ISSN 2007-9737

Single-Stage Refinement CNN for Depth Estimation in Monocular Images

José E. Valdez Rodríguez, Hiram Calvo, Edgardo M. Felipe Riverón

Instituto Politécnico Nacional,
Centro de Investigación en Computación,
Mexico

jvaldezr1000@alumno.ipn.mx, hcalvo@cic.ipn.mx, edgardo@cic.ipn.mx

Abstract. Depth reconstruction from single monocular images has been a challenging task due to the complexity and the quantity of depth cues that images have. Convolutional Neural Networks (CNN) have been successfully used to reconstruct depth of general object scenes; however, proposed works use several stages of training which make this process more complex and time consuming. As we aim to build a computational efficient model, we focus on single-stage training CNN. In this paper, we propose five different models for solving this task, ranging from a simple convolutional network, to one with residual, convolutional, refinement and upsampling layers. We compare our models with the current state of the art in depth reconstruction and measure depth reconstruction error for different datasets (KITTI, NYU), obtaining improvements in both global and local error measures.

Keywords. Depth reconstruction, convolutional neural networks, single stage training, embedded refinement layer, stereo matching.

1 Introduction

Reconstructing depth from a single monocular image (opposed to stereoscopic images) is a challenging task due to the required ability to detect depth cues such as shadows, perspective, motion blur, etc. [14]. Previous works in depth reconstruction have tackled the general problem of estimating the component of depth in a wide range of images, mainly focusing on objects against a solid or a complex background [20, 2, 4, 23, 19]; other works focus on city buildings and people, paying special attention to the problem of reconstructing depth for images where perspective plays a predominant role [25].

In general, most successful models in the state of the art—v. gr. [6, 2]—propose multi-stage networks that require separate training for each one of their stages, making the process of training them complex and time consuming. In this paper, we propose five Convolutional Neural Network models capable of reconstructing depth from a single image that require only one training stage. The models proposed in this work are capable of reconstructing depth both in a global and a local view [14]. We test our models with both objects and city roads with perspective.

This paper is organized as follows: in Section 2 we describe related work to this research; Section 3 describes the proposed method; Section 4 shows the results obtained from our proposal and a comparison with the state of the art, and finally in Section 5 the conclusions of this work are drawn.

2 Related Work

Depth reconstruction from a single image has been tackled by different methods. Although using Convolutional Neural Networks (CNN) has recently become one of the best techniques, using CNN to solve the problem of depth reconstruction can be still considered at as a development stage due to the complexity of the design of these networks. One of the first works to use this technique can be found in [7].

They propose using two CNN: the first one estimates depth at global view, and the second one refines the local view. They establish a CNN architecture to be applied to depth estimation and propose a loss function.

In [6] using three CNN is proposed, one for each stage to estimate depth of a single image. The first network estimates depth at a global view; the second network tries to estimate depth at half the resolution of the input image, and a third one refines or estimates depth at a local level. They propose a new loss function. In [20] a single CNN is presented that uses a pre-processed input based on superpixels. They complement their work using a probabilistic method to improve their results. Then, from the same group [21] presents a new architecture of their network maintaining the stage of improvement of the image. Finally, a single CNN in regression mode is presented in [2] to estimate depth with a different loss function.

In general, the discussed architectures are similar, being the main changes between them the loss function, activation functions, number of filters per layer, and size of the filter. Different databases have been used to train and test their neural networks, making it difficult to directly compare their performances. Despite of this, one of the works that can be considered to achieve the best results in the state of the art is [2]. Additionally, their architecture is based on a single-stage CNN. They present results of reconstructing depth of chair images against different backgrounds.

In the next section we present our proposal. Based on Afifi and Hellwich's CNN, our architectures are based on a single CNN, but, among other differences, we add local refinement in the same training stage in most of our models.

3 Proposed Method

In this section we present our models based on Convolutional Neural Neworks (CNN). We have decided to name them according to the kind of layers they use, namely: *Convolutional* layers (C) [16]; *reSidual blocks* (S) [11]; *Upsample* layers (U) [27]; and Refinement layers (R). We also add biases to each layer. In Fig. 1 a representation of a layer in the CNN models is shown.

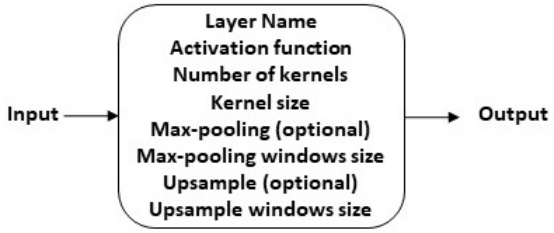

Fig. 1. Single layer representation

As mentioned before we implemented Afifi and Hellwich's CNN model to compare our results, as they perform depth estimation in a single stage, depicted in Fig. 2. They use Convolutional Blocks made by consecutive convolutional layers as shown in Fig. 3.

3.1 Basic Models – Convolutional (C) and reSidual-Convolutional (SC)

In this section we present two CNN models, the first one (C:Convolutional) is presented in Fig. 5 and the second one (SC: reSidual-Convolutional) is presented in Fig. 6. The main differences between both models are the kind of layers used in every layer, one of them only uses convolutional layers and the other one uses residual blocks and convolutional layers.

The Residual Block is shown in Fig. 4, it uses an identity map which helps to improve the results as mentioned in [11]. Both models consist of five layers, four of them use *Rectified Linear Units* (ReLU) [3] as activation function and the output layer uses sigmoid as activation function to obtain values between 0 and 1. In both models the first two layers use max-pooling in order to reconstruct depth at different spatial resolutions of the input image. In every layer of the models the kernel size is 3x3 and subsequently it changes in each layer. Both models reconstruct depth at both global and local.

3.2 Models with Refinement

Aiming to include refinement in a single training stage, we developed three additional models.

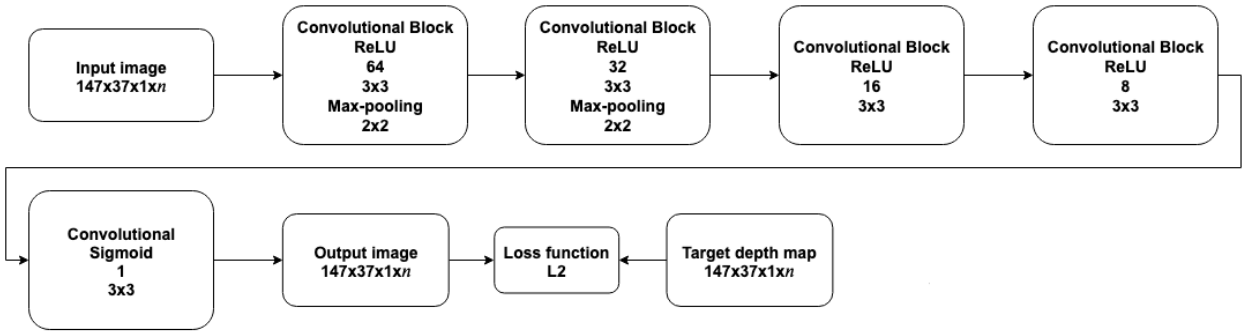

Fig. 2. CNN model by Afifi and Hellwich [2]

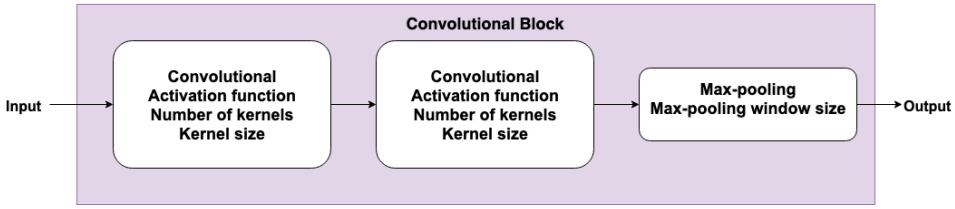

Fig. 3. Convolutional Block [2]

The main difference between them is the upsample operation and the refinement method. The refining methods used in the CNN models are based on Xu et al. [26] and Dosovitskiy et al. [5], both refinement methods use convolutional layers to extract features directly from the input images at local level, the main difference between them is the number of convolutional layers and how it is included on the full CNN model, also Xu et al. performs refinement before the upsampling stage and Dosovitskiy performs refinement after the upsampling stage.

3.2.1 reSidual-Convolutional-Refinement (SCRX) Model

The block diagram of this model is presented in Fig. 7. The input image is the left image of the dataset and the target image is the depth map obtained from the stereo matching algorithm. In Region A we can see the Reduction operation which tries to reconstruct depth at global view; it consists of four Residual Blocks with kernel size of 3×3 and the activation function ReLU; the residual block is used because we tried to avoid weights with zero value.

We use Max-pooling only on the first two Residual Blocks to reduce image resolution and reconstruct depth at different image sizes. At the end of Region A we use a convolutional layer with a sigmoid activation function to limit output limits beetween 0 and 1. Region B tries to estimate depth at local view, joining the output of Region A and two convolutional layers with Max-pooling. The refining method in this model was taken from the work of Xu et al. [26]—hence the X in SCRX. The output of the model is given by the convolutional layer with kernel size 3×3 and a sigmoid activation function in Region B.

3.2.2 reSidual-Convolutional-Refinement-Upsampling Models (SCRXU and SCRDU)

These models consist of an additional upsample layer. Two different refining methods were used in each model: the refining method in the first model

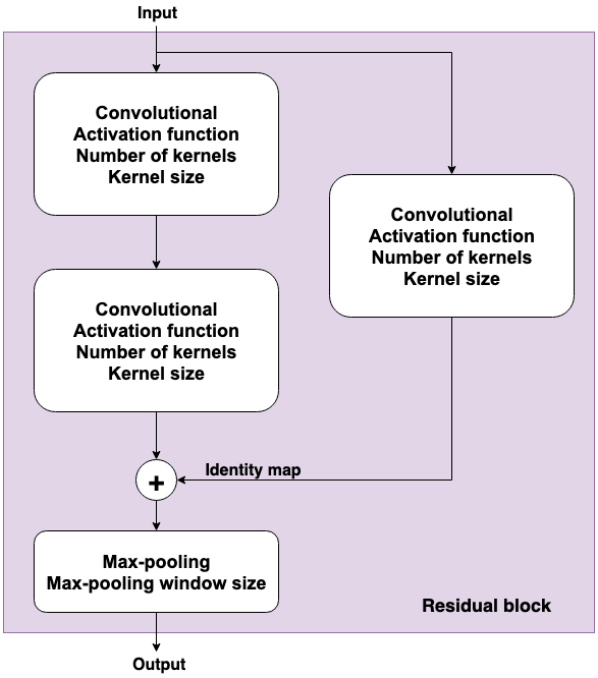

Fig. 4. Residual Block [11], implemented for our CNN models

(SCRXU) is based on [26] and for the second one (SCRDU) is based on [5].

The block diagrams of these models are presented in Fig. 8 and Fig. 9. The input image is the left image of the dataset [8] and the target image is the depth map obtained from the stereo matching algorithm. In Region A in both models, the Reduction operation is performed. These layers aim to reconstruct depth at global view; they are composed of four Residual Blocks with kernel size of 3×3 and ReLU activation function. The residual block is used because we are trying to avoid null weights. We use Max-pooling only on the first two Residual Blocks to reduce image resolution and reconstruct depth at different image sizes.

At the end of Region A, we use a convolutional layer with a sigmoid activation function to limit output limits between 0 and 1. Region B in both models performs the upsample operation by using two Residual Blocks and a Convolutional layer, the kernel size in this Region is 3×3 in every layer. Finally, Region C in both models is the refinement stage, which tries to estimate depth at local view, consisting of two convolutional layers with Max-pooling and ReLU activation function, and a convolutional layer with sigmoid activation function.

As the output of the CNN is smaller than the input image, due to the CNN operations, to recover the original size of the input image, we use Bilinear Interpolation [10], after the last convolutional layer. It is to mention that we describe our models as multi-stage models, but we considered them single-stage models because we only trained them once, we do not train each stage separately as [7] and [6], also we do not apply pre-processing to the input images. The output of all the CNN models are normalized between values of 0 and 1, to optimize the learning of them according to [12], then the images can be converted to grayscale values.

3.3 Loss Function

To train our CNN model we have to minimize an objective loss function. This loss function measures the global error between the target

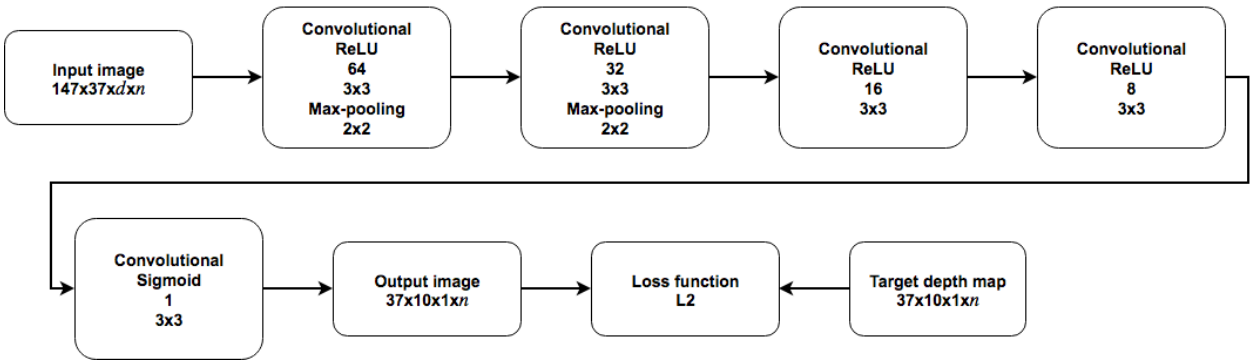

Fig. 5. Convolutional (C) Model

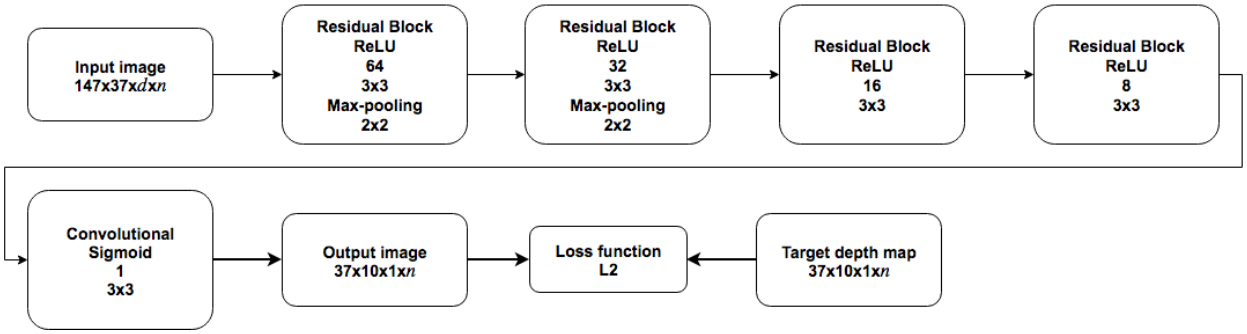

Fig. 6. reSidual-Convolutional (SC) Model

image and the depth reconstruction given by the CNN.

We use the $L2$ norm as the loss function, and it is used by both models. The $L2$ norm can be calculated as shown in eq. 1.

$$L2 = \frac{1}{2n}\sum_{i=1}^{n} \|y(i) - y'(i)\|_2^2, \tag{1}$$

where:

y' = Reconstructed depth map.

y = Target depth map.

n = Number of images per batch.

3.4 Error Measures

To evaluate our models we use several error measures commonly used to compare depth reconstruction with the original targets [20, 2].

Root Mean Square Error (RMSE):

$$\sqrt{\frac{1}{|T|}\sum_{y'\epsilon|T|}(y-y')^2}. \tag{2}$$

Mean Squared Error (MSE):

$$\frac{1}{|T|}\sum_{y'\epsilon|T|}(y-y')^2. \tag{3}$$

Absolute Relative Difference (ARD):

$$\frac{1}{|T|}\sum_{y'\epsilon|T|}\frac{|y-y'|}{y'}. \tag{4}$$

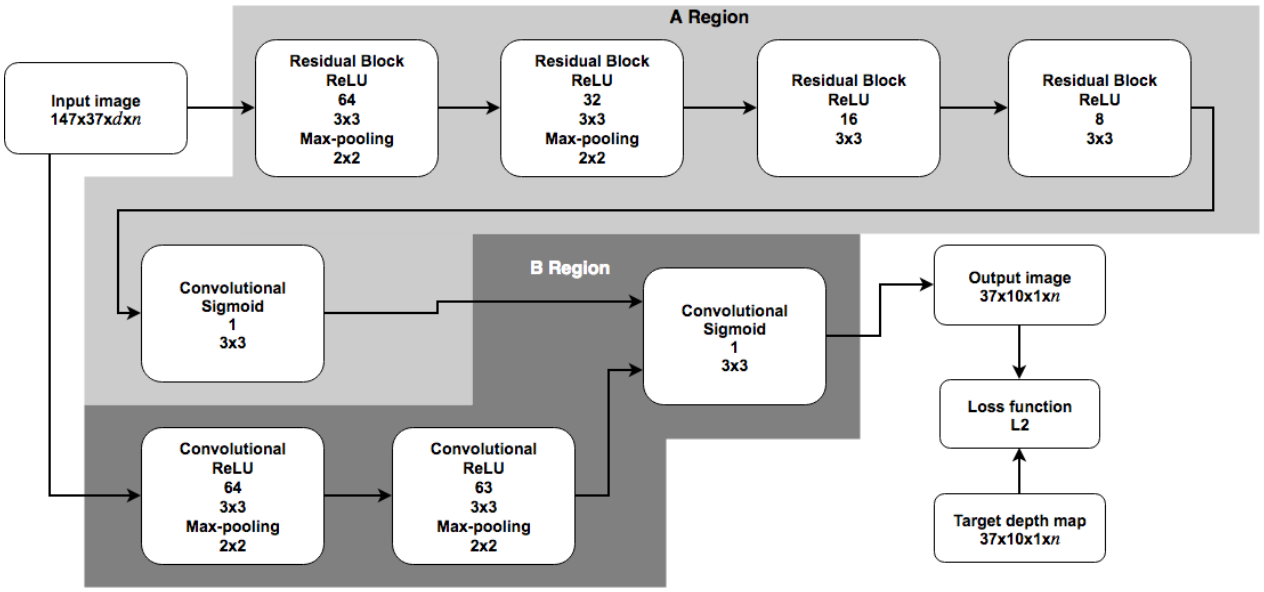

Fig. 7. reSidual-Convolutional-Refinement (SCRX) Model

Logarithmic Root Mean Square Error (LRMSE):

$$\sqrt{\frac{1}{|T|}\sum_{y'\epsilon|T|}(log(y)-log(y'))^2}. \quad (5)$$

Logarithmic Root Mean Square Error Scale-Invariant (LRMSE-SI):

$$\frac{1}{|T|}\sum_{y'\epsilon|T|}(log(y)-log(y'))^2. \quad (6)$$

Squared Relative Difference (SRD):

$$\frac{1}{|T|}\sum_{y'\epsilon|T|}\frac{\|y-y'\|^2}{y'}. \quad (7)$$

where:

y' = Reconstructed depth map.

y = Target depth map.

T = Number of pixels on the images.

RMSE, MSE and ARD tend to represent the global view error in the images, while and LRMSE, LRMSE-SI and SRD are more representative of the local view error.

4 Experiments and Results

In this section we describe the results of our models. We experimented with two different datasets: KITTY and NYU.

4.1 KITTI Dataset

We used the object tracking section of *The KITTI Vision Benchmark Suite* [8]. This dataset contains 15,000 outdoor scenes given as stereoscopic images. This dataset does not directly contain the target depth maps, so we had to reconstruct the target depth maps using different algorithms such as *Semiglobal stereo matching* [13] and *Blockmatching stereo algorithm* [15]. In Fig. 10 an image sample to compare different stereo matching algorithms to obtain the target depth map is shown. Finally, we decided to use the *Efficient large-scale stereo matching* [9] due to its quality and its evaluation reported in [22]. This algorithm receives a pair of stereo images and its result is the target depth map (see Fig. 11).

We trained our models with Backpropagation [18] and Stochastic Gradient Descent [17]. We used 1,000 iterations and a batch size of 40. From

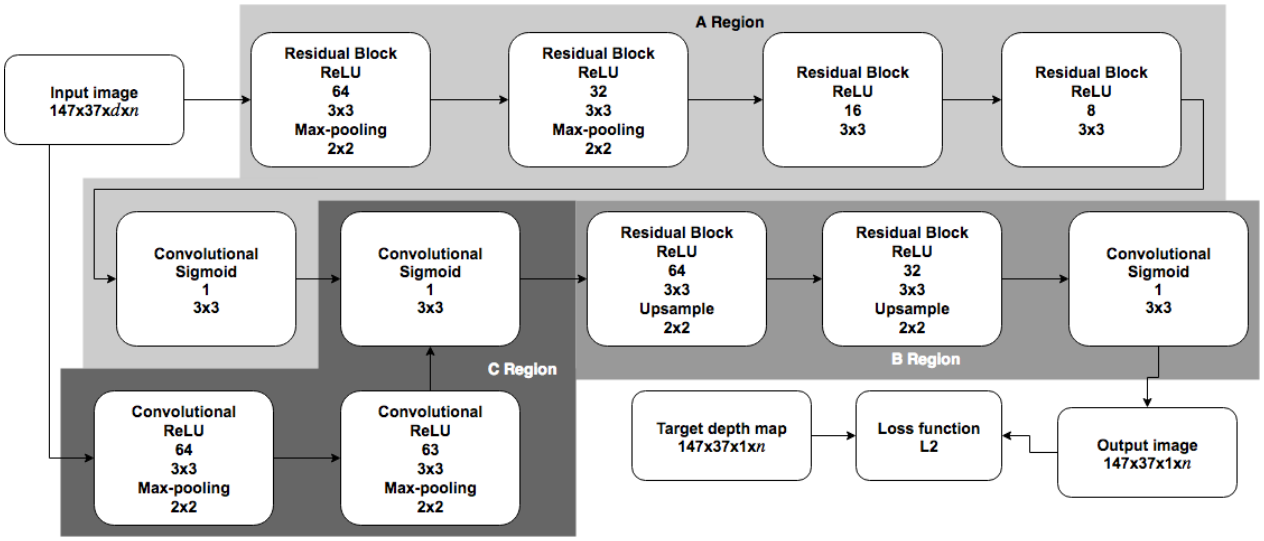

Fig. 8. reSidual-Convolutional-Refinement (Xu)-Upsampling (SCRXU) Model

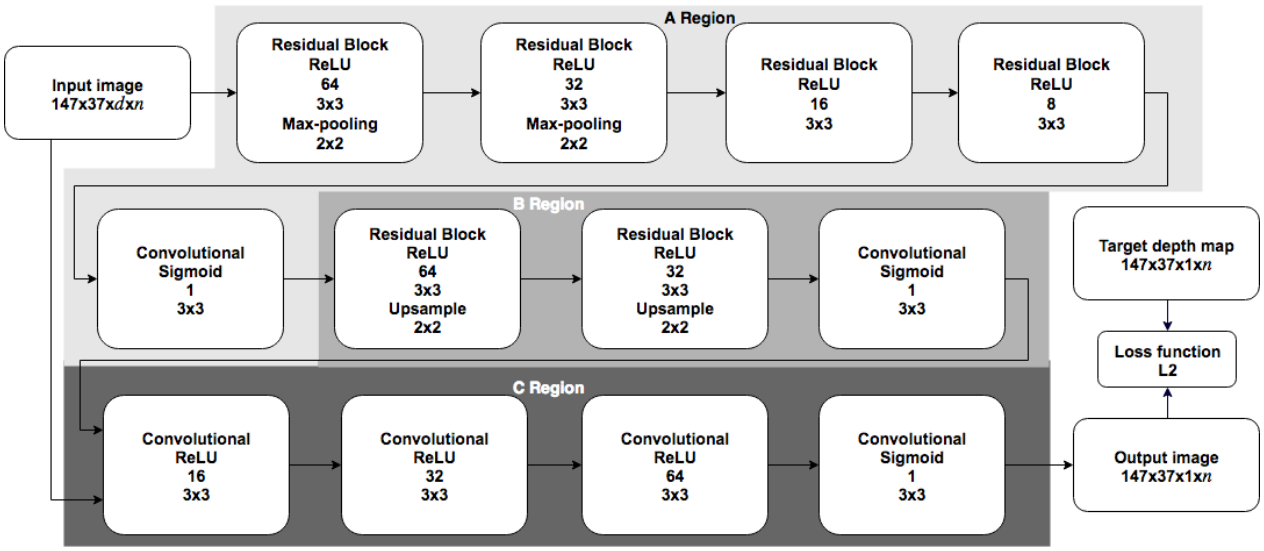

Fig. 9. reSidual-Convolutional-Refinement (Dosovitskiy)-Upsampling (SCRDU) Model

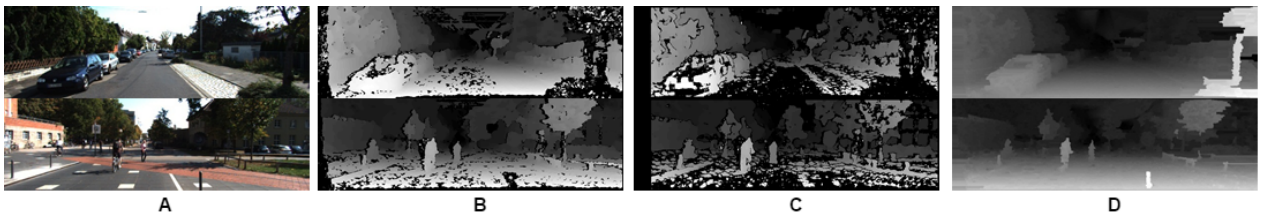

Fig. 10. Target depth map comparison with different stereo matching methods, (A) Stereoscopic images, (B) Semiglobal Matching, (C) Blockmatching method, (D) Large-Scale matching method

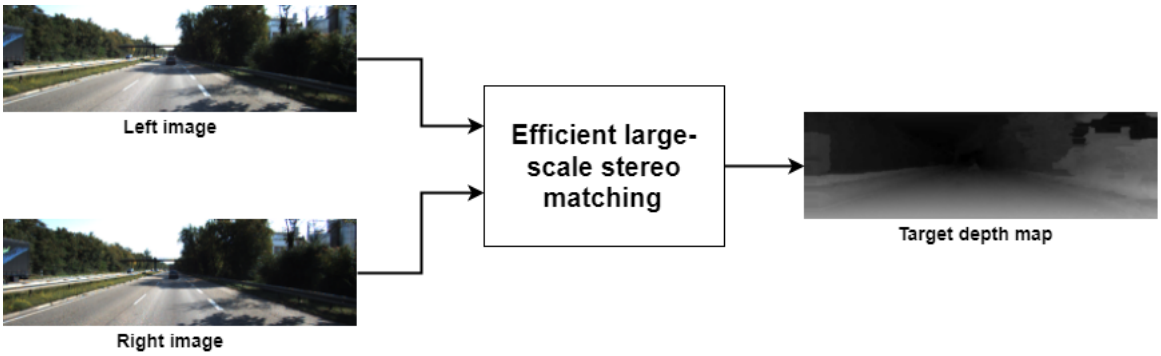

Fig. 11. Target depth map creation

the dataset we used 12,482 images (left *eye*) and its respective target depth map for training, and 2,518 images for testing. Both input and output images have a size of 147×37 pixels, giving $T = 5439$, d is the number of input channels; for grayscale $d = 1$ and $n = 40$ due to the batch size. All images were converted to grayscale before training and testing depth estimation.

We implemented our models with the Python toolbox, *Tensorflow* [1] which can be trained on a GPU for swift performance. We trained our models on a GPU NVIDIA GTX 1080; it took approximately two days for training and less than a second for testing a single image. As explained in Section 2, for comparison we implemented the model proposed by [2].

This model can be trained with the $L2$ norm and still obtain better results than the rest of related work. Figure 12 shows a sample of results of our implementation AH of [2], and our proposed models C, SC, SCRX, SCRXU and SCRDU.

Qualitatively comparing results shown in previous figures, it can be seen that our models are capable of reconstructing depth at global view, with slightly better attention to details in local view than previous proposals. For global view we refer to the image in general avoiding small details and for local view we refer to small details such as cars, pedestrians, etc. For a quantitative analysis, Table 1 presents the error measures of all our models and the reimplemented model. RMSE, MSE and ARD measure the global view error in the images, while and LRMSE, LRMSE-SI and SRD measure the local view error.

Comparing our methods with the state of the art, we obtain better results with most of our models, being SCRX Model better at global view while SC Model performs better at local view.

4.2 Improved SCRX Model

While previous models were tested with images of 147×37 pixels, we selected the model with best performance (SCRX) and modified it to process RGB inputs (instead of grayscale) resulting in $d = 3$ and larger images. We tested with inputs of 310×94 and compared results. A sample output is shown in Figure 13. Quantitative measure errors are listed in Table 2. Comparing to Table 1, it can be seen that using larger RGB inputs allows the network to better estimate depth.

4.3 Tests on the NYU Dataset

The NYU v2 dataset [24] consists of 1,449 images of indoor scenes, from which 795 are used for training and 654 for test (the standard training-test split provided with the dataset). Images were resized to 320×240 pixels prior to training. We performed tests on this dataset with the improved SCRX model. Figure 14 shows a sample output with this dataset. Both grayscale and color inputs were tested.

As with the previous dataset (KITTI), better results were achieved by using color images (See Table 3). Quantitatively, our model performs better than the prior work as shown in Table 4, the main difference between the results of our model is that

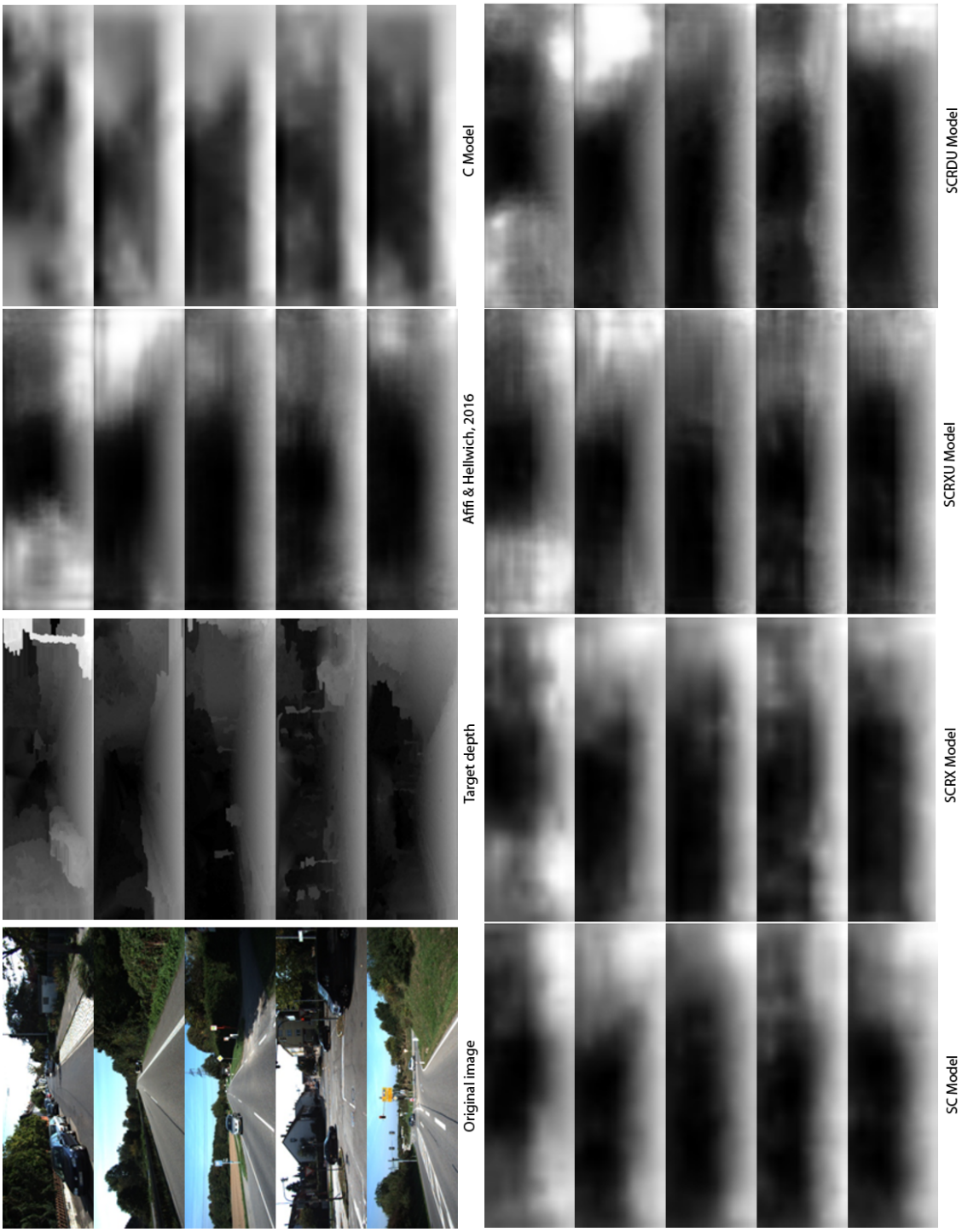

Fig. 12. Sample output on KITTI Dataset for models [2], C, SC, SCRX, SCRXU and SCRDU

Table 1. Comparison of error measures between our approach and AH [2] (smaller is better)

Error	AH	C	SC	SCRX	SCRXU	SCRDU
RMSE	0.1496	0.1301	0.1082	**0.1051**	0.1218	0.1348
MSE	0.0260	0.0193	0.0132	**0.0124**	0.0166	0.0213
ARD	0.4540	0.5858	0.4813	0.4625	0.4661	**0.4350**
LRMSE	0.3377	0.3618	**0.3048**	0.3081	0.3239	0.3260
LRMSE-SI	0.2068	0.2269	**0.1816**	0.1820	0.1980	0.1931
SRD	0.9865	1.1982	0.8678	**0.8016**	0.9528	0.9303

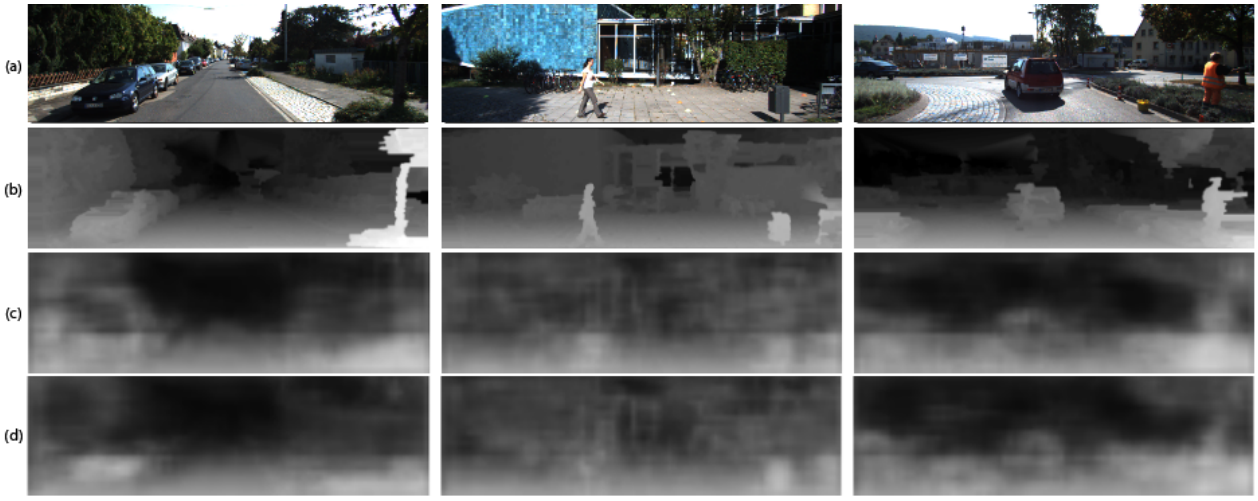

Fig. 13. Sample output of improved SCRX Model. (a) Input image; (b) Target depth map; (c) Output for grayscale input; (d) Output for color input

Table 2. Results of improved SCRX model on the KITTI dataset

	Grayscale	Color
RMSE	0.0964	0.0881
MSE	0.0110	0.0091
ARD	0.3539	0.3434
LRMSE	0.2814	0.2726
LRMSE-SI	0.1588	0.1494
SRD	0.5371	0.5326

Table 3. Results of improved SCRX model on the NYU dataset

	Grayscale	Color
RMSE	0.2138	0.2117
MSE	0.0478	0.0473
ARD	0.0334	0.0205
LRMSE	0.3181	0.2915
LRMSE-SI	0.2376	0.2225
SRD	1.3493	0.7574

we use all the pixels from the target depth map of the dataset, avoiding the cases of invalid pixels.

5 Conclusions and Future work

We have presented five CNN models that require only a single-stage training and perform refinement in the same stage. We tested our models with an existing dataset of stereoscopic images and compared their performance with the state of the art in depth reconstruction.

We found that the use of a residual block instead of convolutional layers improves results.

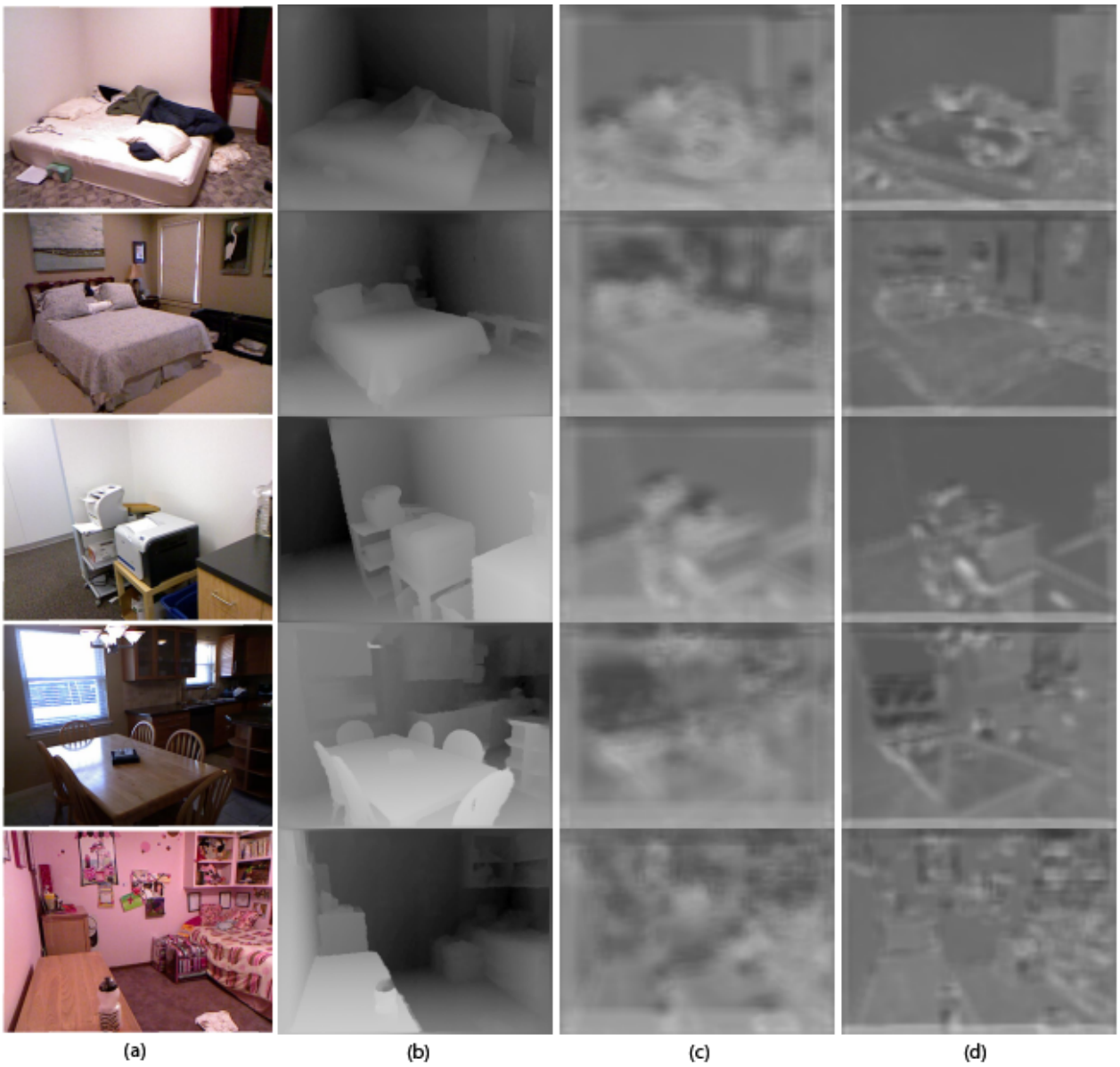

Fig. 14. Sample output of the improved SCRX Model on the NYU dataset. (a) Input image; (b) Target depth map; (c) Output for grayscale input; (d) Output for color input

Using upsample layers improves quantitative results as well, although this may not be directly attested in some cases when examining results qualitatively – some objects appear to be blurred and less defined because of this layer. Using bilinear interpolation instead of upsampling layers improves results. Perspective of the image takes an important role on the reconstruction of depth. Quantitatively, local depth can be estimated with our models, but there is still room for improvement. As a future work we plan to experiment with different kernel sizes, different loss functions

Table 4. Comparison between the state of the art and our proposal on the NYU dataset (smaller is better)

Authors		RMSE	ARD	LRMSE
Ours	Grayscale	0.2138	0.0334	0.3181
	Color	**0.2117**	**0.0205**	0.2915
Liu et al., 2014	DC-CRF	1.0600	0.3350	0.1270
Eigen et al., 2014		0.8770	0.2140	0.2830
Eigen et al., 2015		0.6410	0.1580	0.2140
Baig et al., 2016	GCL	0.8156	0.2523	0.0973
	RCL	0.8025	0.2415	**0.0960**
Mousavian et al., 2016		0.8160	0.2000	0.3140
Lee et al., 2018		0.5720	0.1320	0.1930

and activation functions to further improve our proposed models.

Acknowledgements

The authors wish to thank support by CONACyT-SNI and Instituto Politécnico Nacional (IPN), particularly through grants SIP20195886, SIP20190140, EDI, and COFAA-SIBE.

References

1. **Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015).** TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
2. **Afifi, A. J. & Hellwich, O. (2016).** Object depth estimation from a single image using fully convolutional neural network. *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on*, IEEE, pp. 1–7.
3. **Arora, R., Basu, A., Mianjy, P., & Mukherjee, A. (2016).** Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*.
4. **Baig, M. H. & Torresani, L. (2016).** Coupled depth learning. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, pp. 1–10.
5. **Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., & Brox, T. (2015).** Flownet: Learning optical flow with convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2758–2766.
6. **Eigen, D. & Fergus, R. (2015).** Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2650–2658.
7. **Eigen, D., Puhrsch, C., & Fergus, R. (2014).** Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, pp. 2366–2374.
8. **Geiger, A., Lenz, P., & Urtasun, R. (2012).** Are we ready for autonomous driving? the kitti vision benchmark suite. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
9. **Geiger, A., Roser, M., & Urtasun, R. (2010).** Efficient large-scale stereo matching. *Asian conference on computer vision*, Springer, pp. 25–38.
10. **Gonzalez, W. & Woods, R. E. (2004).** Eddins, digital image processing using MATLAB. *Third New Jersey: Prentice Hall*.
11. **He, K., Zhang, X., Ren, S., & Sun, J. (2016).** Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.
12. **Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R.**

(2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

13. **Hirschmuller, H. (2008).** Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, Vol. 30, No. 2, pp. 328–341.
14. **Howard, I. P. (2012).** *Perceiving in depth, volume 1: basic mechanisms*. Oxford University Press.
15. **Konolige, K. (1998).** Small vision systems: Hardware and implementation. In *Robotics research*. Springer, pp. 203–212.
16. **LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., & Jackel, L. D. (1990).** Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, pp. 396–404.
17. **LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998).** Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324.
18. **LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K.-R. (2012).** Efficient backprop. In *Neural networks: Tricks of the trade*. Springer, pp. 9–48.
19. **Lee, J.-H., Heo, M., Kim, K.-R., & Kim, C.-S. (2018).** Single-image depth estimation based on fourier domain analysis. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 330–339.
20. **Liu, F., Shen, C., & Lin, G. (2015).** Deep convolutional neural fields for depth estimation from a single image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5162–5170.
21. **Liu, F., Shen, C., Lin, G., & Reid, I. (2016).** Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 38, No. 10, pp. 2024–2039.
22. **Menze, M. & Geiger, A. (2015).** Object scene flow for autonomous vehicles. *Conference on Computer Vision and Pattern Recognition (CVPR)*.
23. **Mousavian, A., Pirsiavash, H., & Košecká, J. (2016).** Joint semantic segmentation and depth estimation with deep convolutional networks. *3D Vision (3DV), 2016 Fourth International Conference on*, IEEE, pp. 611–619.
24. **Nathan Silberman, P. K., Derek Hoiem & Fergus, R. (2012).** Indoor segmentation and support inference from rgbd images. *ECCV*.
25. **Saxena, A., Chung, S. H., & Ng, A. Y. (2006).** Learning depth from single monocular images. *Advances in neural information processing systems*, pp. 1161–1168.
26. **Xu, N., Price, B., Cohen, S., & Huang, T. (2017).** Deep image matting. *arXiv preprint arXiv:1703.03872*.
27. **Zeiler, M. D. & Fergus, R. (2014).** Visualizing and understanding convolutional networks. *European conference on computer vision*, Springer, pp. 818–833.

Article received on 29/10/2019; accepted on 06/03/2020.
Corresponding author is Hiram Calvo.