Stereo Matching through Squeeze Deep Neural Networks

Gabriel D. Caffaratti[1,2], Martin G. Marchetta[1], and Raymundo Q. Forradellas[1]

[1] Laboratorio de Sistemas Inteligentes (LABSIN)
Facultad de Ingeniería - Universidad Nacional de Cuyo
Centro Universitario, Mendoza, Argentina
gabriel.caffaratti@ingenieria.uncuyo.edu.ar
martin.marchetta@ingenieria.uncuyo.edu.ar
kike@uncu.edu.ar
[2] CONICET

Abstract. Visual depth recognition through Stereo Matching is an active field of research due to the numerous applications in robotics, autonomous driving, user interfaces, etc. Multiple techniques have been developed in the last two decades to achieve accurate disparity maps in short time. With the arrival of Deep Leaning architectures, different fields of Artificial Vision, but mainly on image recognition, have achieved a great progress due to their easier training capabilities and reduction of parameters. This type of networks brought the attention of the Stereo Matching researchers who successfully applied the same concept to generate disparity maps. Even though multiple approaches have been taken towards the minimization of the execution time and errors in the results, most of the time the number of parameters of the networks is neither taken into consideration nor optimized. Inspired on the Squeeze-Nets developed for image recognition, we developed a Stereo Matching Squeeze neural network architecture capable of providing disparity maps with a highly reduced network size without a significant impact on quality and execution time compared with state of the art architectures.

Keywords: Stereo Matching, Deep Learning, Squeeze Nets, Artificial Intelligence, Artificial Vision, Disparity Maps.

1 Introduction

Stereo Matching is a research field inspired in human capabilities, in particular the stereopsis which is the ability of gathering depth information from the pair of images retrieved by the human binocular vision. Getting this information is essential to make decisions in different applications which interact with the world, like robotic object manipulation, unmanned vehicles navigation, security systems, user interfaces, etc. Since Hannah[1] and Marr et al.[2] proposed the matching of two images to extract stereo information, a number of techniques were developed to achieve this goal. As these Stereo Matching techniques started showing similarities, Scharstein et al.[3] developed a taxonomy that precisely

defined common steps on them, specified their goals and instructed the process to measure them. Since then, most of the efforts have been focused on measuring how accurate were the disparity maps obtained in the matching cost calculation and post-processing steps, and how fast they were built.

Artificial Intelligence, and in particular Machine Learning, offer different techniques to solve complex problems through the training of different models. Within this field, a popular technique was the MultiLayer Perceptron (MLP) developed by Rumelhart et al.[4], a trainable artificial neural network capable of learning models through the adjustment of the weights that connected the different layers of neurons through the Back-Propagation algorithm. Even though these networks were widely applied after their appearance, one of their main problems was their lack of scalability as a product of the exponential growth of the number of weights when there is a big number of inputs and outputs. A different type of networks called Convolutional Neural Networks (CNN), a specific technique of Deep Learning, was also trained through the back-propagation algorithm [5]. However, CNN started to be widely adopted only when Hinton, G. [6] shown how to train Deep Network layers independently by tuning the back-propagation algorithm.

Multiple CNN architectures have been proposed due to their simple and flexible training mechanism. In addition, frameworks like Torch[7], Tensorflow[8] or Theano[9] simplified their construction, training and test. In particular, one of the benefits of using CNN appeared when they were applied in image recognition problems, outperforming all the state of the art techniques [10], and currently surpassing the human performance[11]. Due to their success in image classification problems, CNN were also applied recently by Zbontar et al.[12] for the disparity cost calculation, bringing CNN architectures to the Stereo Matching field for the first time.

Different challenges have been presented for image classification[13] and disparity map generation[14] aiming at reducing the error rate and execution time. The size of the network in terms of number of parameters is very important, because it affects the computational cost for training and execution, and also because several applications require remote updates of the trained architectures, presenting in some cases connectivity restrictions and making the size of the network an important feature to optimize. Iandola et al.[15] proposed a CNN architecture called Squeeze Nets for image recognition which decreases the number of parameters to train and store, thus reducing the size of the network significantly. Inspired on that work, we developed a squeeze-network-based model for the generation of disparity maps for Stereo Matching, which reduces the network size in storage, while also maintaining the runtime memory, execution time and quality of the results.

This paper shows a review of the state of the art techniques presented in Stereo Matching (Section 2), an explanation of our proposed architecture (Section 3), experiments performed based on a case of study (Section 4) and the conclusions and future work proposals (Section 5).

2 Literature Review

The construction of disparity maps consist in calculating the distance between various points or sections in a scene according to the position of the cameras. This task has several complexities due to the nature of the variable characteristics of the pictures and the ambiguous information they contain. Researchers have dealt with these ambiguities by making different assumptions of the images or data contained on it. The first pair of assumptions taken were uniqueness and continuity. Uniqueness establishes that each item of each image must be assigned with at most one disparity value. This condition relies on the assumption that an item correspond to something that has a unique physical position. Continuity states that disparity varies smoothly except on object boundaries where there are depth discontinuities[2]. Another important assumption is the epipolar rectification of the images which reduces the matching process to one dimension, or in other words a matching calculate over an horizontal line. Based on these assumptions stereo matching techniques were able to proceed with the matching of objects. However, we are far away from resolving all the different problems in the topic. Other important problems in stereo matching are the occlusions, textureless or repetitive texture surfaces, shape of the objects, differences in the intensities or noise in the images among others.

At side of the problems presented above, researchers found a number of mechanisms to retrieve dense disparity maps which can be divided in two groups: the ones detached of CNN and the ones based on these type of architectures.

2.1 CNN detached Stereo Matching techniques

Since the taxonomy proposed in [3] multiple techniques were proposed for the retrieval of dept information. These techniques can be categorized in two main groups based on the way the disparity map is calculated. The solutions that calculates the matching cost comparing a windows of neighbor pixels to gradually build disparity maps are considered local methods. Opposite to this, the solutions that retrieve a complete map and iteratively optimize it are considered global methods.

An extensive survey of local and global stereo matching algorithms where different comparisons and measures are made can be found in Hamzah et al.[16] work.

2.2 CNN based Stereo Matching techniques

CNN architectures marked a huge improvement in Artificial Vision areas. Particularly, this kind of artificial networks brought the attention of researchers when the Alexnet created by Krizhevsky et al.[10] reduced more than 10% the error rate on image classification problems reaching a 15.3%. Such achievement caused a revolution in this research field, having multiple CNN architectures proposed in the last five years. Zeiler et al.[17] was able to tune the hyperparameters of AlexNet creating the ZFNet and obtaining a 14.8% error rate. Later

on GoogLeNet architecture by Russakovsky et al.[18] introduced the concept of inception modules enlarging the number of layers of the net with very small convolutional layers obtaining a 6.61% error and a considerable reduction of the parameters. An impressive improvement of the accuracy was achieved by He et al.[11] proposing the addition of residual links between layers in the ResNet achieving an error rate of 3.58%. A parameter reduction approach presented by Iandola et al.[15] decreased the network parameters size by 50x obtaining the same error rate as AlexNet.

All the previously mentioned networks were applied to the recognition of images with great success, however they had potential for other Artificial Vision problems like Stereo Matching. Zbontar et al.[12] worked on this idea inspired by the popularity of CNN and Mei et al.[19] work. He proposed a CNN based cost matching architecture with a series of post-processing steps described in Mei et al. paper obtaining an error rate of 2.63% on their accurate network version. It worth to mention a better implemented architecture designed by Shaked et al.[20] combining Zbontar et al.[12] suggested network with the residual networks proposed by He et al.[11]. Our work is highly inspired on [11,12,19] papers combined with the proposed Squeeze nets proposed in [15].

2.3 Post-processing algorithms

In many occasions, the disparity maps obtained from the matching cost calculation process can be inaccurate, have occluded areas or unmatched sections. In order to improve their quality, different post-processing algorithms have been developed. An adaptive window algorithm based on the color similarity is proposed as cross-based cost aggregation by Ke Zhang et al.[21]. The method suggested by Hirschmüller[22] called semi-global matching (SGM) improves the smoothness term in the energy functions performing fast approximations of the neighbor pixels using Mutual Information in multiple directions. Another interesting post-processing method is proposed by Yao et al.[23] where the left and right based disparity maps are interpolated to get a better approximation of the mismatching areas. In addition, the disparity map results can be improved performing a quadratic interpolation of neighbor pixels as suggested by Miclea et al.[24].

3 Proposed System

Our model is composed of four steps as depicted the figure 1, following the common stereo matching taxonomy suggested by Scharstein et al.[3]. The system performs a cost calculation through a CNN extracting features of the left and right images individually and checking their similarity on a final layer, then a well known post-processing algorithm is applied in a cost aggregation module, then a disparity map is constructed based on the matching costs from the previous steps, and finally different known interpolation and image refinement algorithms are executed to obtain an optimized dense disparity map. Our contribution can be found in the cost calculation module where a modified Squeeze Net architecture[15] is used to build a raw disparity map.

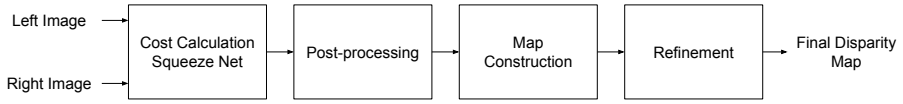

Fig. 1. Four step proposed Stereo Matching system

3.1 Cost Calculation module

The cost calculation of a common stereo matching algorithm is basically the comparison of pixels from different images. We implemented this functionality combining a CNN architecture and refinement algorithms as described below.

Network Architecture: The proposed deep network architecture for cost calculation first processes the pair of images separately, executing two passes on the same layers, and then joins the results in a final layer, as depicted in the figure 2. The cost calculation network provides a raw disparity map, in the form of a 3D matrix where each element (i, j, k) is the matching cost of pixel (i, j) for disparity k.

Each image passes through a set of layers that generate feature maps. The feature maps obtained from each image are then fed into a similarity calculation layer at the end. The weights of the feature extraction layers are shared at the time of processing both the left and right images. The last layer performs a dot product between the feature maps of the left and right images, which were previously normalized. The normalization and dot product steps are equivalent to the cosine similarity measure which is used to retrieve a raw disparity map based on the similarity of the feature maps.

The first three layers are a Convolutional layer, a ReLU activation layer and a Pooling layer. The parameters $< K, S, P >$ shown in figure 2 are the *kernel size*, *stride* and *padding* of the module. FM represents the number of features to be generated by the convolutional layer. The fire modules are composed of a set of layers. Each fire module shares the same $< K, S, P >$ parameters in the internal convolutional layers represented with Sqz, $Exp1x1$ and $Exp3x3$, but they differ in terms of feature maps.

Fire Module: The Fire module, depicted in the figure 3, is composed of two sequential steps with the purpose of squeezing the number of features received, and then expanding them in the next step, inspired by the Squeeze Net presented by Iandola et al.[15].

This model offers two advantages in terms of reduction of parameters. First, it reduces the number of input parameters to the convolutional layers. For example, an initial convolutional layer with 1x1 kernels (CL1x1) reduces by a factor of 9 the number of parameters of the layer when compared to a convolutional layer with 3x3 kernels (CL3x3), helping to decrease the size of the network. In

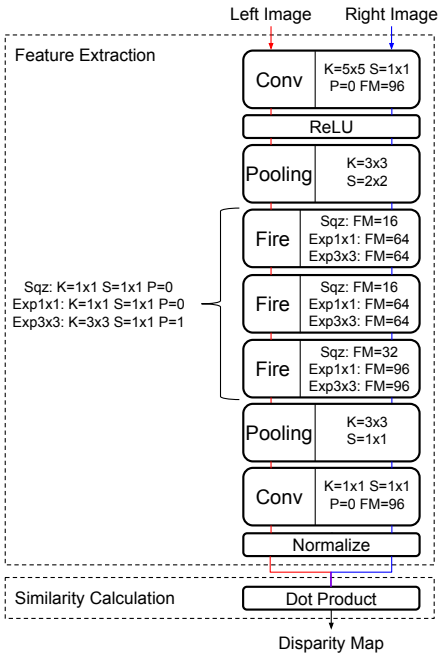

Fig. 2. Cost Calculation network architecture. The left and right images are passed through the same feature extraction layers, and both results are then processed in the similarity calculation layer

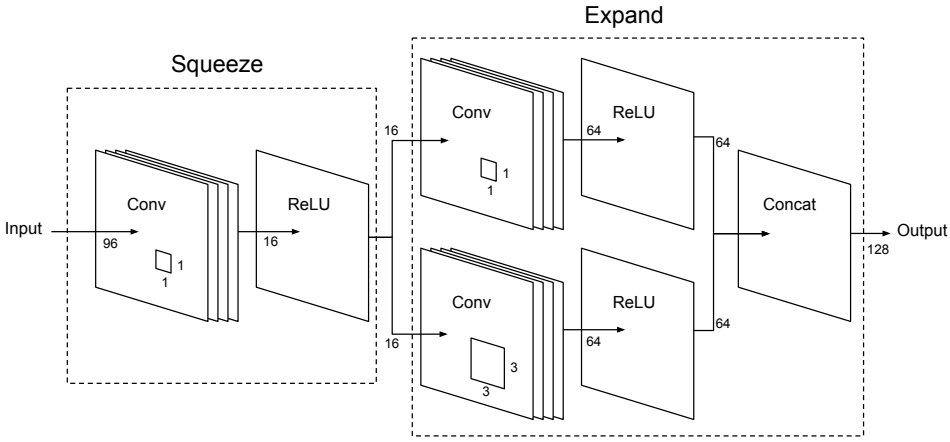

Fig. 3. Fire Module layers composition. The number of feature maps in the output of the layers is illustrative and changes in each one of the fire modules of the architecture.

addition, the number of features produced by the squeeze step is significantly reduced in contrast with other networks, generating a minimized input for the next CL3x3 of the expand module.

Second, in order to increase the number of features produced by the Fire module, a parallel CL1x1 is added, thus generating features maps that should otherwise be produced by the CL3x3. Consequently, the input required by both convolutional modules in the expand component to provide variety of feature maps can be reduced and so the number of parameters to be stored. It worths to mention that the CL1x1 of the expand module has padding equal to 1 in order to maintain the output feature map size equivalent to the CL3x3 one.

Cost Calculation: Since we assume that the images are rectified, their epipolar lines are completely horizontal and vertically coincident on both images. Unless they are occluded, the corresponding pixels can then be found in the same row, but differing by a certain disparity. Conversely, having the disparity corresponding to a pixel of one image, we can get the position of the matching pixel on the other image.

To train the proposed Squeeze Net we used the KITTI 2012 dataset[25] which offers both left and right images rectified and a group of ground truth disparities. Taking these latter ones along with one of the images, we can create training patches with examples of correct and incorrect matches as suggested by Zbontar et al.[12]. Each patch is a matrix containing pixel intensities. The negative examples are obtained by adding an offset to the disparity provided by the ground truth, and getting the patch from the new position. Also we augmented the training samples by performing a series of transformations on the patches like rotation, brightness and saturation level modifications, among others. In this way we can train the network with positive and negative matching examples for each position and disparity.

Our cost function is described in the equation 1, where P^L and P^R are the left and right patches, $\mathbf{p}$ is the position of the center of the matrix representing the patch, d is the horizontal displacement and f is the network output, that measures the similarity of the patches, being zero when they match exactly.

$$C_{\text{Squeeze-Net}}(\mathbf{p}, d) = f\big(P^L(\mathbf{p}), P^R(\mathbf{p} - d)\big) \tag{1}$$

Because the output size we want to produce is fixed, we need to calculate the patch size ws_k to produce it. The patch size is calculated only once, and is defined by equation 2, being k the number of the last convolutional or pooling layer, $<S, P, K>$ the module's *stride*, *padding* and *kernel size* respectively, and ws_{i-1} the output size of the module. Notice that ws_k is calculated iteratively, where variable i is initialized to 1 and it is increased step by step up to the number of convolutional and pooling layers, accumulating its values. In other words, to obtain the network minimum input size, equivalent to the patch size, we need to go backwards from the last convolutional or pooling layer to the first one.

$$ws_i = \begin{cases} 1 & i = 1 \\ \big((ws_{i-1} - 1) * S_{k-i+1}\big) - (2 * P_{k-i+1}) + K_{k-i+1} & 1 < i \leq k \end{cases} \quad (2)$$

Network Training: During training, the network analyzes pairs of left/right patches, and the result is a measure of their similarity, as stated in equation 1. Then, the loss function to be minimized to adjust the networks parameters applies the hinge-loss to pairs of positive and negative matching samples, as defined by equation 3

$$L = max(0, margin + m^- - m^+) \quad (3)$$

where m^- and m^+ are the results of the negative and positive matching examples, and $margin$ is the tolerance margin. When the similarity of the positive example is greater than the similarity of the negative one by a certain margin the loss will be zero. In our experiments the $margin$ used was 0.2.

3.2 Post-processing module

The cost calculation network provides a representation of a raw disparity map, in the form of a 3D matrix where each element (i, j, k) is the matching cost of pixel (i, j) for disparity k. The quality of this map can be improved through a set of post-processing steps. In our work we only perform a Semi-Global Matching inspired by Hirschmüller et al.[22].

Semi-global Matching (SGM) An important refinement of the disparity map is related to smoothness. Since objects usually have similar disparities, intuitively we can say that differences in the disparity of neighboring pixels should be penalized. In general, the penalty applied increases according to how strong the difference between disparities is. We defined the local neighborhood of a pixel by moving 1 pixel away of if in the up, down, left and right directions, following the suggestion in [12]. If we call $\mathbf{r}$ the vector of the directions where the pixels $\mathbf{q}$ are found, a set of adjustments $C_{\mathbf{r}}(\mathbf{p}, d)$ for the matching cost $C_{\mathbf{r}(\mathbf{p},d)}$ computed by the CNN are calculated through equation 4 (and later averaged, as described below).

$$C_{\mathbf{r}}(\mathbf{p}, d) = C_D(\mathbf{p}, d) - \min_i C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, i) + \min \Big(C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d), \\ C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d-1) + P_1, C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d+1) + P_1, \min_i C_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, i) + P_2 \Big) \quad (4)$$

Here C_D is the matching cost calculated by the CNN, P_1 is the penalty when the difference between the disparity $D_{\mathbf{p}}$ and $D_{\mathbf{q}}$ of pixels $\mathbf{p}$ and $\mathbf{q}$ in the local neighborhood is 1. P_2 is the penalty when that difference is higher than 1, and i

are the valid disparities. The second term of the equation compensates for cases where the values of the third term of the equation grow too large, for smoothing special cases (e.g. occluded or mismatching pixels).

The penalty values P_1 and P_2 varies according to disparity of the pixel compared with neighbors in edges of objects. Therefore, penalties are lower when pixels are detected in borders. The parameters are defined as follows:

$$D_1 = |I^L(\mathbf{p}) - I^L(\mathbf{p}-\mathbf{r})| \qquad D_2 = |I^R(\mathbf{p}-\mathbf{d}) - I^R(\mathbf{p}-\mathbf{d}-\mathbf{r})|$$

$$\begin{array}{llll} P_1 = P_1, & P_2 = P_2, & \text{if } D_1 < D_{sgm} \wedge D_2 < D_{sgm}; \\ P_1 = P_1/Q_2, & P_2 = P_2/Q_2, & \text{if } D_1 \geq D_{sgm} \wedge D_2 \geq D_{sgm}; \\ P_1 = P_1/Q_1, & P_2 = P_2/Q_1, & \text{otherwise} \end{array} \tag{5}$$

where I is the pixel intensity. In case the disparity shows a big discontinuity in the disparity map, i.e. when D_1 and D_2 are equal or bigger than a certain D_{sgm}, the penalty is reduced by a large factor Q_2 as it is assumed that it is a steep border. In case just D_1 or D_2 meets this condition, the border is not that steep so the penalty is reduced by a smaller factor Q_1. Otherwise the case is not a border. In case of vertical directions a different factor V is used to reduce P_1 as the disparities changes shown in ground truth images are more frequently vertical.

After computing all these values for each direction, the final smoothed cost is an average of the results obtained (equation 6).

$$C_{\mathrm{SGM}}(\mathbf{p}, d) = \frac{1}{4} \sum_{\mathbf{r}} C_{\mathbf{r}}(\mathbf{p}, d) \tag{6}$$

3.3 Map Construction

The disparity map is constructed by gathering the minimum cost from the matrix of matching costs for each pixel position. This strategy is called winner-takes-all and is defined in the equation 7.

$$D(\mathbf{p}) = \arg\min_{d} C(\mathbf{p}, d) \tag{7}$$

3.4 Refinement module

Once the disparity map is constructed, problems like mismatching or occluded pixels can reduce its accuracy. This can be improved through different interpolation and refinement steps. These steps were inspired by Mei et al.[19] work. The execution order is depicted in figure 4.

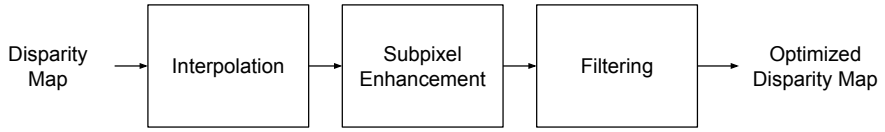

Fig. 4. Refinement steps performed after the map construction module

Interpolation: The disparity map can be calculated either using the left image as reference or the right image. Changing the reference image produces different maps since the occluded pixels are different on each one. Having both disparity maps can help us to determine which are those occluded object by comparing the disparities of a matching pixel. If the disparities match (the absolute difference is less than one), we can consider them as correct. If the disparity of the pixel in one map matches the disparity of a pixel in the other map other than the corresponding one, we consider it as incorrect. If the disparity does not match any other disparity in the other map, it is an occluded pixel.

The occluded pixels disparity is obtained by looking at the nearest correct pixel at the left. For mismatching pixels we look for a disparity value as the median of sixteen directions pixels around them.

Sub-pixel Enhancement: In this step we use the SGM cost of pixel p and disparity d and its closest "disparity" neighbors to get a smoothing subtraction term, to improve the result, as shown in equation 8.

$$D(\mathbf{p},d) = D(\mathbf{p},d) - \frac{C_{SGM}(\mathbf{p},d+1) - C_{SGM}(\mathbf{p},d-1)}{2\Big(C_{SGM}(\mathbf{p},d+1) - 2C_{SGM}(\mathbf{p},d) + C_{SGM}(\mathbf{p},d-1)\Big)} \quad (8)$$

Filtering: This module applies a median filter with a 5x5 kernel followed by a bilateral filter, for the purpose of smoothing disparity changes without affecting the edges.

4 Case Study

In this section we present details of the environment where the different algorithms were executed, the training specifications and the system setup. The results obtained are shown in terms of parameters reduction, error rate, execution time and memory consumption. The experiments were performed on an AMD Ryzen 1700 CPU with 32 GB DDR–4 2400 MHz ram and a NVidia Titan Xp GPU.

Training set: We used the 2012 KITTI dataset [25] composed by 194 training pairs of images of 1240 x 376 pixels. These images have a maximum of 228 disparity levels. The dataset provides around 30% of the image disparities measured with laser scanners. Leaving 40 images for test, the remaining 154 images made a training set of around 19 million positions with measured disparity. Since we get positive and negatives examples as mentioned in Section 3.1, the number of training examples (sampled patches) is more than 38 million. Each sample is subject to several image transformations in order to provide different samples on each epoch. As a product of the CNN layers' kernels and padding the window size of our patches is 9x9.

Learning parameters: These training samples were provided in batches of 128 samples during 15 epochs. We used a learning rate of 0.05 which is decreased after epoch 11 by a factor of 10. The parameters used for image post-processing are the same used in Zbontar et al.[12] fast architecture.

Cost calculation parameters: The proposed Squeeze Deep Neural network considerably decreased the number of parameters in relation with the other two networks. This result is the product of the reduction of features in the squeeze layer of the fire module that are fed into the expansion CL3x3 module and the expansion of features through a parallel CL1x1. A comparison of the network size, the parameters involved and the reduction rate is shown in the table 1.

Table 1. Cost Calculation Deep Neural network sizes and parameters

Network	Size (KB)	# Parameters	Reduction
Zbontar Accurate	2,534	648,592	87.81%
Zbontar Fast	440	112,564	29.77%
Squeeze Net	**309**	**79,040**	-

System accuracy and execution time: The system was executed in two instances: with post-processing and refinement, and without it. Table 2 shows the different results obtained as an average over the 40 testing images.

Table 2. System accuracy and execution time

	Error		Execution time	
Network	CNN only	Full method	CNN only	Full method
Zbontar Accurate	6.03%	2.54%	33.24 sec	33.84 sec
Zbontar Fast	8.39%	2.93%	0.33 sec	0.65 sec
Squeeze Net	11.87%	3.69%	0.58 sec	0.89 sec

Even through the error is 6% higher in case the system is executed without post-processing and refinement, the degradation of the quality including these two modules is below 1.5% even compared with the accurate architecture. The execution time of the network is only comparable with the fast architecture and is over 38 times faster than the accurate architecture. The resulting disparity maps of each model, including post-processing and refinement steps, are shown in figure 5.

Fig. 5. Full method generated disparity maps

GPU Memory consumption: There is a reduction of memory consumption by our solution, as compared to the fast architecture, making it feasible to use in smaller devices with less GPU resources as presented in table 3.

Table 3. Architecture GPU memory consumption comparison

Network	Zbontar Accurate	Zbontar Fast	Squeeze Net
GPU Mem	∼4200 MB	∼1332 MB	∼1975 MB

5 Conclusions and Future Work

The long term objective of this research is to reduce the computational resources required by these models, to make their deployment on smaller devices feasible. These resources include memory, execution time, storage and communications size. In this paper we presented a Cost Calculation CNN based module built as a Squeeze Network to generate an initial disparity map. This network was combined with post-processing and refinement algorithms to improve the final disparity map quality. In the tests performed the system showed a reduction of almost 30% of the number of parameters. The cost of such a reduction was less than 1.5% of accuracy and less than 250 ms of execution time when compared to state of the art networks. The GPU memory used was comparable with the fast architecture and consumed less than a half the accurate architecture. Thus, the results show the utility of our architecture in terms of reducing the size of the network, and it is a first step towards the more general goal of reducing the execution time and memory required.

The model proposed might be improved to obtain better quality on disparity maps and less execution time by experimenting with different architectures and hyperparameters, like the combination of squeeze nets and residual networks. Also, different techniques of parameter size reduction like pruning, as mentioned in Iandola et al[15], to minimize even more the network size, will be explored in future work.

6 Acknowledgements

This research was supported by CONICET (National Council of Scientific and Technological Research), and LABSIN (Intelligent Systems Laboratory) of the School of Engineering (National University of Cuyo). We also want to gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

1. Hannah, M.J.: Computer matching of areas in stereo images. Technical, Stanford University (1974)
2. Marr, D., Poggio, T.: Cooperative computation of stereo disparity. Science **194**(4262), 283–287 (1976)
3. Scharstein, D., Szeliski, R.: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. International Journal of Computer Vision **47**(1–3), 7–42 (2002)
4. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
5. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation **1**(4), 541–551 (1989)

6. Hinton, G.E.: Learning multiple layers of representation. Trends in Cognitive Sciences **11**(10), 428–434 (2007)
7. Torch home page, http://torch.ch/
8. Tensorflow home page, https://www.tensorflow.org/
9. Theano home page, http://deeplearning.net/software/theano/
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems **25**(2), 1097–1105 (may 2012)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015), http://arxiv.org/abs/1512.03385
12. Žbontar, J., LeCun, Y.: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. Journal of Machine Learning Research **17**, 1–32 (2016)
13. ImageNet Large Scale Visual Recognition Challenge (ILSVRC) home page, http://image-net.org/challenges/LSVRC/
14. The KITTI Vision Benchmark Suite evaluation page, http://www.cvlibs.net/datasets/kitti/eval_stereo.php
15. Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. CoRR **abs/1602.07360** (2016), http://arxiv.org/abs/1602.07360
16. Hamzah, R.A., Ibrahim, H.: Literature Survey on Stereo Vision Disparity Map Algorithms. Journal of Sensors **2016**, 1–23 (2016)
17. Zeiler, M.D., Fergus, R.: Visualizing and Understanding Convolutional Networks. Computer Vision-ECCV 2014 **8689**, 818–833 (2014)
18. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision **115**(3), 211–252 (2015)
19. Mei, X., Sun, X., Zhou, M., Jiao, S., Wang, H., Xiaopeng Zhang: On building an accurate stereo matching system on graphics hardware. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops). pp. 467–474. IEEE, Barcelona (nov 2011)
20. Shaked, A., Wolf, L.: Improved stereo matching with constant highway networks and reflective confidence learning. CoRR **abs/1701.00165** (2017), http://arxiv.org/abs/1701.00165
21. Ke Zhang, Jiangbo Lu, Lafruit, G.: Cross-Based Local Stereo Matching Using Orthogonal Integral Images. IEEE Transactions on Circuits and Systems for Video Technology **19**(7), 1073–1079 (jul 2009)
22. Hirschmüller, H.: Stereo Processing by Semiglobal Matching and Mutual Information. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**(2), 328–341 (feb 2008)
23. Yao, G., Liu, Y., Lei, B., Ren, D.: A rapid stereo matching algorithm based on disparity interpolation. In: Proceedings of 2010 Conference on Dependable Computing. pp. 5–10 (2010)
24. Miclea, V.C., Vancea, C.C., Nedevschi, S.: New sub-pixel interpolation functions for accurate real-time stereo-matching algorithms. In: 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP). vol. 20, pp. 173–178. IEEE (sep 2015)
25. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)