Stereopsis with Convolutional Neural Networks and Conditional Random Fields

Reuben Feinman & Ambuj Ojha

Abstract

The human visual system collects two distinct images of the environment, corresponding to the right and left eyes. These images differ slightly, and the brain uses the relative difference–known as "binocular disparity"–to perceive depth in the underlying visual scene [4]. Since its original formulation by Marr & Poggio in 1976 [5], the question of what computations might underlie this perceptual task has sprawled into an active area of vision research. In this project, we present a powerful computational framework for decoding depth from stereo vision. Our method makes use of a combination of Convolutional Neural Networks (CNNs) and Conditional Random Fields (CRFs), and it performs impressively well while requiring **zero labelled training examples.** We first give a background on disparity computation literature. Then, we present our algorithm and results. In addition to presenting our method, we are releasing an open-source code repository with highly modularized Python implementations of the disparity computation algorithm[1].

1 Background

The brain uses binocular vision to perceive depth in the surrounding environment [4]. It achieves this task by computing "binocular disparity" across the visual field, i.e. the relative difference between left and right eye signals. This disparity value is directly correlated with depth in the underlying visual scene. In computer vision, stereo matching algorithms attempt to replicate this process, receiving as input one "left eye" image and one "right eye" image and searching for correspondences between the two images to estimate disparity. The algorithms look at local blocks of pixels, finding matches between blocks in the left and right image [2, 3, 13]. These methods perform considerably well when measured by the average of individual pixel results; however, the resulting depth maps are often very noisy, and they appear to neglect local dependencies that exist between latent disparity variables.

Preliminary work with Conditional Random Field models (CRFs) demonstrates that it is beneficial to incorporate a smoothness prior over local neighborhoods of disparity space [11, 8]. In these models, unary factors assign a preference for data fitting, and binary factors assign a preference for smoothness. The data fit is computed by means of standard stereo matching algorithms that operate in pixel space, and smoothness is computed as the total difference between neighboring latent variable values. By incorporating a prior that prefers local consistency (or "smoothness"), the resulting depth maps are much less noisy and more sensical.

We present a powerful new method to decode depth from binocular vision. The key ingredients of our method are summarized as follows:

1. **CNN feature extraction.** Standard block matching algorithms (discussed above) operate strictly in pixel space. In biological vision, however, visual stimuli are first transformed into a psychological space before disparity computation is performed [6]. We simulate this initial feature extraction phase by using a CNN to extract feature maps from each of the two images before the disparity computation. Each feature holds information about a block of pixels within its receptive field, thus, we can match single features rather than blocks of features.

2. **Efficient disparity range selection.** An important component of the disparity computation task is to select the range of candidate disparities to consider. In our framework we present a theoretically sound method for adaptive disparity range selection. The method is based on normalized entropy of disparity "energies" at each pixel location.

[1] The code repository is available at `https://github.com/rfeinman/binocular-disparity`

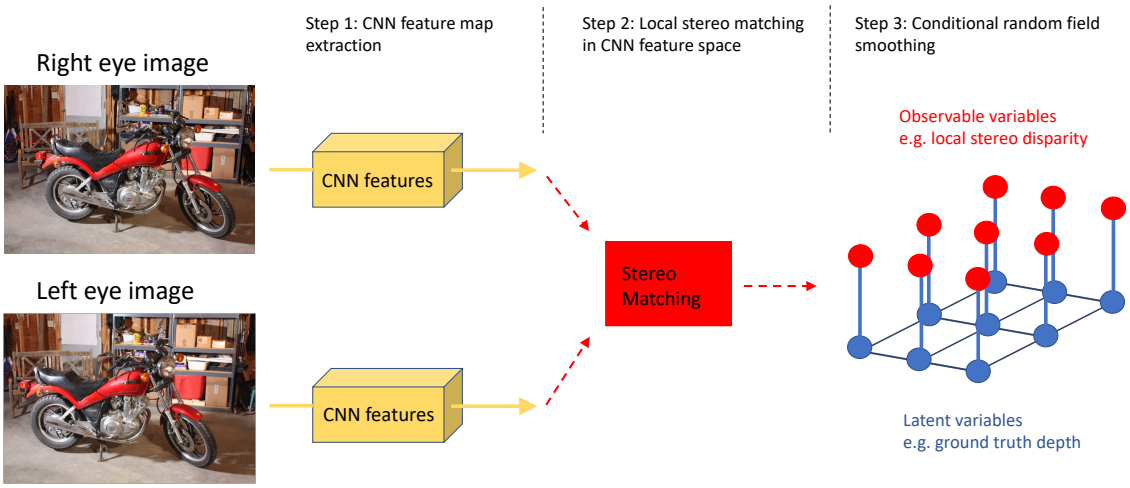


Figure 1: An overview of our method.

3. **MAP inference with CRFs.** We deploy a Conditional Random Field model to estimate the underlying disparity variables given the output of our CNN disparity model. The model makes use of a novel binary potential function designed specifically for our task. A maximum a-posteriori (MAP) estimate of the CRF latent variables is inferred via two methods: 1) greedy gradient-based methods and 2) loopy belief propagation.

Importantly, our method requires **zero labeled data.** We provide a function that maps left-right images to disparity values without requiring any training pairs of the two variables. The human brain is able to perceive depth without the use of any labelled training examples; the inspiration for this work was to see if we could do the same.

2 Method

The overarching framework of our algorithm is depicted in Fig. 1. The left and right images are first each processed independently through a CNN to obtain a feature map for each image. We use the VGG16 network, which has been trained on the ImageNet classification task. Then, local stereo matching is performed on the output feature maps to determine binocular disparity values for each pixel location.

2.1 CNN Feature Extraction

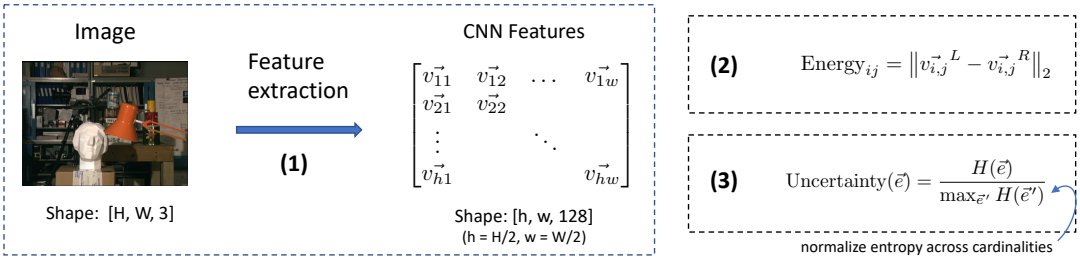
Rather than compute disparity with raw pixels, we first transform the pixels into a suitable feature space. In biological vision, visual stimuli are first transformed into a psychological space before disparity computation is performed [6]. Information from the left and right eyes is initially processed along two independent channels through the retina and LGN. During this phase, features are extracted from the left and right visual stimuli and initial representations are formed. These independent channels then converge in brain area V1, where neurons are believed to perform disparity computation [4].

Mathematically, image pre-processing is important because it extracts the important information from the received signal and removes some of the non-informative variance. A great deal of research has gone into determining the properties of an ideal image representation [9, 1]. Here, we use the 5th layer output of the VGG16 convolutional neural network, which has been trained to perform object recognition on the ImageNet classification task. This layer provides a suitable feature space for disparity computation.

One additional benefit of using CNN features instead of raw pixels is that we do not need to do block matching. The features themselves each hold information about a block of pixels, so we can match individual locations of feature space rather than blocks.

The CNN processes an image of size [H, W, 3] and returns a feature map of size [H, W, 128][2]. In other words, we get an N-dimensional feature vector for each pixel location. This is depicted in Fig. 2 **(1)**. For each location in the right feature map, we search horizontally along the left feature map for the best matching location. Matching is measured by computing the euclidean distance between the left and right feature vectors. For each candidate

[2]it actually returns size [H/2, W/2, 128] but we compensate for the downsampling with bilinear interpolation.



A. Disparity Energy Computation

1. Select N, the number of disparity values to consider. Set this large for now
2. Obtain features for the L and R images via **(1)**
3. For n = 1:N, do
   1. Shift left eye feature map by n pixels
   2. Compute E_n for all <i,j> using **(2)**
4. Convert E from shape [N, h, w] to [N, H, W] via bilinear interpolation. Return E

B. Threshold Selection

1. For n = 1:N, do
   1. Set E = $E_{1:n}$ (look at first n elements of E)
   2. For each pixel location <i,j>, compute an uncertainty score for that location by applying **(3)** to the n-length vector e_{ij}. Store vector of uncertainties as U_n
2. Choose n with the fewest high-uncertainty pixels; i.e. find U_n with lowest 75th percentile uncertainty

Figure 2: CNN feature extraction. Algorithm A returns the "disparity energy" map E. This contains one HxW map for each candidate disparity value. Algorithm B looks at E and determines an appropriate disparity threshold, using normalized entropy statistics.

disparity value, we call this distance the disparity "energy" and it is computed as

$$\text{Energy}_{ij} = \left\| \vec{v_{i,j}}^{L} - \vec{v_{i,j}}^{R} \right\|_2 \tag{1}$$

A visual depiction of this whole process is given in Fig. 2. We obtain one HxW energy map for each candidate disparity value, giving us an energy array E of size $[N, H, W]$ where N is the number of disparity candidates. Rather than considering all of the disparity candidate values, we first select a disparity threshold according to the statistics of E. This adaptive range selection helps to remove noise and is a necessary component of our algorithm. Details are shown in Fig. 2 **B.**

After we select the disparity threshold, we obtain our initial disparity assignment by computing the argmin of the energy vectors at each pixel location (argmin over $[N, H, W]$ array along the N axis). This serves as our "observation" for the CRF model.

2.2 CRF Inference

The problem with recovering the disparity map just by looking at each individual pixel and finding the best match is that it ignores neighbouring/context/spatial information. We expect pixels near each other to have similar disparity, unless there is a genuine boundary. This is where CRFs are useful.

Once stereo matching is complete, the binocular disparity values b_i of each pixel location i are then passed on to our CRF for smoothing. The disparity values b_i correspond to the red "observable" variables in Step 3 of Fig. 1. We'd now like to use our CRF to decode the latent depth variables d_i for all i. The CRF is defined by the following unary and binary potential functions:

$$\phi_i = (x_i - y_i)^2 \tag{2}$$

$$\phi_{ij} = \alpha * \frac{(x_i - x_j)^2}{(x_i - x_j)^2 - \beta} \tag{3}$$

We can formulate the stereo problem in terms of the CRF as the following energy function

$$\text{Energy}(X, Y) = \sum_i \phi_i(y_i, x_i) + \sum_{<i,j>} \phi_{ij}(x_i, x_j) \tag{4}$$

$$= \text{Data Fit} + \text{Smoothness} \tag{5}$$

Here, $< i, j >$ refers to the set of all neighboring pixels i and j. The aim is to find a labeling for X (disparity map for stereo) that produces the lowest energy.

With the unary potentials, ϕ_i, we assign a penalty for deviations from the original disparity computation ("data fitting" prior). With the binary potentials, ϕ_{ij}, we assign a penalty to lack of spatial coherence ("smoothness" prior), i.e. to large differences between neighboring depth values. As shown in the Fig. 3, we use a binary potential function ϕ_{ij} that tapers off at a certain cutoff β in order to avoid oversmoothing at boundaries. With the α parameter, we control how much to weight smoothness vs. data fitting.

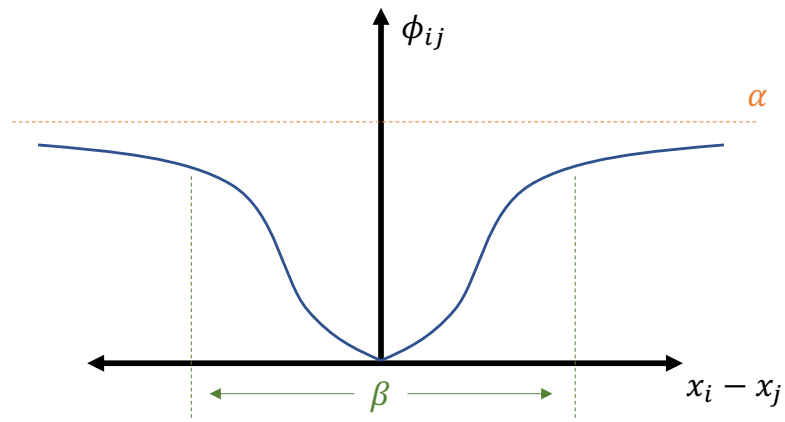


Figure 3: Smoothness Cost Function ϕ_{ij}

2.2.1 Inference Methods

We test out a variety of different methods to perform MAP inference and marginal inference in our CRF. These include belief propagation (max-sum) and gradient descent for MAP inference and sum-product inference algorithm for marginal inference.

2.2.2 Gradient Descent

We can transform the MAP objective into a more amenable optimization problem through the dual decomposition technique. Since the objective function, the Lagrangian, is continuous and convex, we may minimize it either through subgradient descent or by block coordinate descent. We select a method similar to subgradient descent.

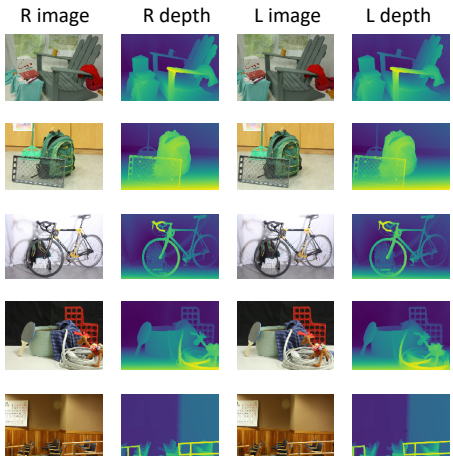
In order to get the MAP assignment from the energy function (Equation 5), we use the RMSPropOptimizer from tensorflow library which implements an adaptive learning rate method, mini-batch gradient descent algorithm. We run the algorithm for 100 iterations with a learning rate of 0.01. This method gives very similar results to Max Product Belief Propagation and is much faster but may reach local minima rather than global since the objective function is not strictly convex.

2.2.3 Max Product Message Passing

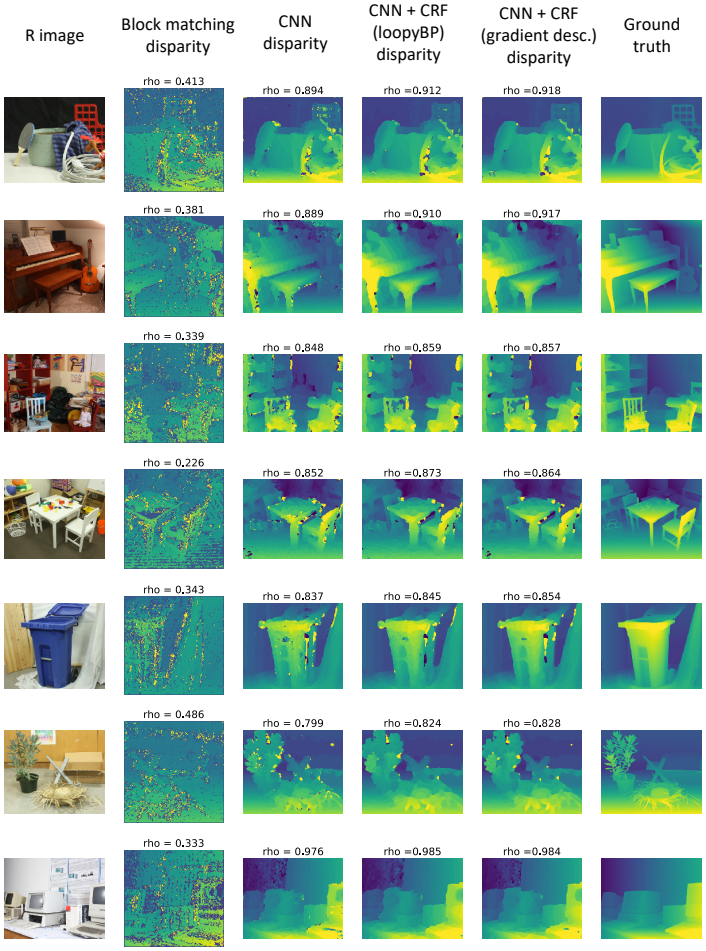
Using the above energy function (Equation 5) with its unary and binary constituents, we use pystruct library's implementation of the Max Product, loopy belief propagation to solve the inference problem and get a MAP assignment of the disparities for every pixel. We run the algorithm for 20 iterations. The number of disparities used for each image comes from the "Threshold Selection" method described earlier. Belief propagation is typically slower than Gradient Descent but the MAP estimates it produces are sometimes more optimal.

2.2.4 Sum Product Message Passing

In order to compare MAP assignment with an assignment coming from the mode of the marginal probability distribution of disparities at each pixel, we ran the Sum-Product Loopy Belief Propagation on our energy function. We used a custom implementation of the sum-product algorithm and tried the simulation for different iterations and different choices of unary and binary potential functions. Emperically, we find that this method produces results not much worse from the max product algorithm which gives the MAP assignment.



(a) Middlebury dataset



(b) Algorithm results

Figure 4: Middlebury stereo dataset, and the results of our disparity algorithm.

Method	**Mean Spearman Corr.**	**Mean Pearson Corr.**
Block matching	0.304 +/- 0.109	0.222 +/- 0.100
CNN	0.787 +/- 0.110	0.700 +/- 0.159
CNN + CRF (loopyBP)	0.803 +/- 0.112	0.732 +/- 0.181
CNN + CRF (gradient desc.)	0.801 +/- 0.112	0.732 +/- 0.157

Figure 5: Results on the Middlebury Dataset. Showing correlation with ground truth disparity. Spearman and Pearson correlations are used. Means and standard deviations shown.

3 Dataset

Stereo vision is one of the most heavily researched topics in computer vision and as such many standard test images and benchmarks are available with different resolutions, scene complexity, realism, and accuracy of ground truth.

We used the datasets given here: Stereo Data. Each of the 23 figures has a left and a right image along with the corresponding "ground truth" disparity matrices. These datasets are generated by techniques mentioned in [7]. They combine disparity estimates from multiple projector positions to achieve a disparity accuracy of 0.2 pixels on most observed surfaces, including in half occluded regions giving us very accurate training data. Some examples from the dataset are shown in Figure 4(a)

Since in order to calculate disparity, we assume the left image is shifted from its initial "right" position, we use the right image ground truth to judge how good our final disparity calculations are.

4 Results

We ran our algorithm on each of the 23 datasets and the collated results are presented in Fig. 4 and Fig. 5.

Figure 4(b) shows the original "right" image, followed by the disparity matrix from the Block Matching algorithm. The CNN disparity matrix follows after which we show CNN + CRF (loopyBP) and CNN + CRF (Gradient Descent) disparity matrices. The right most column shows the ground truth for the "right" image.

In 4(b) ρ stands for the Spearman Rank Correlation Coefficient values between the current disparity when compared to the "right" image's ground truth. As can be seen from Figure 4(b) and Table 5, ρ values for CNN disparity is higher than block matching disparity. Also smoothing through CRF by either loopy BP or Gradient Descent increases ρ values and both methods have more of less similar effect.

Apart from mean ρ values, Table 5 also shows values for the Pearson Correlation. As expected, CNN disparities perform much better than the disparities obtained from simple Block Matching. CRF smoothing increases Pearson Coefficient values as well.

5 Code Demo

```
from disparity import cnn, mrf, util

# Create a function to load your left and right image.
image_left, image_right = load_images()
height, width, _ = image_left.shape

# Compute disparity energies for a left-right image pair.
# This returns an array of size (height, width, numDisparities)
energies = cnn.compute_energies(image_left, image_right, numDisparities=120)

# Select an optimal disparity threshold based on energy entropy
threshold = util.select_disparity_threshold(energies)
energies = energies[:,:,:threshold]

# Initialize MRF loopy belief propagation model
smoother = mrf.LoopyBP(height, width, num_beliefs=threshold)

# Perform MAP inference with loopy BP (max-product message passing)
disparity = smoother.decode_MAP(energies, iterations=20)
```

Figure 6: A demonstration of our code repository.

6 Conclusion & Future Work

In this project we have developed a fast and highly effective computational framework for computing depth from stereo images. Our model achieves high performance with zero labelled training data, and it improves considerably over baseline block matching methods. We are releasing an open-source code repository with complete, easy-to-use implementations of our algorithms and thorough documentation.

In future work, we'd like to think about how to handle occlusion. Occlusion is one of the biggest challenges in stereo matching. For two-frame stereo, occluded pixels are only visible in one image. In certain applications such as 3D modeling and view interpolation, a stereo algorithm should not only estimate accurate depths at visible areas, but it should also provide reasonable guesses of depths at occluded areas. As shown in [10], we can augment our stereo matching algorithm to handle occlusions in either the left or the right image. The visibility constraint requires

occlusion in one image and disparity in the other to be consistent. We can embed the visibility constraint within our energy minimization framework, resulting in a symmetric stereo model that treats left and right images equally. Belief Propagation can then be used to minimize the energy of both stereo disparity as well as occlusion.

Since depth discontinuities occur on boundaries of segmented regions [12], we also can incorporate segmentation results into our basic stereo model as soft constraints (priors) under a probabilistic framework [11].

References

[1] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.

[2] G. Egnal. Mutual information as a stereo correspondence measure. *Computer and information science*, 2000.

[3] H. Hirschmuller, P.R. Innocent, and J.M. Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, 47(1/2/3):229–246, 2002.

[4] D. Marr. *Vision*. MIT Press, 1982.

[5] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283—287, 1976.

[6] N. Qian. Computing stereo disparity and motion with known binocular cell properties. *Neural Computation*, 6(3):390–404, 1994.

[7] 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 *GCPR*, pages 31–42, 2014.

[8] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *CVPR*, pages 1–8, 2007.

[9] E. P. Simoncelli and W. T. Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *IEEE International Conference on Image Processing*, pages 1–8, 2007.

[10] J. Sun, Y. Li, S.B. Kang, and H.Y. Shum. Symmetric stereo matching for occlusion handling. In *CVPR*, pages 399–406, 2005.

[11] J. Sun, N.N. Zheng, and H.Y. Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[12] H. Tao, H.S. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *ICCV*, pages 532–539, 2001.

[13] K.J. Yoon and I.S. Kweon. Adaptive support-weight approach for correspondence search. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 28(4):650–656, 2006.