

[21] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, “Deeper depth prediction with fully convolutional residual networks,” in 3DV, 2016, pp. 239–248

[22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016, pp. 770–778

[23] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in CVPR, 2018, pp. 7132–7141

[24] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, “From big to small: Multi-scale local planar guidance for monocular depth estimation,” arXiv preprint arXiv:1907.10326, 2019

[25] M. Mancini, G. Costante, P. Valigi, and T. A. Ciarfuglia, “Fast robust monocular depth estimation for obstacle detection with fully convolutional networks,” in IROS, 2016, pp. 4296–4303

[26] I. Alhashim and P. Wonka, “High quality monocular depth estimation 152 via transfer learning,” arXiv preprint arXiv:1812.11941, 2018

[27] W. Yin, Y. Liu, C. Shen, and Y. Yan, “Enforcing geometric constraints of virtual normal for depth prediction,” in ICCV, 2019, pp. 5684–5693

[28] M. Yue, G. Fu, M. Wu, and H. Wang, “Semi-supervised monocular depth estimation based on semantic supervision,” JIRS, 2020

[29] M. P. Muresan, S. Nedevschi, and R. Danescu, “A multi patch warping approach for improved stereo block matching,” in Proc. Int. Conf. Comput. Vis. Theory App., 2017, pp. 459–466

[30] N. Einecke and J. Eggert, "A multi-block-matching approach for stereo," *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 585-592, doi: 10.1109/IVS.2015.7225748

[31] Zhang, Y., Chen, Y., Bai, X., Yu, S., Yu, K., Li, Z., & Yang, K. (2020). Adaptive Unimodal Cost Volume Filtering for Deep Stereo Matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(07), 12926-12934

[32] H. Wang, R. Fan, P. Cai and M. Liu, "PVStereo: Pyramid Voting Module for End-to-End Self-Supervised Stereo Matching," in *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4353-4360, July 2021, doi: 10.1109/LRA.2021.3068108

[33] D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” International Journal of Computer Vision, vol. 47, no. 1, p. 742, May 2002

[34] F. Zhang and B. W. Wah, "Fundamental Principles on Learning New Features for Effective Dense Matching," in *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 822-836, Feb. 2018, doi: 10.1109/TIP.2017.2752370

[35] J. Sun, N.-N. Zheng, and H.-Y. Shum, “Stereo matching using belief propagation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 25, no. 7, pp. 787–800, Jul. 2003

[36] Y. Chen, C. Cai, and K.-K. Ma, “Stereoscopic video error concealment for missing frame recovery using disparity-based frame difference projection,” in 2009 16th IEEE International Conference on Image Processing (ICIP), Nov 2009, pp. 4289–4292

[37] T. Y. Chung, S. Sull, and C. S. Kim, “Frame loss concealment for stereoscopic video plus depth sequences,” IEEE Transactions on Consumer Electronics, vol. 57, no. 3, pp. 1336–1344, August 2011

[38] V. Miclea, L. Miclea and S. Nedevschi, "Real-time Stereo Reconstruction Failure Detection and Correction using Deep Learning," *2018 21st* International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 1095-1102, doi: 10.1109/ITSC.2018.8569928

[39] C. Godard, O. M. Aodha and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6602-6611.

[40] Y. Ji, Q. Zhang, K. Sugimoto and S. Kamata, "Disparity refinement with stability-based tree for stereo matching," 2015 IEEE Intelligent Vehicles Symposium (IV), 2015, pp. 469-474.

[41] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[42] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," Int. J. Rob. Res., vol. 32, pp. 1231-1237, 2013

[43] M. Ferrera, A. Boulch and J. Moras, "Fast Stereo Disparity Maps Refinement By Fusion of Data-Based And Model-Based Estimations," 2019 International Conference on 3D Vision (3DV), 2019, pp. 9-17

[44] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," *2015* IEEE Confderence on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3061-3070

[45] A. Chakrabarti, Y. Xiong, S. J. Gortler and T. Zickler, "Low-level vision by consensus in a spatial hierarchy of regions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4009-4017

[46] Yamaguchi K., McAllester D., Urtasun R. (2014) Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham