Gradient-learned Models for Stereo Matching
CS231A Project Final Report

Leonid Keselman
Stanford University
leonidk@cs.stanford.edu

Abstract

In this project, we are exploring the application of machine learning to solving the classical stereoscopic correspondence problem. We present a re-implementation of several state-of-the-art stereo correspondence methods. Additionally, we present new methods, replacing one of the state-of-the-art methods for stereo with a proposed technique based on machine learning methods. These new methods out-perform existing heuristic baselines significantly.

Figure 1. An example of a stereo left-right pair from the Middlebury 2014 dataset [18]. The motorcycle scene will be consistently used through this paper as a visual example result .

1. Introduction

Stereoscopic correspondence is a classical problem in computer vision, stretching back decades. In its simplest form, one is given a calibrated, rectified image pair where differences between the pair exist only along the image width. The task is to return a dense set of corresponding matches. An example image pair is shown in figure 1.

While this task may seem straightforward, the primary challenge comes from challenging, photo-inconsistent parts of the image. Parts of the image will often contain ambiguous regions, a lack-of-texture, and a photo-metric mismatch for a variety of reasons (specular reflections, angle of view, etc.).

This field is well studied, and there exist many standard datasets, such as Middlebury [18] and KITTI [6]. Both of these datasets contain many rectified left-right pairs, along with corresponding ground truth matches. The former dataset contains high resolution images from largely indoor scenes, and comes from using a method of dense structured light correspondence method. In contrast, the KITTI dataset contains much lower resolution images, and consists of outdoor data gathered from a vehicle perspective. Additionally, the KITTI dataset's annotations come from a LIDAR technique, and are generally sparse. While KITTI seems more popular based on the size of their leader-board, the dense annotations available in Middlebury [18] are useful for the problem tackled in this paper, and our results are only reported there.

The interest in this problem has important practical application in autonomous vehicles and commercial applications. For example, the KITTI dataset was formed to test if low-cost stereoscopic depth cameras to replace high-cost LIDAR depth sensors for autonomous vehicle research. In a different field, commercial depth sensors such as the original Microsoft Kinect and Intel RealSense R200 use stereoscopic correspondence to resolve depth for tracking people and indoor reconstruction problems. As such, improved methods for stereoscopic correspondence have wide application and use.

2. Related Work

2.1. Previous Work

In the field of stereo matching, one of the recent innovations in the past few years was the use of convolutional neural networks in improving the quality of matching results [21]. At the time of it's announcement at CVPR 2015, it was the top performer in both standard datasets. Even at the present day, all better performing methods on the Middlebury leaderboard use the Matching Cost CNN (MC-CNN) costs as a core building block. Their primary contribution is to train a convolutional neural network (CNN) to replace the block-matching step of a stereo algorithm. That is, instead

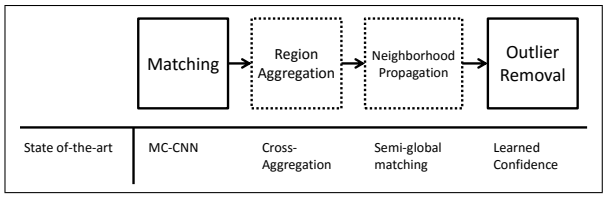


Figure 2. The architecture and algorithm flow for state-of-the-art methods in stereo. They first use the MC-CNN cost function [21], then combine those results with cross-based aggregation [22], and share them with neighbors using semi-global matching [7]. Methods in dashed lines are heuristic methods, while the ones with solid lines use a machine-learned method.

of using a sum of absolute differences cost such as

$$Cost(source, target) = \sum_i \sum_j |source_{ij} - target_{ij}|$$

or a robust non-parametric cost function such as Census [20]

$$R(P) = \otimes\zeta(P, P_{ij})$$

$$Cost(source, target) = popcnt(source_{ij} \oplus target_{ij})$$

the authors of [21] learn a network to compute a $Cost(source, target)$ metric based on the ground truth available from stereoscopic correspondence datasets. An example of their network architecture is shown in figure 3.

However, in order to obtain their final result, they use a combination of algorithms to select an optimum match. Namely, they use a combination of their cost metric [21], cross-based aggregation [22], and semi-global matching [7]. This flow is shown in figure 2. We hypothesize that a short-coming of this state-of-the-art method is that two of the techniques used in the algorithm flow make use of a heuristic method for completing a certain task. We hope to build on the success of the MC-CNN method and use gradient based learning [12] to replace other components of the stereo algorithm. The value in picking this specific classification algorithm is that it has the ability for us to eventually design an end-to-end gradient-learned system that trains an MC-CNN along with our proposed system. The goal for the project is first implement these baseline algorithm methods, and then begin to to test and design algorithms and methods to replace one of the two heuristic algorithms in the traditional stereoscopic pipeline, namely semiglobal matching [7] or cross-based aggregation [22]. In this report, we only present methods for replacing semiglobal matching, but not yet cross-based aggregation.

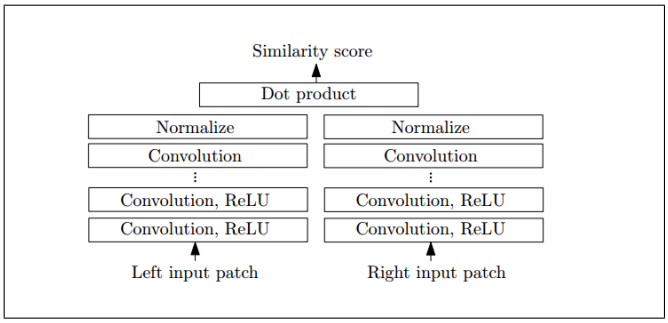


Figure 3. The matching architecture of [21], the current state-of-the-art in stereo matching.

2.2. Key Contributions

1. **A fast, flexible implementation of stereo matching** We present a new, from-scratch implementation of state-of-the-art methods in stereo matching, including Census [20], semiglobal matching [7], cost-volume filtering [10]. Along with standard methods for hole filling, like those used in MC-CNN [21], and many outlier removal methods [16]. The implementation is cross-platform, C++, multi-threaded, and uses no libraries except those for loading and saving images. It is fast, and produces competitive RMS error results on standard datasets. See section 3.1.

2. **A machine-learned method for correlation selection** We've implemented and tested several semiglobal matching replacement architectures, trained them on the Middlebury training data, and demonstrated that they perform significantly better on out-of-bag examples than semiglobal matching. See section 4.

3. Baseline Implementation

3.1. C++ Baseline

First, we implemented stereo matching baselines using current, non-machine learned methods. The stereo algorithms described below were implemented from scratch, in C++, with no external libraries outside of image loading. The performance of our baseline is later summarized in table 1.

An example left-right pair is show in 1. We're using quarter-sized training images from the latest Middlebury dataset [18]. These are roughly 750x500 pixels in resolution. The results from our algorithm are compared to the ground truth visually in figure 4 and quantitatively in table 1. An elaboration of the different papers and methods implemented for each section is described below. The code is all C++11, and compiles on Visual Studio 2013 and gcc, with no external libraries. Threading is implemented via OpenMP [15] to parallelize cost computation across all pix-

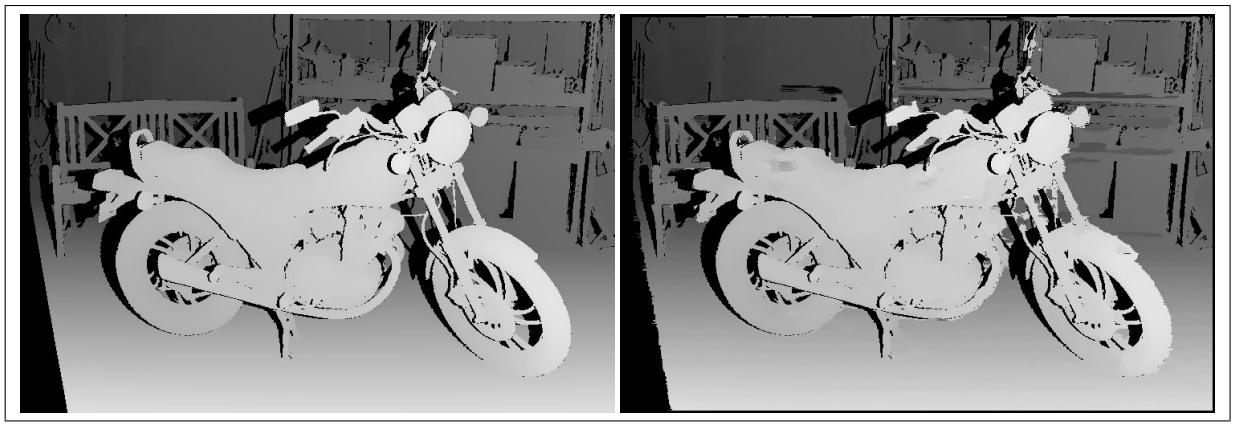


Figure 4. The image on the left is ground truth for the motorcycle scene in 1. The image on the right is the results of our semi-global matching pipeline with naive hole filling as described in section 3.1. For visual comparison, occluded and missing ground truth pixels from both images are masked out.

els in a given scanline. This cost accumulation is the primary computational bottleneck in the system so parallelizing that component is enough to provide sufficient scaling across processor cores.

3.1.1 Cost Computation

As a baseline method of cost computation, we've implemented both standard sum of absolute differences, and the robust Census metric [20]. Census was recently tested and shown to be the best performing stereo cost metric [8]. The weighted sum of absolute differences and Census was additionally state-of-the-art for Middlebury until a year or two ago [13]. The state of the art in this space is MC-CNN method [21], which implemented a CNN algorithm to replace traditional methods. However, since our project focuses on implementing neural networks in other parts of the stereo pipeline, re-implementing this cost metric is not a high priority.

Specifically, we implemented Census with 7x7 windows, which allows us to exploit a sparse census transform [5], and fit the result for every pixel into 32-bits. This enables efficient performance with the use of a single *popcnt* instruction on modern machines.

3.1.2 Region Selection

For our region selection baseline, we've implemented both box correlation windows and weighting with a non-linear smoothing algorithm such as the bilateral filter [19]. This was inspired recent unpublished ECCV 16 submissions on the Middlebury leader-board, which claim to replace the popular cross-based [22] with a smooth affinity mask method like a bilaterial filter, as first shown in [10].

3.1.3 Propagation

In order to perform propagation across the image, we've implemented semi-global matching [7], in full, as described in the original paper. We chose to perform 5-path propagation for each pixel, as it represents a row causal filter on the image, using a pixel's left, right, top, top-left, and top-right neighbors. This produces an answer that satisfies the cost function of Hirchmuller [7]

$$E(D) = \sum_{p}(C(p, D(P)) \tag{1}$$

$$+ \sum_{q} P_1 \cdot 1\{D(p) - D(q)\} = 1$$

$$+ \sum_{q} P_2 \cdot 1\{D(p) - D(q)\} > 1$$

Additionally, we added naive hole filling by propagating pixels from left-to-right, in order to fill occluded regions. This is a naive metric, but is a large part of the hole filling used in the state of the art work [21].

4. Learning Propagation

Most papers in the KITTI dataset build on top of the successful method of semi-global matching [7], which is an algorithm for propagating successful matches from pixels to their neighbors. The goal of this part of the project was to replacing this function with either a standard neural network, or recurrent neural network. Depending on one's perspective on what operation semiglobal matching is performing, there is a wide array of neural network architectures that may be amenable to replace it. An overview of the formulations is shown in figure 5.

The first and most straightforward view of what the energy function, as stated in equation 1, is that it regularizes a

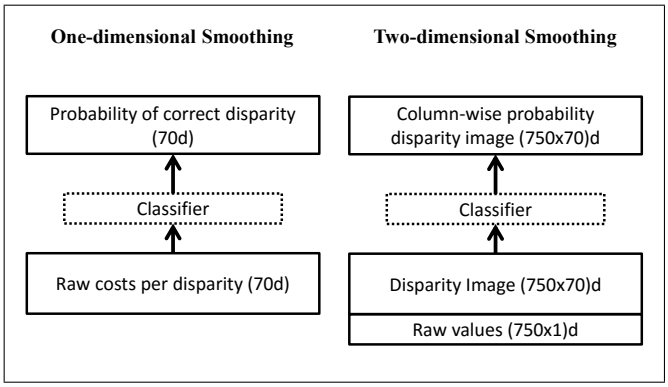


Figure 5. An example of two different ways to formulate semi-global matching as a classification task. The one on the left is explored in section 4.1, while the one on the right is explored in 4.2.

single pixel's correlation curve into a more intelligent one. This view is fairly simple, doesn't incorporate any neighborhood information, but in our testing was the most successful model. This is elaborated in section 4.1.

A second view of what semiglobal matching does in practice is that is regularizes an entire scanline at a time, performing scanline optimization and producing a robust match for an entire set of correlation curves at once. This was the view we took when building models in section 4.2.

A third view of what semiglobal matching does is it serves as a way of remembering good matches, and propagating their information to their neighbors. This is straightforward and almost certainly what semiglobal matching does. This would require a pixel recurrent neural network such as that in [14]. In our limited time and testing, we were unable to get any of these architectures to converge and hence have excluded them from this paper. However, our primary focus was on building a bidirection RNN with GRU [3] activations. In practice, small pixel patches didn't converge while large patches were not able to fit into the memory of the machines we had available for training.

For testing and training, we gather a subset of the Middlebury images [18], and split into into a random training and testing set with a 80%-20% split. The unseen samples are then used for evaluation. Middlebury provides 15 images for training and 15 for evaluation. For the classifiers in section 4.1, this results in roughly 500,000 annotations per image (using quarter sized images), and 500,000 tests of the network. While for the classifiers in section 4.2, this results in 500,000 annotations computed over about 1,000 runs of the network (since it computes 500 outputs at the same time). See below for details of how this is implemented.

4.1. 1D Smoothing

One straightforward view of semi-global matching is simply as regularization function on top of a pixel's correlation curve. A correlation curve is the set of matching costs for a single pixel and it's candidates. If this input is negated, and fed a softmax activation function, as used to train many neural networks, it treats the values as unnormalized log probabilities, and selects the maximum (which would be the candidate with lowest matching cost).

$$L_i = -\log(\frac{e^{f_{y_i}}}{\sum_j e^{f_i}})$$

Our original implementations for this method were all straightforward multi-layer perceptions (MLP), using a one, two, or three layer neural network to produce a smarter minimum selection algorithm. However, no matter the loss function, shape, dimensions, regularization, or initialization function, we were unable to get any MLP to converge. That is, using a 0-layer neural network (the input itself) was better than any learned transformation to that shape and size.

Instead, we found success by using a one dimensional convolutional neural network as shown in figure 6. We suspect a CNN was able to handle this task better, as one bank of convolutions could learn an identity transform, while others could learn feature detectors that incorporated interesting feedback into that identity transform. In comparison, a randomly initialized fully connected network may struggle to learn a largely identity transform with minor modifications. We implemented the neural network on top of Keras [4] and TensorFlow [1]. We additionally learned several non-gradient based classifier baselines such as SVMs and random forests using scikit-learn [17].

4.2. 2D Smoothing

As shown in figure 5, there is an alternative concept of how semiglobal propagation. This one incorporates pixel neighborhoods, and seems a more natural fit for the energy function presented in equation 1. For this formulation of a neural network, the correlation curves of an entire scanline are reshaped into an image in disparity cost space notation is described in figure 7. We then create a model using a two-dimensional convolutional neural network [12] on top of these disparity cost space images The top level is a column-wise softmax classifier of the same size as the input dimensions. In order to implement this in TensorFlow [1], we first pass in a single disparity image as a single batch. We run our convolutional architecture over this model, and then reshape the output into pixel-many "batches" for each of which we have a label. This allows the built-in softmax and cross-entropy loss formulations to work out-of-the-box with no hand-made loops.

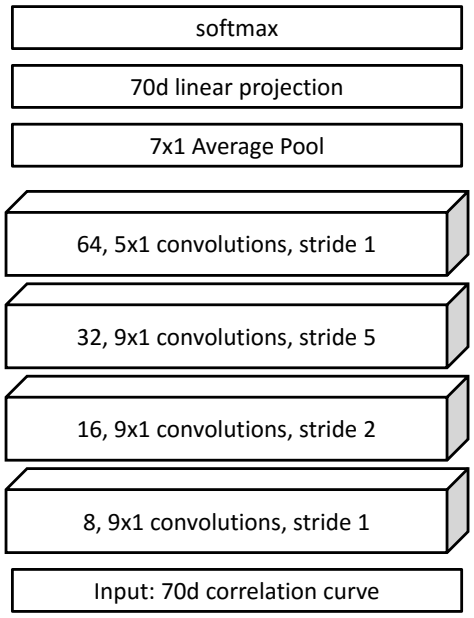


Figure 6. An architectural view of our most successful machine learned method, a 1D CNN for predicting better minimums in correlation curves.

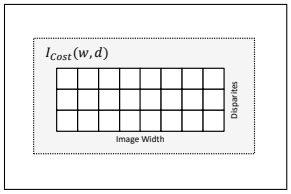


Figure 7. A brief visual diagram of a Cost Image for single scanline of stereo matching. Each pixel contains the cost of matching for that value, at that image pixel. Across the entire image, there exists a cost volume across all scan-lines in a stereo pair, our proposed architectures only deal with a known, discrete number of cost images.

5. Experiments

5.1. Baseline Method

We tested our baseline C++ implementation of modern stereo matching for both time and quality of output. Specifically, we focused on simply a single Middlebury training image (the Motorcycle) to validate that our results were within expectation for a stereo matching baseline. Our two key metrics were runtime and root-mean-squared error for all the dense, all-pixel label ground truth. This is just one of the metrics for Middlebury, but is one that measures the quality of all pixels predicted by the classifier. A summary table is shown in table 1. We show that our baseline implementation is on the same order of magnitude as the SSE-optimized, hand-tuned implementation of semiglobal matching available from OpenCV [2]. We believe that both the performance and accurate matching is a function of us using the robust and fast ADCensus [13] [20] weighted cost function. Since the primary focus of this project as to simply provide a flexible baseline for quickly generating data for the machine learned methods in section 4, we did not spend much time micro-optimizing or tuning algorithm hyper-parameters. However, if one wished to tune this algorithm there are dozens of knobs, including the relative weighting of absolute differences and Census, the regularization strengths of P_1 and P_2 from semiglobal matching, and the weights used in the bilateral filter.

Model	RMS Error	Runtime
Census	28.92	1.2s
SGBM	28.12	3.1s
SGBM + BF	32.8	5.8s
OpenCV SGBM	38.00	0.9s
MC-CNN (acct)	27.5	150s

Table 1. A summary table of numerical results on the training Motorcycle image. The error metric is root-mean-squared error in disparity space, and the run-times are on a quad-core i7 desktop. The first three lines are baseline implementations implemented by us, while the last two are standard algorithms available on the dataset website [18]. The MC-CNN results were run on a GPU [21] (which were on an GPU).

5.2. Learned Propagation methods

In the scope of testing the various propagation classifiers, we adopt two different evaluation metrics. The first is the traditional training/test split used in machine learning methods. The other is the RMS error metric used for stereo algorithm evaluation. A result comparing standard methods and our proposed classifiers on test data is shown in figure 8.

We see that the one-dimensional CNN as presented in section 4.1 and shown in figure 6 outperforms the current standard methods for smoothing matches. That is, when fed with the ADCensus correlation curves generated by our matching algorithm, the neural network generates predictions that are much more accurate than the heuristic semiglobal matching method used in state-of-the-art papers such as MC-CNN [21]. This result is even true when we take the network's predictions back to the matching algorithm and use it to generate a full correspondence image. Even though the neural network (currently) lacks the ability to make subpixel accurate guesses, it generates lower RMS error than standard methods like Census and semiglobal matching, which do have subpixel matching built into the baseline.

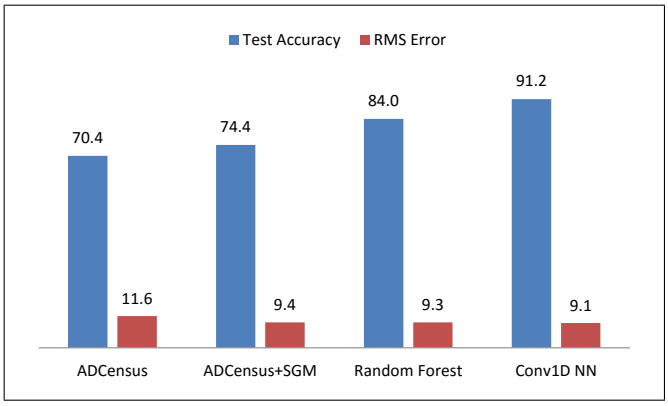


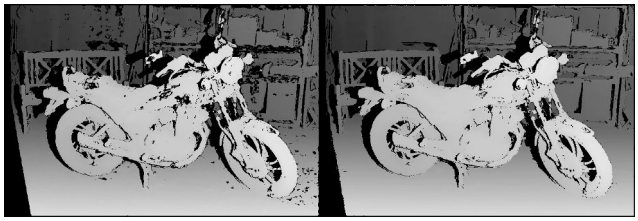
Figure 8. Numerical results on the out-of-bag testing data across the a subset of the Middlebury [18] images.

Model	Out-of-bag Accuracy
Census	67.7%
SGBM	69.2%
1D CNN Training	76.4%
1D CNN Test	75.8%
2D CNN Training	58.5%
2D CNN Test	55.4%

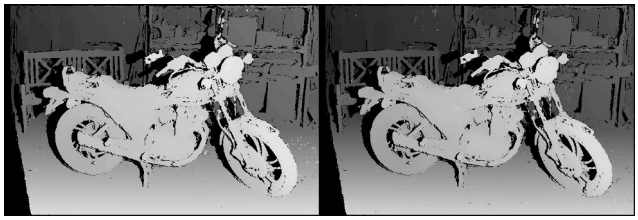
Table 2. An summary table of numerical results when testing on a large batch of Middlebury testing images

Additionally, as can be seen in table 2, the 1D CNN model is not yet exhibiting overfitting on out-of-bag samples, and might benefit from additional training time. It can also be seen that our best 2D CNN architecture drastically underperforms even the standard baselines. While there may be some more optimal 2D CNN architecture than the one we tried, our poor initial results made us moved towards trying to build an RNN method instead. However, we did not have enough time to finish designing and training our RNN models for replacing semiglobal matching.

Another interesting experimental result is the qualitative performance of the classifier models. As shown in figure 9, the classification-based models sometimes generate completely erroneous results for parts of the image. While Census will fail to generate a result sometimes, and semiglobal matching learns a smooth transformation. In contrast, while the classification models have lower error, they sometimes predict very non-smooth results, as the classifier is run per pixel. This is suggestive that a classifier, such as an RNN, that accounts for neighborhood information may perform even better. Also, while we did not combined semiglobal matching with our 1D CNN, it is possible to use the normalized probabilities from the neural network together with semiglobal matching to overcome this lack of smoothness and achieve perhaps an even better result.



(a) Census and Semiglobal Results



(b) Random Forest and 1D CNN Results

Figure 9. A qualitative example using the presented classifiers. It can seen that the original cost method (Census) is able to resolve certain parts of the scene. On the other hand, semiglobal propagation is able to in-paint the image and generate a smooth disparity image. On the other hand, the errors made by the two classifier models, although having better accuracy and RMS error than the heuristic methods, sometimes generate what looks like completely erroneous results.

6. Conclusion

We have presented a new method for taking stereoscopic correlation costs and smoothing them into a more refined estimate. This method is gradient-trainable, and outperforms the semiglobal matching [7] heuristic technique used in state-of-the-art methods such as MC-CNN [21]. This leads support to the hypothesis proposed in the introduction, which is that continuing to replace components of the stereo matching pipeline with machine-learned models is a way to improve their performance. Since the models presented here were done with ADCensus costs [13] and not MC-CNN costs [21], and we did not have enough time to train on the full Middlebury dataset [18], we don't present a new state-of-the-art for stereoscopic correspondence. However, we believe that these results suggest that one may be possible by simply running the proposed techniques with MC-CNN on the full dataset.

In addition, we've created a new, simply, fast and cross-platform stereo correspondence implementation. We've shown it to be about as fast as the one in OpenCV, and to produce results that are notably more accurate. We hope this can be used as a base for others to experiment with other stereoscopic correspondence ideas without having to dive into complicated OpenCV SSE code or deal with slow MATLAB implementations.

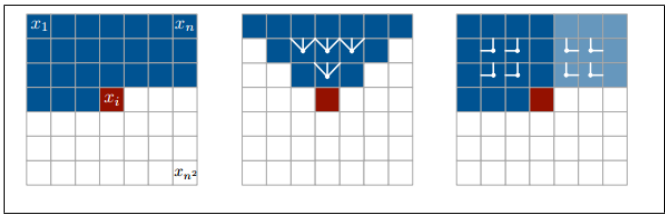


Figure 10. An example of a pixelwise RNN from [14], a gradient-learned method for propagating information across images.

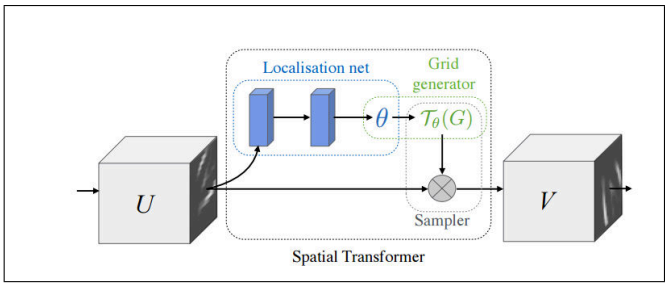


Figure 11. An example of a spatial transformer for region selection [11], a gradient-learned method for region selection.

7. Next Steps

To continue this theme of research, we wish to explore additional architectures for stereo correspondence algorithms that are trained with error gradients. While the one-dimensional CNN presented here works well, it isn't able to capture the neighborhood information that semiglobal matching can. To incorporate neighborhood information, we'd like to explore recurrent neural network models, which we began to design but were unable to get running in time for this project submission. By coupling our 1D-CNN architecture with either a spatial transformer networks front-end [11], or a recurrent neural network backend [3] [9] , we might produce a new state-of-the-art algorithm for the classic stereo problem. Examples of these models are shown in figures 10 and 11.

8. Code

Code is made available at `https://github.com/leonidk/centest`. Running the stereo matching algorithm is straightforward and documented in the README, but running the learning algorithms (found in the learning/ folder) varies depending on the method.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] G. Bradski. Opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[4] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[5] W. S. Fife and J. K. Archibald. Improved census transforms for resource-optimized stereo vision. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(1):60–73, 2013.

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] H. Hirschmüller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.

[8] H. Hirschmüller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(9):1582–1599, 2009.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] A. Hosni, M. Bleyer, C. Rhemann, M. Gelautz, and C. Rother. Real-time local stereo matching using guided image filtering. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[11] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang. On building an accurate stereo matching system on graphics hardware. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 467–474. IEEE, 2011.

[14] A. V. D. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016.

[15] OpenMP Architecture Review Board. OpenMP application program interface version 3.0, May 2008.

[16] M.-G. Park and K.-J. Yoon. Leveraging stereo matching with learning-based confidence measures. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 101–109. IEEE, 2015.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss,

V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[18] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *Pattern Recognition*, pages 31–42. Springer, 2014.

[19] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.

[20] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Computer VisionECCV'94*, pages 151–158. Springer, 1994.

[21] J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *CoRR*, abs/1510.05970, 2015.

[22] K. Zhang, J. Lu, and G. Lafruit. Cross-based local stereo matching using orthogonal integral images. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(7):1073–1079, 2009.