This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Real-Time Semantic Segmentation-Based Stereo Reconstruction

Vlad-Cristian Miclea and Sergiu Nedevschi, *Member, IEEE*

***Abstract*— In this paper, we propose a novel semantic segmentation-based stereo reconstruction method that can keep up with the accuracy of the state-of-the art approaches while running in real time. The solution follows the classic stereo pipeline, each step in the stereo workflow being enhanced by additional information from semantic segmentation. Therefore, we introduce several improvements to computation, aggregation, and optimization by adapting existing techniques to integrate additional surface information given by each semantic class. For the cost computation and optimization steps, we propose new genetic algorithms that can incrementally adjust the parameters for better solutions. Furthermore, we propose a new post-processing edge-aware filtering technique relying on an improved convolutional neural network (CNN) architecture for disparity refinement. We obtain the competitive results at 30 frames/s, including segmentation.**

***Index Terms*— Stereo reconstruction, semantic segmentation, deep learning, genetic algorithm, census, SGM, refinement.**

I. Introduction

DUE to the fast evolution of intelligent vehicles, real-time depth perception has become a major area of interest. Stereo reconstruction still remains one of the most feasible methods in depth perception due to its low-cost and high resolution output, being extremely useful for environment understanding [22], [26].

During the last several decades stereo reconstruction methods have been classified as being either local or global. Local methods rely on a small support window over which a similarity criterion is applied. Global methods compute the disparity of all pixels in the image by optimizing a global energy function. Scharstein and Szeliski [40] standardize the stereo reconstruction problem, by dividing it into four main phases – cost computation, aggregation, optimization, refinement, each phase being responsible for solving a particular sub-problem. Most of the approaches on Kitti [32] and Middlebury [39] benchmarks follow the standard taxonomy, proposing new deep learning methods for each of these sub-problems.

Manuscript received April 4, 2018; revised October 23, 2018 and February 8, 2019; accepted March 31, 2019. This work was supported in part by the UP-Drive Project, Automated Urban Parking and Driving, of the Horizon 2020 EU, under Grant 688652, and in part by the Romanian National Authority for Scientific Research (UEFISCDI), a Romanian National Research Agency, through the national research projects, PN III PCCF SEPCA (Semantic Visual Perception and Integrated Control for Autonomous Systems), under Project 9/2018, and in part by Multispectral Environment Perception by Fusion of 2D and 3D Sensorial Data from the Visible and Infrared Spectrum (MULTISPECT), under Project PN-III-P4-ID-PCE-2016-0727. The Associate Editor for this paper was Z. Duric. *(Corresponding author: Vlad-Cristian Miclea.)*

The authors are with the Department of Computer Science, Technical University of Cluj-Napoca, 400027 Cluj-Napoca, Romania (e-mail: vlad.miclea@cs.utcluj.ro; sergiu.nedevschi@cs.utcluj.ro).

Digital Object Identifier 10.1109/TITS.2019.2913883

Recently, deep leaning methods such as [23], [30] show that disparity can be directly estimated from the left and right images, end-to-end training being employed. These methods can obtain extremely accurate results, but they need a large amount of data for training and most of them cannot run in real time. Moreover, we are not 100% sure how such methods behave when dealing with an unrecognizable situation, which has not been met in training. This is of great importance especially in driving scenarios, where a total failure is extremely costly and must be avoided.

In this paper we try to follow the classic taxonomy by using improved variations of traditional methods for computation, aggregation and optimization steps. For each phase we aim for decent accuracy and focus on reducing computational cost when running on a regular GPU. We propose the use of semantic segmentation (Figure 1(b)) as a guidance for deep scene understanding. Finally, we replace the unreliable disparity pixels by using a CNN-based filter. The dense disparity map obtained after applying our stereo reconstruction algorithm (without post-processing) is shown in Figure 1(c), while Figure 1(d) depicts the filtered disparity map, obtained after refinement. The main contributions of this paper are:

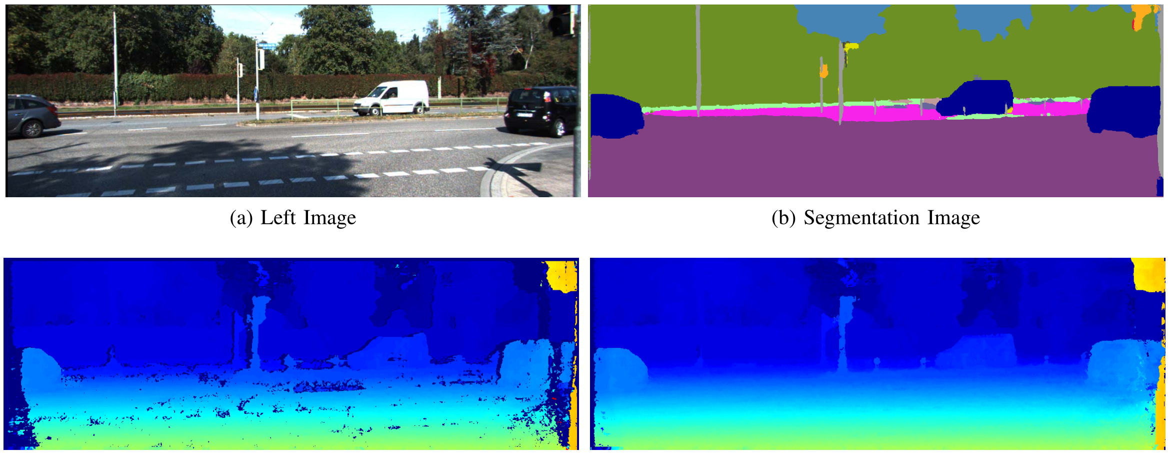
- a collaborative integration of semantic segmentation into stereo reconstruction;
- a novel optimal low-cost Census-based cost computation adapted to each particular segment class;
- an enhanced cost aggregation scheme that incorporates object boundaries;
- an optimization technique based on SGM that adapts P_1 penalty to surfaces;
- a refinement filter obtained with a CNN capable to increase the disparity reliability in driving scenarios;
- real-time (30 fps on a regular GPU) results, with an accuracy close to state of the art.

Section II deals with presenting the state of the art in stereo reconstruction solutions. Section III shows semantic segmentation-based improvements for cost computation, aggregation and optimization and explains the genetic algorithms employed for these tasks. We describe the neural network proposed for post processing in Section IV. The effectiveness of our stereo method (for driving scenarios) and of our refinement (in particular) is shown in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

A. Classic Taxonomy

Cost computation algorithms generally rely on a metric to find similarities between patches from left and right images.



(a) Left Image (b) Segmentation Image

(c) Disparity Image (d) Filtered Disparity Image

Fig. 1. Disparity maps obtained with our method on traffic images.

Traditional metrics [18] are either intensity-based [47] – SAD, SSD, NCC or non-parametric – Rank Transform, Census Transform [50]. Lately, feature-based approaches have been used for this step. Such methods either use hand-crafted features [45] or enable learning skills by using convolutional neural networks [35], [51].

In terms of aggregation most of the approaches [1] prefer choosing a squared support window. According to [13] this window should not exceed 7×7 pixels, whilst 5×5 represents a good trade-off between accuracy and speed. Particular improvements have been proposed by [31], in which the cross-based cost aggregation is introduced. This approach controls the aggregation window by 2 parameters corresponding to locality and intensity similarity. A different technique is proposed by [25] in which a residual neural network is responsible for edge detection. Edges are detected at different scales and are later used for setting the aggregation window boundaries.

Semi-global matching (SGM) [19] is one of the most robust optimization algorithms, ensuring close-to global consistency while consuming a reasonable amount of resources. The method approximates a 2D energy minimization by several 1D optimizations. There are various implementations of SGM on different platforms CPU [43], GPU [15] or FPGA [3], most of them obtaining real-time performances. Other similar optimization methods are Graph Cuts [24] or Belief propagation [44], all of them being very expensive in terms of speed.

The last step in the workflow is disparity refinement. In this phase unreliable parts of the disparity are replaced with pixels with higher confidence. This is generally accomplished by applying classic (edge-aware) filtering techniques such as median filters [20], bilateral [36], [46] or guided [16].

More recently, a new category of edge-aware filters have been introduced. Classic filters are approximated by optimization mechanisms [5], [48], obtaining more accurate results, with higher computing performances. An edge-aware deep learning-based guided filter is proposed by [27]. The method introduces a joint convolutional neural network that enhances a target image by using priors extracted from its RGB counterpart.

B. End-to-End Stereo

Mayer *et al.* [30] propose an encoder-decoder like neural network architecture. Left and right images are stacked together and pass through several fast convolutions and deconvolutions, resulting in an accurate disparity map obtaining also a very good computation time. Moreover, the paper introduces a large training synthetic dataset with dense ground truth containing scenarios for both driving and indoor applications.

One of the top approaches currently on the Kitti 2015 dataset is GC-Net [23]. The solution exploits extensive 2D and 3D convolution layers for feature and context extraction. Additionally, it uses a probabilistic disparity selection method (differentiable soft-argmin) that facilitates end-to-end training. Besides these, stereo reconstruction can be computed even without a humanly generated ground truth, by employing either self-supervised [53] or unsupervised [54] learning.

C. Semantic Segmentation-Based Stereo

Most of the algorithms that combine image segmentation with stereo try to increase disparity accuracy by using object information in the post processing step. The authors of [**?**] propose a plane fitting-based segmentation by filling the disparity map relying only on confident disparity pixels. The authors of [49] achieve the same goal by using super-pixels as means to group similar pixels. The problem with all these implementations is the increased computational effort making them not viable for real-time usage.

More recently semantic segmentation has become more and more reliable for confidently highlight scene objects. One of the most accurate methods on Kitti dataset [32] – Displets [14] – rely on the similarity given by specific object structures to fill sparse disparity estimates. In a more recent article [41] the authors propose to enrich the scene information and obtain more reliable results by using semantic information in relation with their stixel-based stereo method.

III. STEREO PROCEDURE

In our work we propose to apply a "divide and conquer" approach, separating the driving scene into homogeneous

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

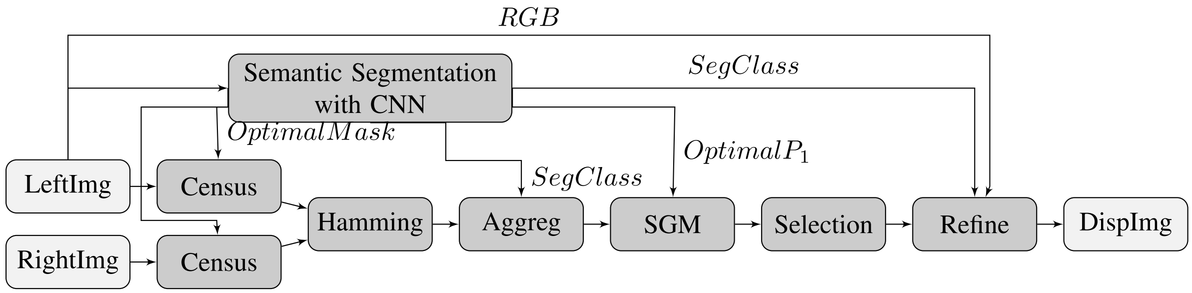


Fig. 2. Workflow of the proposed method.

regions. In the initial step of the algorithm we train a convolutional neural network for accurate pixel-wise scene segmentation. Then, for each class we determine an optimal census mask, an improved aggregation window, an optimal SGM penalty P_1 and reliable information for a better refinement. An overview of the proposed stereo method can be seen in Fig. 2.

A. Semantic Segmentation

The initial step in our solution is to compute a semantic segmentation of the scene. Classic segmentation methods have been combined with stereo [49], generally being used as a post-processing [21]. However, we can now take advantage from the boost that semantic segmentation lately received with the introduction of deep neural networks. Cityscapes dataset [8] enables methods such as [29] or [37] to accurately classify object categories at pixel level.

One of the most robust approaches in semantic segmentation is ERFNet [38], having one of the best trade-offs in terms of accuracy (69.7 for IoU) vs speed (around 20 ms on a NVIDIA Titan X). The method uses an encoder-decoder architecture, with 23 layer blocks. The key for speed and precision is their novel layer block – a mixture of residual connections and factorized convolutions that preserves the structure in the image and reduces computational costs. The method classifies the scene into 19 foreground and background classes. We slightly modify several parts in this method to accommodate the needs in our stereo pipeline. Therefore our solution will only need a subset of classes:

- For the computation and optimization phases we are more interested in surface types than in object boundaries. We have divided the object scene into 7 classes: road, vehicles, traffic signs, buildings, sidewalk, vegetation, terrain, that correspond to horizontal, vertical, slanted, or more complex surfaces. This division is empirically selected through exhaustive testing, while observing that increased segmentation granularity might lead to increased computational costs.
- For aggregation and refinement the segmentation map will include all 20 (19 + unknown) segment classes. In this case we need more a-priori object information, and edge detection is really important.

B. Cost Computation

State of the art [32] shows that deep-learning approaches are viable for cost computation. However, very good accuracy comes with increased computational workload and such methods cannot yet achieve real-time performance and might fail when dealing with unrecognizable data. Therefore, we rely on classic non-parametric methods that are less efficient but fast and reliable (because of their geometric nature).

While intensity-based metrics suffer from poor adaptation to bad illumination conditions, Census Transform stands out as the best trade-off between accuracy and speed. Moreover, its center-symmetric implementation is shown to further improve the accuracy [43]. Therefore we compute the cost by:

$$C_{comp}(p,d) = Ham(T_l(p, N(p)), T_r(p-d, N(p-d))) \quad (1)$$

where Ham is the Hamming distance, T is the Census Transform of the image patch centered in p. The pixels selected for the image patch are given by the bitstring census mask:

$$N(p) = \sum_{i \in Neigh(p)} b_i 2^i \quad (2)$$

where $Neigh$ is the chosen neighborhood, and b_i is either 1 or 0 according to the importance of the pixel.

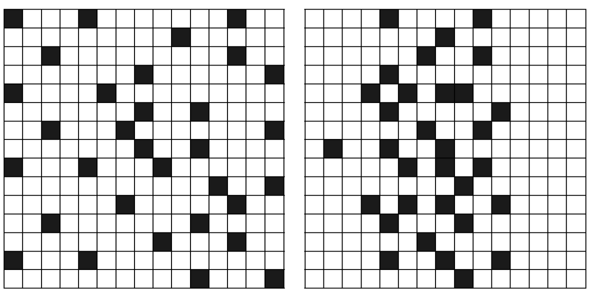
We introduce here a method to find the optimal census masks for each particular segment class. The method uses stochastic optimization based on genetic algorithms to optimally generate a bitstring for each particular segment class. As presented in our previous works [33], [34], a viable census mask usually covers a surface of maximum 15×15 pixels, giving enough information and allowing for maximum 32 pixels to be selected. Moreover, the computation time increases proportionally to each additional pixel, so larger census windows might lead to lower frame rates. We present the methodology for finding a segment-dependent optimal census mask in Algorithm 1.

The initial population is composed by a set of randomly generated census masks (bitstrings). For each member of a population we define a fitness function as being the percent of misclassified pixels (obtained with that specific census mask) with respect to a given ground truth. We optimize the census mask by means of selection (best k census masks in a generation), crossover (interchanging two top-selected individuals) and mutation (randomly flipping several bits). We run the optimization procedure until the difference between the best results obtained in several consecutive generations is smaller than a predefined threshold.

To sum up, according to this method, we add an additional parameter to the neighborhood selection and optimally select

Algorithm 1 Algorithm for Optimal Census Mask

```
procedure GENERIC ALG. FOR CENSUS
    for all segments do
        initialize population(0)
        d_CT ← apply CT(population(0))
        population.fitness(0) ← err(d_CT, d_GT)
        repeat
            perfrom selection, crossover and mutation
             on population(i)
            partially initialize population(i+1)
            population(i+1) ← population_mut(i) +
               population(i+1)
            d_CT ← apply CT(population(i+1))
            population.fitness(i+1) ← err(d_CT, d_GT)
        until i=finalGeneration
    end for
end procedure
```



Census mask for road surface   Census mask for vertical surfaces

Fig. 3. Pixels used for census in cases of road and vertical surfaces.

a census mask for each particular semantic class. Therefore, the census mask is given by:

$$N(p, S(p)) = \sum_{i \in Neigh_s} b_i 2^i \tag{3}$$

Figure 3 exemplifies two census masks (for road surfaces vs small vertical surfaces). On the left side we have showed a generated mask for road surfaces. The GA selects pixels such that the entire window is covered. On the other hand, the right image shows the window that corresponds to vertical surfaces (i.e. poles). In this case the GA selects only pixels from the center of the image, without accounting for pixels at left and right ends.

We choose this approach since it extracts important features for each type of surface in the scene, without any cumbersome deep learning procedure that may increase the resource cost.

C. Cost Aggregation

The key aspect for a good aggregation scheme is finding similar neighboring pixels. Generally, similar pixels within a predefined squared window are selected:

$$C_{Aggr}(p, d) = \sum_{i \in Neigh(p)} C_{comp}(i, d) \tag{4}$$

where the $Neigh(p)$ is a squared window centered in p.

However, a predefined window is not always beneficial since it accounts only for locality, not for intensity similarity. A better aggregation technique is to control the window expansion using two parameters: λ, corresponding to locality, and σ, responsible for intensity. Therefore, the window N(p) is adapted such that it satisfies:

$$(\|p - i\|_2 < \lambda) \wedge (|I(p) - I(i)| < \sigma) \tag{5}$$

where $\|p - i\|_2$ is the Euclidean distance between the pixels p and i, while the values for σ and λ are predefined. In our Kitti-related experiments, $\sigma = 5$, while $\lambda = 10$. This trick is shown to alleviate some of the aggregation errors, but it might still aggregate unreliable pixel information (beyond edges).

As we try to rely on classes obtained through semantic segmentation we propose a new aggregation technique in accordance to the segmentation. Therefore, we modify the formula in equation 5 such that it includes a third term, that prevents the aggregation window expansion from including pixels outside object boundaries. The aggregation window $Neigh(p, S(p))$ contains an additional parameter and it becomes controlled by:

$$(\|p - i\|_2 < \lambda) \wedge (|I(p) - I(i)| < \sigma) \wedge (S(i) = S(p)) \tag{6}$$

where $S(i)$ is the segment class of pixel i.

D. Optimization

We use the SGM optimization technique. The most critical part in SGM is the penalty selection: good values of P_1 – penalty for small disparity changes and P_2 – penalty for large disparity disruptions are necessary. Even though ideally we would have particular penalties for each pixel (generated through deep learning [42]), we prefer to adapt them for each segment class $S(p)$ and for each direction r. Therefore, the SGM optimization formula becomes:

$$\begin{aligned} L_r(p, d) = C_{Aggr}(p, d_p) + min(&L_r(p - r, d_p), \\ &L_r(p - r, d_p - 1) + P_1(r, S(p)), \\ &L_r(p - r, d_p + 1) + P_1(r, S(p))), \\ &min_{k \in D} L_r(p - r, k) + P_2(r)) \end{aligned} \tag{7}$$

$$C_{Opt}(p, d) = \sum_r L_r(p, d_p) \tag{8}$$

For this step we also use a stochastic optimization based on Genetic Algorithms (GA). All $P_1(r, S(p))$ and $P_2(r, S(p))$ values are plugged in the algorithm and we optimize according to the resulting pixel error (with respect to a specific ground truth). Kitti2015 [32] training images are used for optimization.

Early results show that error largely fluctuates when both P_1 and P_2 are introduced into the optimization scheme. We choose to only adapt the P_1 value, selecting a P_2 penalty that only depends on its neighbor intensities (as in [4]):

$$P_2(r) = \frac{P_2'}{|I_L(p) - I_L(p - r)|} \tag{9}$$

where P_2' is predefined. For most of our experiments, $P_2' = 35$.

This choice is compliant with our semantic segmentation-driven method since we can fully rely on a-priori detected

Algorithm 2 Algorithm for Optimal SGM Penalty P_1

procedure GENERIC ALG. FOR P_1
  for `seg = 1 to all segments` **do**
    initialize population(0) with $P_1(seg)$
  end for
  $d_{SGM} \leftarrow$ *apply SGM(population(0))*
  population.fitness(0) $\leftarrow err(d_{SGM}, d_{GT})$
  repeat
    perfrom selection, crossover and mutation on population(i)
    for `seg = 1 to all segments` **do**
      partially initialize population(i+1) with $P_1(seg)$
    end for
    population(i+1) $\leftarrow$ *population_mut(i)* + *population(i+1)*
    $d_{SGM} \leftarrow$ *apply SGM(population(i+1))*
    population.fitness(i+1) $\leftarrow err(d_{SGM}, d_{GT})$
  until `i=finalGeneration`
end procedure

object boundaries. It results in a total number of $\#params$ of additional P_1 parameters, where $\#params$ is defined as:

$$\#params = \#directions \cdot \#classes \tag{10}$$

The stochastic optimization algorithm used for P_1 generation is presented in Algorithm 2. We consider that GA converges when the error criterion ϵ (equation 11) is smaller than a predefined threshold or the maximum number of iterations is reached.

$$\epsilon = \sum_{r,N_s} |P_1(r, s(p))(i) - P_1(r, S(p))(i+1)| \tag{11}$$

where i is the current iteration number.

Disparity is then selected according to the WTA (winner takes all) approach. Left-right consistency checking is done, unreliable pixels being eliminated.

IV. DISPARITY REFINEMENT

A. Filtering as Post-Processing

Like in previous stereo steps, the key for a good refinement is to find image edges so that scene objects are clearly delimited one from another. While the usage of the fast (as in computation time) median filter is quite limited only to small disparity gaps, bilateral and guided filters need too many computational resources.

We rely here on classes given by the semantic segmentation, which will define areas with similar characteristics. Due to the high scene complexity, for this part we propose a new filtering scheme based on deep learning.

B. Refinement Architecture

For the learning-based refinement filter we employ a 3-input ConvNet architecture (Fig. 4). The first part of the network consists in two similar branches, with the role of extracting reliable features from both depth and RGB image.

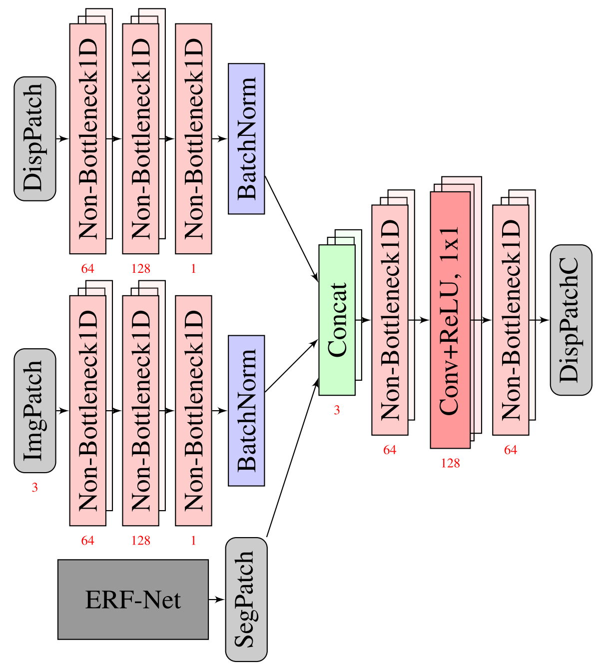


Fig. 4. Architecture of the proposed filter.

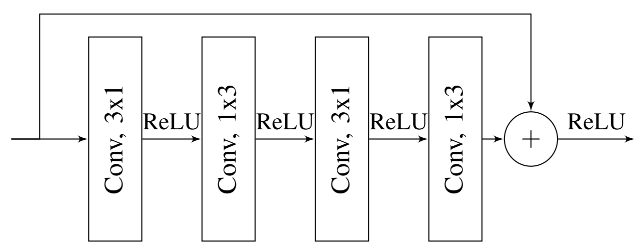


Fig. 5. Non-bottleneck1D block having a receptive field size of 5×5.

A 30×30 patch from the Left RGB image is the input to the first branch, while a patch from the incomplete disparity (resulted from optimization step) is plugged in to the second one. Each branch consists of three residual Non-bottleneck1D blocks, followed by a Batch Normalization layer. The first block contains 64 feature maps, the second 128, while the third produces just one feature map, that incorporates the most relevant features extracted from each branch.

The convolution layers are designed using the speed-up techniques presented in [38]. A Non-Bottleneck1D (Fig. 5) block is therefore shaped by:

- Residual connections – important information extracted from initial layers is preserved throughout the entire network so that later layers can benefit from it;
- 2D convolution layers approximated by two 1D convolutions – this trick reduces the number of convolution weights by more than a half while preserving stability and accuracy;
- ReLU units inserted after each convolution – used to zero the gradients on negative input values.

Exhaustive testing showed us that RGB features are not enough to provide effective guidance for the filter. This problem is mainly caused by the mixture and the variety of information RGB maps carry. Although the first part of the network tends to extract more effective features and provide relevant information, we consider useful to aid this process with an additional term. Therefore results of the two branches are also concatenated with a patch extracted from the segmentation image. The semantic segmentation map is the third set of important features for our network, containing information about object boundaries and linking together similar structures.

The second part of the network consists in two additional Non-Bottleneck-1D blocks, interleaved by a layer of 1×1 convolutions. The role of the second part is to join together the three maps and generate a more reliable disparity, simulating a non-linear regression. This way we will integrate the knowledge extracted from the three aforementioned feature maps. We avoided concatenating multiple feature maps from RGB and incomplete depth branches for two reasons: 1) to keep the complexity of the regression branch as low as possible; and 2) to keep the weight of semantic map equal to the other two.

Numerically, a pixel-wise mean squared error is then computed between the resulting completed depth patch and the ground truth, thus estimating the degree of convergence for our method.

C. Parameters and Training Details

All patches are normalized by subtracting the mean and dividing with the maximum image intensity. Similar learning rates have been given to both depth and image branches. Experimental testing showed that our network converged only when the segmentation learning rate was set to 1/5 of the learning rate for RGB and Depth. In other scenarios segmentation features became too powerful, and depth information was dropped. We tried two optimization methods: Stochastic Gradient Descent and Adaptive Moment Estimation (Adam). Adam seemed to properly control the learning so we chose it as our optimizer.

We trained the network for 400 epochs, with a batch size of 128, decreasing the learning rate with a factor of 0.1 at the interval of 100 epochs.

V. Experimental Results

The main prerequisites for training are:

1) RGB left images for semantic segmentation with driving scenarios
2) accurate ground truth (GT) depth images for training the GA in computation and aggregation
3) dense accurate ground truth for refinement filtering

The semantic segmentation network [38] is trained on the Cityscapes dataset, using Torch7 framework [7], the results being normalized according to Kitti 2015 mean and standard deviation.

Although Kitti 2015 dataset is adequate for the first two needs, the GT it provides is sparse (given by LIDAR). DispNetC [30] provides the dense GT we need, but its synthetic nature reduces the inherent difficulties found in real situations (eg. unexpected illumination, scene complexity). A different option is to choose the disparity obtained with a top stereo method (MC-CNN acrt [51]) as our dense ground truth. While this method has a low error rate on the evaluated Kitti pixels, it is adapted to real-life situations. We choose a mixture of these two, benefiting from both pixel-wise accurate ground truths and real-life driving scenarios. The training set is divided into training (80%) and validation (20%) subsets.

While our stereo method and the genetic algorithms are implemented in CUDA, the refinement CNN is trained using Torch7. The testing of our refinement network (forward pass) is performed by using our own CUDA-based deep learning framework. The framework resembles Torch7, using NVIDIA CuDNN [6] primitives for deep learning layers. This allows us to integrate all phases in a single application.

A. Stereo Method

1) Results Obtained for Various Census-Based Cost Computation Methods: We test the following masks:

- Star – the pattern proposed by [28]. This is a symmetric pattern containing 32 pixels inside a 9×9 window;
- Center-avoiding – the pattern introduced in [13]. This pattern selects pixels that are situated at a 2-3 pixel distance from the center, neither too far, nor too close;
- Dense – This pattern is the most simple one. It accounts for all the pixels in the image. Because of its proportional spreading to size feature, it gives larger processing times for larger windows. We considered here a 7×7 window;
- GA – This is the method based on the genetic algorithm proposed in [33], selecting an optimal census mask for the entire set of images. It selects optimally 32 pixels inside a 11×11 window so that resulting cost is represented on 32 bits.
- GA + Seg – The optimal masks and optimal P_1 are given by the proposed genetic algorithms 1 and 2. This contains a set of 6 census masks, each of them containing specific (max 32) pixels inside a 11×11 window and 6 penalties ranging from 7 to 35.

The results obtained at pixel-level are presented in table I. A first observation is that since more than 50% of the pixels belong to the road surface (Seg 1), algorithms that lead on that specific surface (Dense, and GA-based) top the overall ranking. Another remark is that although regular (dense) CT thrives on regular surfaces – fronto parallel (Seg 3 and Seg 4) and road, it behaves poorly on irregular objects such as vegetation (Seg 6) and terrain (Seg 2).

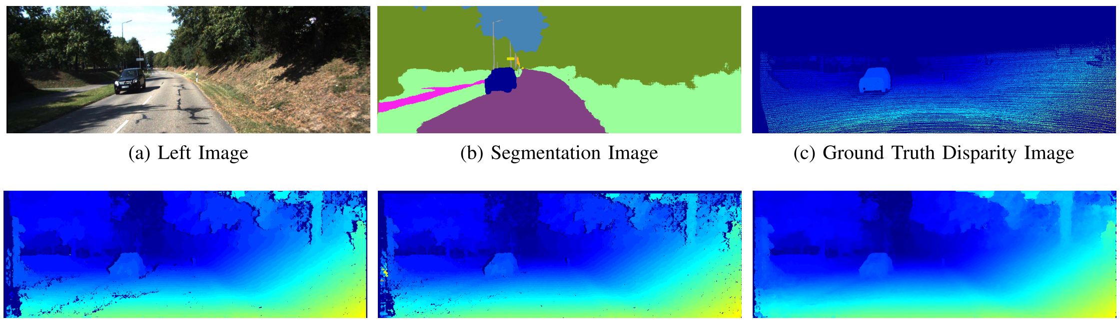
All in all, we can notice that our newly introduced approach ranks first in this classification, outperforming the non-segmented GA approach with almost 10% for the Census-only case, and the other methods by more than 17%. However, this margin strongly decreases when we introduce the energy minimization term. This happens because the SGM energy minimization compensates for the lack of correlation accuracy. An additional uncertainty is added by the inherent segmentation errors at object interactions so we can say that our method would work even better with improved semantic segmentation techniques [9].

2) Results Obtained With Our Stereo Method on Kitti Dataset: In order to evaluate our stereo system, we first show the behavior of each particular phase in our algorithm. We evaluate the speed and accuracy for the enhancements introduced with respect to a regular classic SGM implementation. The classic SGM is composed by a regular 5x5 Census,

TABLE I

AVERAGE ERROR PERCENT FOR CENSUS-ONLY AND SGM OBTAINED WITH VARIOUS CENSUS MASKS WITH ERROR THRESHOLD $T = 3$

	Seg1		**Seg2**		**Seg3**		**Seg4**		**Seg5**		**Seg6**		**Total**	
Method	Census	SGM	Census	SGM	Census	SGM	Census	SGM	Census	SGM	Census	SGM	Census	SGM
Star	89.09%	6.84%	65.51%	15.75%	74.59%	12.01%	65.15%	25.51%	71.14%	25.44%	84.25%	15.46%	82.46%	13.26%
CenterAv	92.87%	11.61%	70.23%	15.48%	82.62%	16.39%	67.59%	25.11%	74.80%	25.45%	89.67%	14.93%	87.79%	13.98%
Dense	83.45%	5.86%	72.16%	16.15%	71.63%	11.58%	62.07%	23.14%	69.10%	25.00%	89.43%	17.67%	79.64%	13.25%
GA	76.49%	4.29%	53.02%	14.17%	60.80%	13.06%	54.32%	21.41%	63.61%	23.78%	72.24%	12.88%	70.40%	11.82%
GA+Seg	63.02%	3.92%	51.32%	14.17%	60.32%	12.74%	67.19%	20.56%	63.62%	23.15%	71.60%	12.34%	59.27%	10.19%



(a) Left Image

(b) Segmentation Image

(c) Ground Truth Disparity Image

(d) Disparity Image using regular SGM

(e) Disparity Image using proposed stereo solution

(f) Filtered Disparity Image

Fig. 6. Disparity maps obtained with various methods on Kitti 2015 images.

TABLE II

EXECUTION TIME INCREASE PER EACH STAGE (MS) VS ERROR REDUCTION FOR KITTI 2015 TRAINING IMAGES (1242×375) WITH A MAXIMUM DISPARITY OF 128

Method	**Evaluate**	**Seg**	**+ Census**	**+ Aggreg**	**+ SGM**	**+ PostProc**
Classic SGM	Error	-	79.64%	29.43%	14.7%	13.25%
	Total Time	-	1 ms	3 ms	9 ms	9 ms
Proposed	Error	-	59.27%	24.42 %	10.19%	5.25%
	Total Time	17 ms	18 ms	20 ms	26 ms	34 ms

5x5 squared aggregation, SGM optimization with fixed parameters, followed by a median filter. Figure 6 (d) reveals the dense disparity map obtained with the classic SGM, while Figure 6 (e) shows the improved disparity given by our solution. 17 ms are required for the four stereo steps (of which 8 ms for refinement). The post-processing network gives the largest accuracy improvement, reducing the overall error with almost 5%. Figure 6 (f) shows the filtered image, in which pixels from difficult areas (eg. around shadowed surfaces and edges) get reconstructed correctly. All other steps prove to outperform their regular counterparts by introducing only a 1 ms delay. We show that we can further optimize each step while introducing only a small number of additional computations. Numerical results can be seen in Table II.

B. Refinement

1) Filter Variation: For this part we use the same baseline stereo method, and see how our filter behaves in comparison with other state of the art approaches. We chose some of the most commonly used refinement filters: median [20], bilateral [46], guided [16], fast bilateral [36] and DeepJoint [27]. We implemented our own median and fast bilateral filters and used the OpenCV implementations for the bilateral and guided filters. DeepJoint filter has been trained using images from Kitti2015 dataset. Our filter is shown to outperform its counterparts by a large margin while maintaining a relatively low time consumption. This evaluation also reveals the improvement we obtained (around 2.5% wrt the joint filter) by introducing the segmentation map into the CNN. Table III shows numerical results.

TABLE III

PERFORMANCE OF VARIOUS POST-PROCESSING TECHNIQUES

Method	**Accuracy**	**Speed**	**Platform**
Without	10.19%	-	-
Median [20]	9.37%	0.3 ms	GPU (CUDA)
Bilateral [47]	9.22%	8 ms	CPU (C++)
Guided [16]	9.07%	20 ms	CPU (C++)
Fast bilateral [37]	9.17%	180 ms	CPU (C++)
DeepJoint [28]	7.72%	100 ms	GPU (Matlab)
Proposed	**5.25%**	**8 ms**	**GPU (CUDA)**

TABLE IV

ACCURACY OF THE FILTER WHEN APPLIED TO VARIOUS STEREO METHODS

Method	**Error**	**Error(+PostProc)**
Census-only	59.27%	68.67%
OCV-BM	30.89%	29.86%
OCV-SGBM	14.75%	11.65%
MC-CNN fast	3.79%	3.23%
MC-CNN acrt	3.03%	2.77%
Proposed	10.19%	5.25%

2) Refinement Behavior When Changing the Stereo Method: We are also interested to see how our filter behaves when the underlying stereo method is changed. We chose the following stereo methods as input to post-processing: Census-only, Block Matching (BM) and Semi-Global Block Matching (SGBM)

TABLE V
ERROR OBTAINED IN EACH STEP WITH MISSING/ERRONEOUS SEMANTIC INFORMATION

SemanticInfo	+ Census	+ Aggreg	+ SGM	+ PostProc
With ERF-Net	59.27%	24.42 %	10.19%	5.25%
Without	62.11%	30.15 %	14.06%	11.41%
Erroneous	67.45%	29.77 %	13.83%	11.88%
With PSPNet	58.88%	23.83%	10.02%	4.98%
With Seg GT	56.23%	20.34%	9.20%	4.05%

from OpenCV, MC-CNN fast, MC-CNN accurate from [51] and our method. For each of these methods we performed left-right consistency to eliminate wrong results (false matches). The post-processing network was trained only once, with patches from a subset of Kitti and DispNetC images.

Since the goal for our network is to filter out only small disparity inconsistencies, it cannot cope with the large errors found in Census-only case. Stereo solutions with an average error can really benefit from our refinement, the error largely decreasing for our method and for OpenCV-SGBM. On the other hand, stereo solutions with low error can only marginally gain from post-processing, since they use other optimization mechanisms. Numerical results are shown in Table IV.

C. Results With a Better, Without or With Misleading Semantic Maps

In order to demonstrate the robustness (with respect to driving scenarios) of our method we test its behavior when a correct semantic map information is unavailable. Since convolutional neural networks are employed for the semantic task, we consider possible that an unrecognizable situation will appear (possibly untrained by algorithm creators). Furthermore, we would like to see the results we can obtain with our stereo method by using the best segmentation possible. We consider the following cases:

- the semantic map is missing. This is the most common case, in which the semantic information is not available;
- the semantic map is wrong. In this case we test if a bad semantic segmentation will mislead the stereo solution.
- the semantic map is more accurate. In this case we test how much the stereo results will modify if the segmentation was computed with a different CNN. For this case we used PSPNet [52], a method that performs better than ERF-Net (81.2 IoU for Cityscapes images).
- the semantic map is perfect. For this case we used the semantic ground truth from Kitti dataset [2].

Cost computation and optimization steps are less dependent on learnable features at testing phase – a bad segmentation will only prompt the cost computation and optimization algorithms to use a wrong census mask and P1 penalty, giving sub-optimal results. Aggregation is more dependent on the segmentation especially in the areas around edges (a bad segmentation might include unwanted information in the aggregation scheme or might exclude relevant costs). Furthermore, refinement will largely suffer since $1/3$ of the features we plug-in to the regression branch will be incorrect. Since these blocks directly rely on semantic data for edge information, they require further testing.

The results are presented in table V. The overall error increases if the semantic information is not present (around 5% for computation, 4% for aggregation, 2% for optimization and 2-3% for refinement). For aggregation the error percent is larger when no semantic information is given whereas in the computation, optimization and post-processing a misleading semantic map will confuse the genetic algorithms and the refinement filter.

On the other hand, we can observe that the smallest number of mismatched pixels is obtained when the best segmentation map (GT) is employed. Also, our results show that a better segmentation (PSPNet) leads to better stereo solutions.

All in all, we can see that since our method only relies on learning as an augmentation process to a well-built geometric method, in case the CNN fails (eg. in scenarios that have not been met during training), our method still gives a decent disparity map. On the other hand, an end-to-end stereo method will most likely fail when dealing with a similar situation.

D. Results on Kitti 2015 Testing Dataset

In addition to the previously mentioned testing, we also show the results obtained with our method on the Kitti 2015 testing dataset. As top approaches rely on feature-based cost computation (expensive in terms of speed), we compare here only approaches on the dataset that can (almost) run in real-time ($t < 100ms$). We show the error given by the mismatched pixel percent (with a threshold error of 3) for both background and foreground non-occluded pixels. Numerical results can be seen in Table VI.

DispNetC [30] gives very accurate results and this can be seen especially for foreground objects. In terms of accuracy, our solution is as good as DeepCostAggr [25], both methods relying on good object boundary estimates for enhanced reliability. These results prove that our refinement method can compensate the reduced accuracy of cost computation. All CNN-based methods outperform the accuracy of classic counterparts, but need dedicated hardware implementations to keep up with the real-time constraint. Moreover, both of these methods rely on CNNs to directly compute the disparity so are susceptible to undesired errors caused by untrained scenarios. This drawback makes them not viable for autonomous driving applications. In contrast, our method will not suffer from this drawback (as we showed in previous section).

In terms of time performance our method is shown to run at 30 fps on a regular GPU. However, half of this time is actually consumed by semantic segmentation and this task is generally required by other algorithms in perception. All in all the results obtained for both accuracy and time performance

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE VI
AVERAGE ERROR OF ON KITTI 2015 TESTING DATASET FOR NON-OCCLUDED PIXELS

Method	**D1-all**	**D1-bg**	**D1-fg**	**Speed (ms)**	**Platform**	CNN
DispNetC [31]	4.05%	4.11%	3.72%	60	Nvidia GTX Titan X (Caffe)	Yes
DeepCostAggr [26]	5.61%	4.82%	10.11%	30	GPU @ 2.5 Ghz (C/C++)	Yes
AABM [12]	6.26%	4.49%	15.22%	80	1 core @ 3.0 Ghz (C/C++)	No
SNCC [11]	6.69%	5.00%	15.21%	80	1 core @ 3.0 Ghz (C/C++)	No
CSCT+SGM+MF [17]	6.56%	5.37%	12.58%	6.4	Nvidia GTX Titan X (CUDA)	No
PCOF + ACTF [10]	8.03%	5.98%	18.40%	80	1 core @ 3.0 Ghz (C/C++)	No
Proposed	**5.67%**	**4.83%**	**10.75%**	**34 (17+17)**	**Nvidia GTX 1080 (CUDA)**	**Yes**

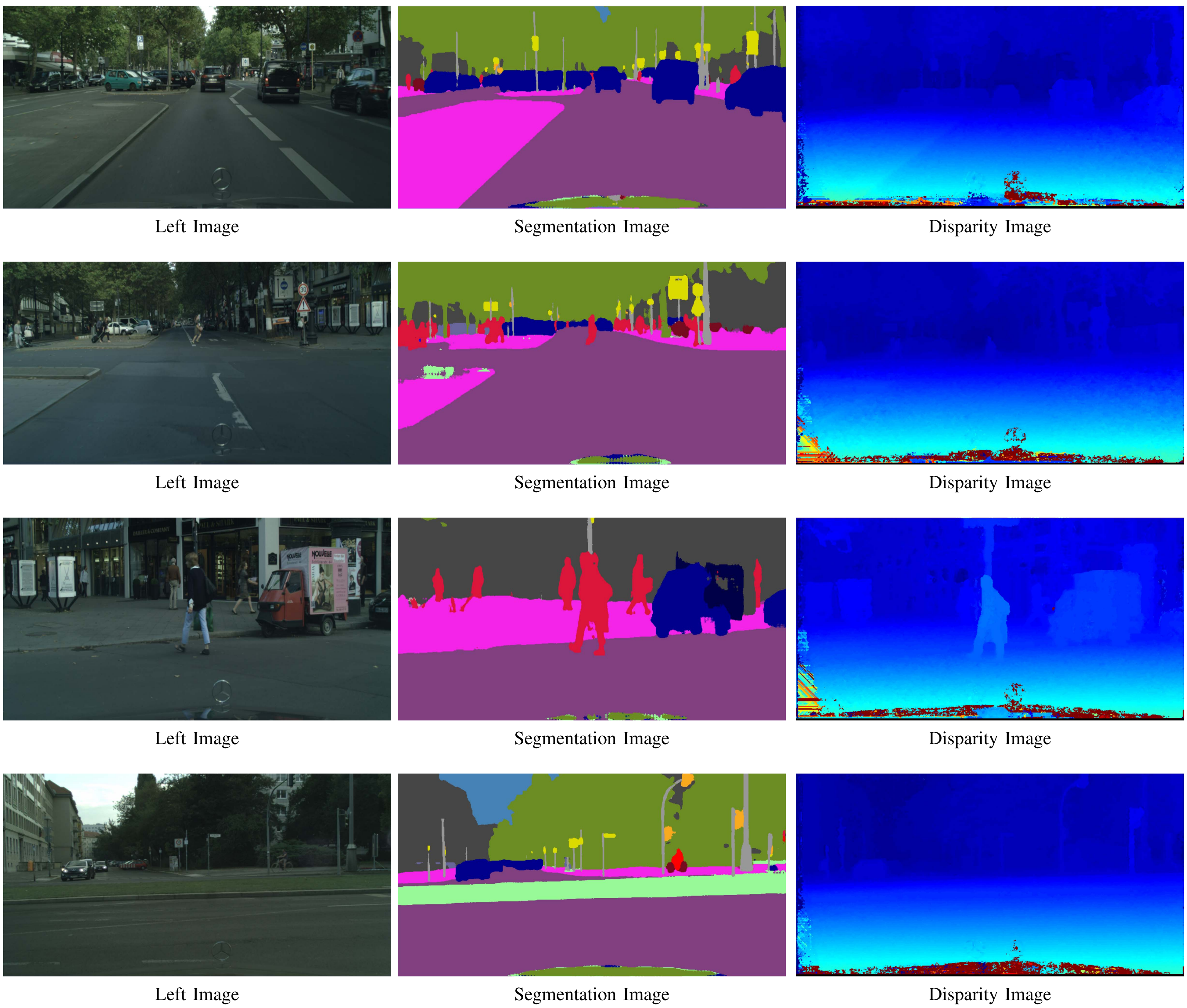


Fig. 7. Disparity maps obtained for traffic images for Cityscapes dataset.

rank our stereo reconstruction as one of the top approaches on Kitti 2015 benchmark.

E. Results on Traffic Images

We also present the results we obtained on images from Cityscapes dataset [8]. Since our method requires color traffic images (for semantic segmentation) with left and right simultaneously captured images and disparity ground truth, we cannot present numerical results in this case. However, Figure 7 shows the disparity we obtained in several traffic scenes. It can be seen that our results are both dense and accurate.

VI. CONCLUSION

We have presented here a stereo reconstruction method that gives accurate results while running in real-time on a regular GPU. The method is based on a semantic segmentation of the scene, which provides additional object information (boundaries, surfaces). Cost computation, aggregation and optimization are further improved according to segment classes. Furthermore, we introduce a novel refinement technique based on CNNs that can filter out unreliable pixels from the scene.

For now, our solution focuses on the perception of driving scenes, but could be easily extended to other scene categories by training on other datasets. We also consider to extend the usage of our edge-aware filter to other problems such as upsampling or noise filtering.

References

[1] F. Tombari, S. Mattoccia, L. D. Stefano, and E. Addimanda, "Classification and evaluation of cost aggregation methods for stereo correspondence," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.

[2] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets deep learning for car instance segmentation in urban scenes," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, Sep. 2017, pp. 1–12.

[3] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, "Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation," in *Proc. Int. Conf. Embedded Comput. Syst. (SAMOS)*, Jul. 2010, pp. 93–101.

[4] C. Banz, P. Pirsch, and H. Blume, "Evaluation of penalty functions for semi-global matching cost aggregation," in *Proc. Int. Arch. Photogram., Remote Sens. Spatial Inf. Sci. (ISPRS)*, Jul. 2012, pp. 1–6.

[5] J. T. Barron and B. Poole, "The fast bilateral solver," in *Proc. ECCV*, Oct. 2016, pp. 617–632.

[6] S. Chetlur *et al.*, "cuDNN: Efficient primitives for deep learning," 2014, *arXiv:1410.0759*. [Online]. Available: http://arxiv.org/abs/1410.0759

[7] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *Proc. BigLearn, NIPS Workshop*, 2011, pp. 1–6.

[8] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.

[9] A. D. Costea and S. Nedevschi, "Fast traffic scene segmentation using multi-range features from multi-resolution filtered and spatial context channels," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 328–334.

[10] M. Derome, A. Plyer, M. Sanfourche, and G. Le Besnerais, "A prediction-correction approach for real-time optical flow computation using stereo," in *Proc. German Conf. Pattern Recognit.*, Cham, Sutherland: Springer, Sep. 2016, pp. 365–376. doi: 10.1007/978-3-319-45886-1_30.

[11] N. Einecke and J. Eggert, "A two-stage correlation method for stereoscopic depth estimation," in *Proc. Int. Conf. Digit. Image Comput., Techn. Appl.*, Dec. 2011, pp. 227–234.

[12] N. Einecke and J. Eggert, "Stereo image warping for improved depth estimation of road surfaces," in *Proc. Intell. Vehicles Symp.*, Jun. 2013, pp. 189–194. [Online]. Available: http://dblp.uni-trier.de/db/conf/ivs/ivs2013.html#EineckeE13

[13] W. S. Fife and J. K. Archibald, "Improved census transforms for resource-optimized stereo vision," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 1, pp. 60–73, Jan. 2013.

[14] F. Güney and A. Geiger, "Displets: Resolving stereo ambiguities using object knowledge," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4165–4175.

[15] I. Haller and S. Nedevschi, "GPU optimization of the SGM stereo algorithm," in *Proc. IEEE 6th Int. Conf. Intell. Comput. Commun. Process.*, Aug. 2010, pp. 197–202.

[16] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proc. Eur. Conf. Comput. Vis.*, Berlin, Germany: Springer, 2010, pp. 1–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1886063.1886065

[17] D. Hernandez-Juarez, A. Chacón, A. Espinosa, D. Vázquez, J. C. Moure, and A. M. López, "Embedded real-time stereo estimation via semi-global matching on the GPU," *Procedia Comput. Sci.*, vol. 80, pp. 143–153, Dec. 2016.

[18] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proc. CVPR*, 2007, pp. 1–8.

[19] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[20] T. Huang, G. Yang, and G. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 27, no. 1, pp. 13–18, Feb. 1979.

[21] M. Humenberger, T. Engelke, and W. Kubinger, "A census-based stereo vision algorithm using modified Semi-Global Matching and plane fitting to improve matching quality," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2010, pp. 77–84.

[22] C. G. Keller, M. Enzweiler, M. Rohrbach, D. F. Llorca, C. Schnorr, and D. M. Gavrila, "The benefits of dense stereo for pedestrian detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 1096–1106, Dec. 2011.

[23] A. Kendall *et al.*, "End-to-end learning of geometry and context for deep stereo regression," 2017, *arXiv:1703.04309*, [Online]. Available: http://arxiv.org/abs/1703.04309

[24] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, Vancouver, BC, Canada, vol. 2, 2001, pp. 508–515. doi: 10.1109/ICCV.2001.937668.

[25] A. Kuzmin, D. Mikushin, and V. S. Lempitsky, "End-to-end learning of cost-volume aggregation for real-time dense stereo," 2016, *arXiv:1611.05689*. [Online]. Available: http://arxiv.org/abs/1611.05689

[26] L. Li, B. Qian, J. Lian, W. Zheng, and Y. Zhou, "Traffic scene segmentation based on RGB-D image and deep learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1664–1669, May 2018.

[27] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep joint image filtering," in *Proc. Eur. Conf. Comput. Vis.*, Cham, Sutherland: Springer, Oct. 2016, pp. 154–169. doi: 10.1007/978-3-319-46493-0_10.

[28] M. Loghman and J. Kim, "SGM-based dense disparity estimation using adaptive census transform," in *Proc. Int. Conf. Connected Vehicles Expo (ICCVE)*, Dec. 2013, pp. 592–597.

[29] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Boston, MA, USA, Jun. 2015, pp. 3431–3440.

[30] N. Mayer *et al.*, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4040–4048.

[31] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang, "On building an accurate stereo matching system on graphics hardware," in *Proc. ICCV*, Nov. 2011, pp. 467–474.

[32] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3061–3070.

[33] V.-C. Miclea and S. Nedevschi, "Optimizing census-based semi global matching by genetic algorithms," in *Proc. IEEE 12th Int. Conf. Intell. Comput. Commun. Process. (ICCP)*, Sep. 2016, pp. 193–198.

[34] V.-C. Miclea and S. Nedevschi, "Semantic segmentation-based stereo reconstruction with statistically improved long range accuracy," in *Proc. Intell. Vehicles Symp.*, Jun. 2017, pp. 1795–1802.

[35] V. D. Nguyen, H. Van Nguyen, and J. W. Jeon, "Robust stereo data cost with a learning strategy," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 2, pp. 248–258, Feb. 2017.

[36] S. Paris and F. Durand, *A Fast Approximation Bilateral Filter Using a Signal Process. Approach*. Berlin, Germany: Springer, 2006, pp. 568–580. [Online]. Available: https://doi.org/10.1007/11744085_44

[37] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*. [Online]. Available: http://arxiv.org/abs/1606.02147

[38] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 263–272, Jan. 2018.

[39] D. Scharstein *et al.*, "High-resolution stereo datasets with subpixel-accurate ground truth," in *Proc. German Conf. Pattern Recognit.*, Sep. 2014, pp. 31–42.

[40] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, nos. 1–3, pp. 7–42, Apr. 2002. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=64200

[41] L. Schneider *et al.*, "Semantic Stixels: Depth is not enough," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 110–117.

[42] A. Seki and M. Pollefeys, "SGM-Nets: Semi-global matching with neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6640–6649.

[43] R. Spangenberg, T. Langner, S. Adfeldt, and R. Rojas, "Large scale semi-global matching on the CPU," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2014, pp. 195–201.

[44] 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.

[45] E. Tola, V. Lepetit, and P. Fua, "DAISY: An efficient dense descriptor applied to wide-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 815–830, May 2010.

[46] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. 6th Int. Conf. Comput. Vis.*, Bombay, India, Jan. 1998, pp. 839–846. [Online]. Available: http://dl.acm.org/citation.cfm?id=938978.939190

[47] W. van der Mark and D. M. Gavrila, "Real-time dense stereo for intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 38–50, Mar. 2006.

[48] L. Xu, J. Ren, Q. Yan, R. Liao, and J. Jia, "Deep edge-aware filters," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, Jun. 2015, pp. 1669–1678. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045296

[49] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient joint segmentation, occlusion labeling, stereo and flow estimation," in *Computer Vision—ECCV* (Lecture Notes in Computer Science), vol. 8693, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014.

[50] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proc. Eur. Conf. Comput. Vis.* in Lecture Notes in Computer Science, J.-O. Eklundh, Ed. Berlin Heidelberg, Springer, 1994, vol. 801, pp. 151–158.

[51] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1592–1599.

[52] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," 2016, arXiv:1612.01105 [Online]. Available: http://arxiv.org/abs/1612.01105

[53] Y. Zhong, Y. Dai, and H. Li, "Self-supervised learning for stereo matching with self-improving ability," 2017, *arXiv:1709.00930*. [Online]. Available: https://arxiv.org/abs/1709.00930

[54] C. Zhou, H. Zhang, X. Shen, and J. Jia, "Unsupervised learning of stereo matching," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1576–1584.

Vlad-Cristian Miclea received the M.S. degree from the Ecole Normale Superieure de Lyon (ENS), Lyon, France. He is currently pursuing the Ph.D. degree with the Faculty of Automation and Computer Science, Technical University of Cluj-Napoca (TUCN), Cluj-Napoca, Romania. His research interests include image processing, stereo vision, and machine learning.

Sergiu Nedevschi (M'99) received the M.S. and Ph.D. degrees in electrical engineering from the Technical University of Cluj-Napoca (TUCN), Cluj-Napoca, Romania, in 1975 and 1993, respectively. From 1976 to 1983, he was a Researcher with the Research Institute for Computer Technologies, Cluj-Napoca. In 1998, he was appointed as a Professor of computer science and the Founder of the Image Processing and Pattern Recognition Group, TUCN. From 2000 to 2004, he was the Head of the Computer Science Department, TUCN. From 2004 to 2012, he was the Dean of the Faculty of Automation and Computer Science. He is currently the Vice-Rector of TUCN. He has published more than 200 scientific papers and has edited over ten volumes, including books and conference proceedings. His research interests include image processing, pattern recognition, computer vision, intelligent vehicles, signal processing, and computer architecture.