Stereopsis with Convolutional Neural Networks and Conditional Random Fields

Reuben Feinman, Ambuj Ojha

Abstract

- The brain uses the relative difference between signals from left and right eyes (binocular disparity) to perceive depth
- We present a powerful computational framework for decoding depth from stereo vision using Convolutional Neural Networks and Conditional Random Fields
- Our method requires **zero labeled training data**
- We are releasing an open-source code repository with highly efficient, modularized Python implementations of the disparity computation algorithm[1]

[1]Python package is available for download at https://github.com/rfeinman/binocular-disparity

Model Overview

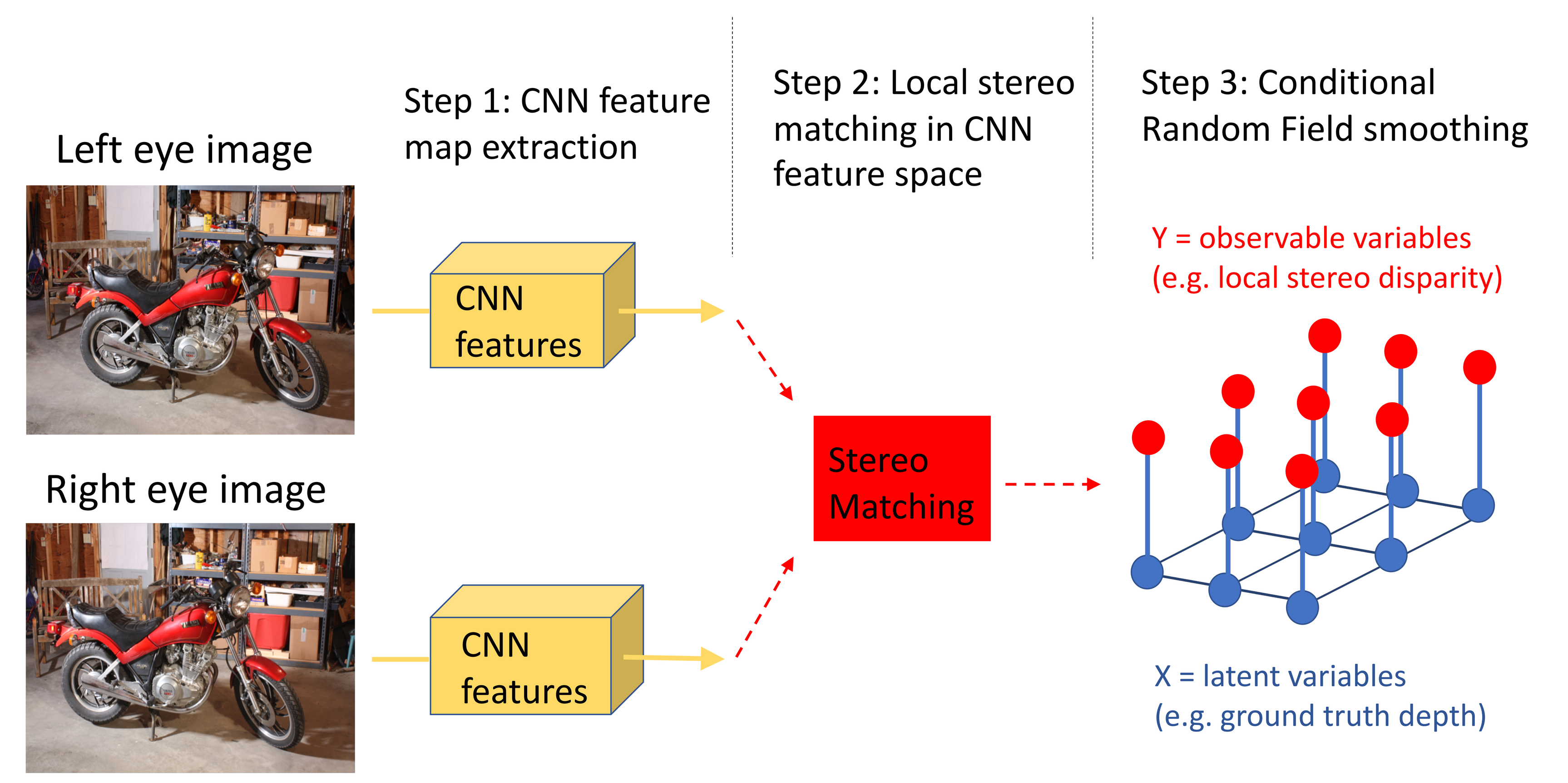
- Standard block matching estimates disparity by searching for matching blocks of pixels in the 'left' and 'right' images. But:
  - Raw pixels are noisy; uninformative variance
  - Standard block matching neglects local dependencies

Convolutional Neural Network (CNN):

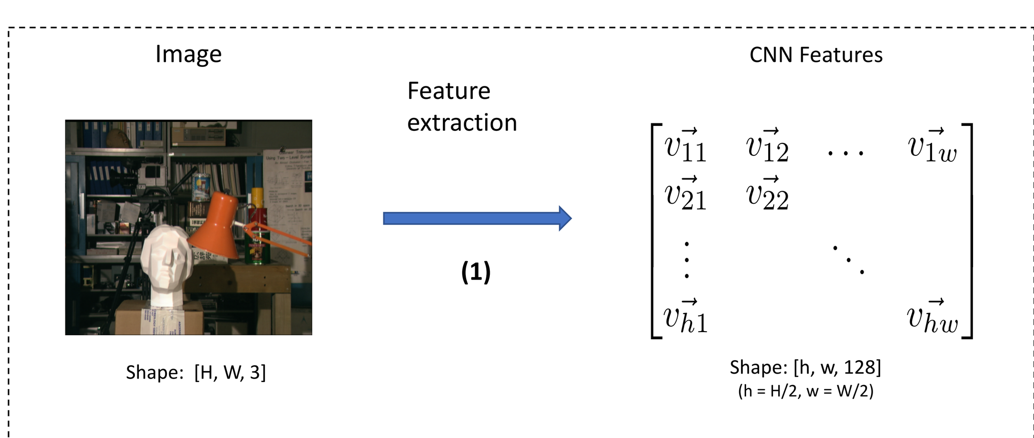
- In biological vision, stimuli are transformed into a psychological space before disparity computation
- Important to first decompose the signal, e.g. into orientation and frequency subbands.
- We use a pre-trained CNN to extract feature maps from the two images, then perform stereo matching with the features

Conditional Random Field (CRF):

- The disparities are expected to be piecewise smooth since most surfaces are smooth.
- MAP inference options: 1) loopy BP 2) greedy gradient-descent



Convolutional Neural Network



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

(3) $\text{Uncertainty}(\vec{e}) = \frac{H(\vec{e})}{\max_{\vec{e}'} H(\vec{e}')}$

normalize entropy across cardinalities

A. Disparity Energy Computation

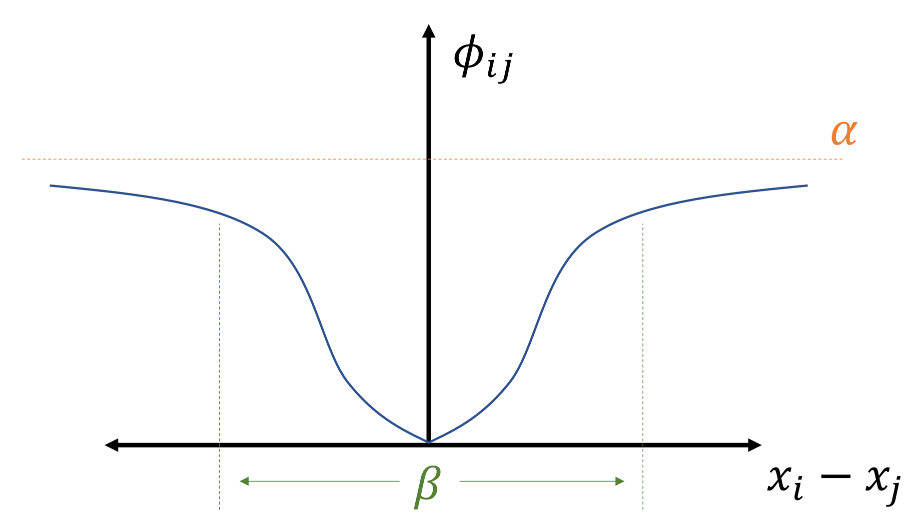
1. Select N, the number of disparity values to consider. Set this large for now
2. Obtain features for L and R images via **(1)**
3. For n = 1:N, do
   1. Shift left eye feature map by n pixels
   2. Compute E_n using eq. **(2)**
4. Convert E from shape [N, h, w] to [N, H, W] via bilinear interpolation. 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 eq. **(3)** to the n-length vector e_{ij}. Store uncertainties as U_n
2. Select n with the least high-uncertainty pixels; i.e., choose U_n with lowest 75^{th} percentile uncertainty

Return $\mathbf{E}_{1:threshold}$. These are initial beliefs for each pixel, i.e. disparity probabilities

Conditional Random Field



Avoids over-smoothing at surface boundaries!

Y: observation X: latent variable

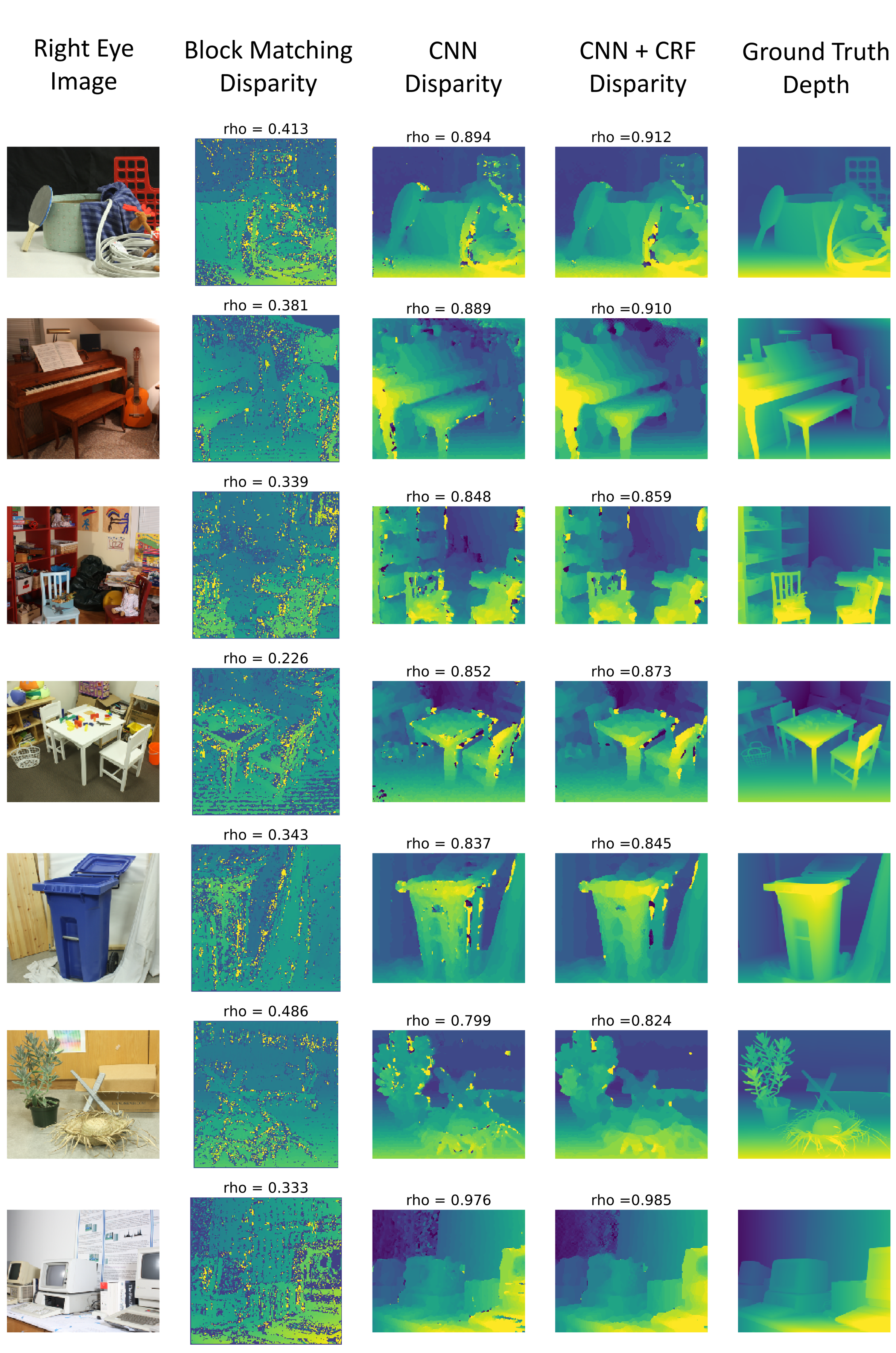
$\phi_i = (x_i - y_i)^2$ Unary potentials

$\phi_{ij} = \alpha * \frac{(x_i - x_j)^2}{(x_i - x_j)^2 + \beta}$ Binary potentials

Inference Algorithms:

1. Loopy belief propagation (20 iterations)
   - MAP: max-product message passing
   - Marginal modes: sum-product message passing
   - Slow algorithm, best results
2. Greedy stochastic gradient descent (100 iterations)
   - Very fast, slightly worse results

Results: CNN + Loopy BP



*rho values show the Spearman correlation with ground truth

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	0.801 +/ 0.112	0.732 +/- 0.157

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)
```

Future Work

- Compare Belief Propagation to other Inference Methods e.g. Gibbs Sampling, Variational Inference
- Augment our stereo matching algorithm to handle occlusions in either the left or the right image
- Incorporate image segmentation results into our basic stereo model as soft constraints (priors) under a probabilistic framework