Computer Vision

Lecture 4 – Stereo Reconstruction

Kumar Bipin

BE, MS, PhD (MMMTU, IISc, IIIT-Hyderabad)

Robotics, Computer Vision, Deep Learning, Machine Learning, System Software

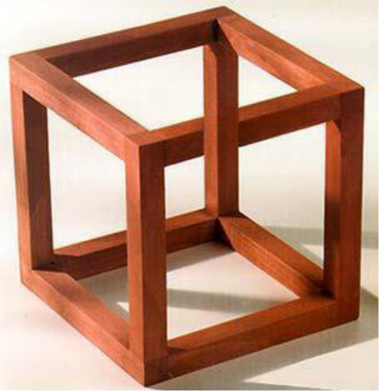
Agenda

4.1
Preliminaries

How to recover 3D from an image?

In general, this is an ill-posed problem, but there are several cues:

- Occlusion
- Parallax
- Perspective
- Accomodation
- Stereopsis



How to recover 3D from an image?

In general, this is an ill-posed problem, but there are several cues:

- Occlusion
- Parallax
- Perspective
- Accomodation
- Stereopsis

How to recover 3D from an image?

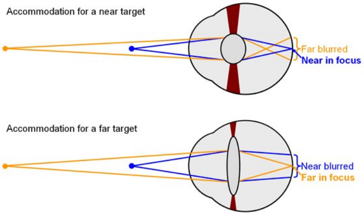
In general, this is an ill-posed problem, but there are several cues:

- Occlusion
- Parallax
- Perspective
- Accomodation
- Stereopsis

How to recover 3D from an image?

In general, this is an ill-posed problem, but there are several cues:

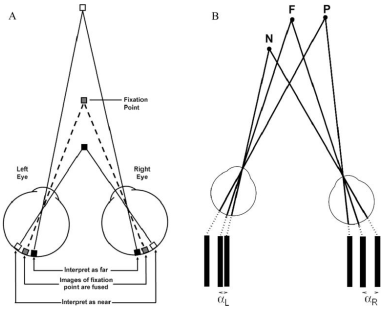
- Occlusion
- Parallax
- Perspective
- Accomodation
- Stereopsis



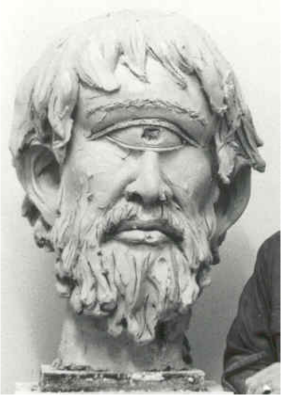
How to recover 3D from an image?

In general, this is an ill-posed problem, but there are several cues:

- Occlusion
- Parallax
- Perspective
- Accomodation
- Stereopsis



Why Binocular Stereopsis?



Cyclope

Leela

Cyclopes had only one eye after making a deal with Hades, god of the underworld, in which they traded one eye for the ability to see the future and predict the day they would die.

Why Binocular Stereopsis?

Stalk-Eyed Fly

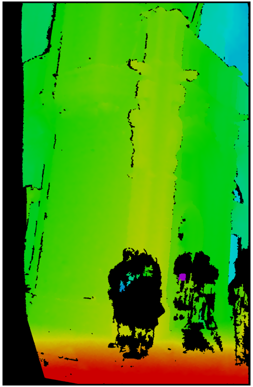
Stereoscopic Rangefinder

A stereoscopic rangefinder is an optical device that measures distance from the observer to a target, using the observer's capability of binocular vision. The prisms are rotated until the mark appears to overlap in the operator's combined view. The range is derived from the rotation.

Two-View Stereo Matching

Left Image

Right Image



Disparity Map

- **Goal:** Recover disparity (=inverse depth) for every pixel from two input images

Geiger, Roser and Urtasun: Efficient Large-Scale Stereo Matching. ACCV, 2010.

Two-View Stereo Matching

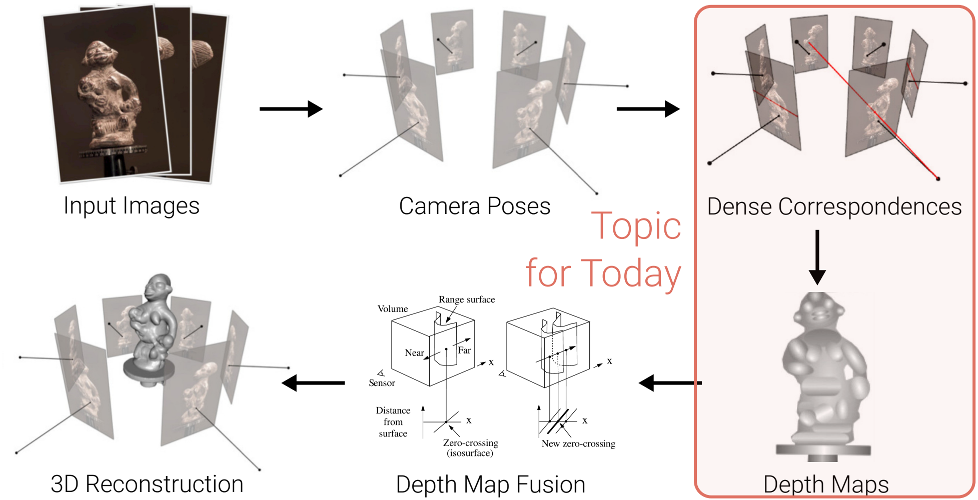
Task:

- Construct a **dense** 3D model from 2 images of a static scene

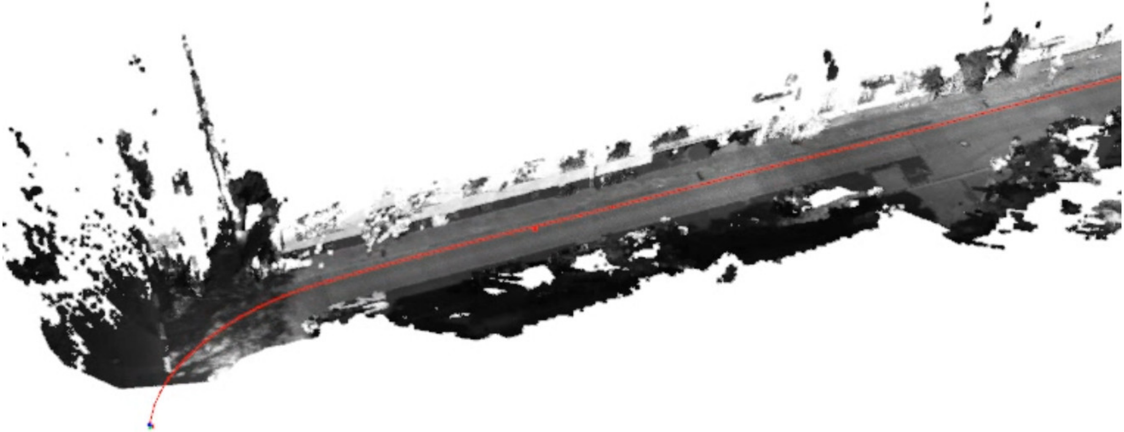
Pipeline:

1. **Calibrate cameras** intrinsically and extrinsically (Lecture 3.1)
2. **Rectify images** given the calibration
3. **Compute disparity map** for reference image
4. **Remove outliers** using consistency/occlusion test
5. **Obtain depth** from disparity using camera calibration
6. **Construct 3D model**, e.g., via volumetric fusing and meshing (Lecture 8.4)

3D Reconstruction Pipeline



3D Model



Geiger, Ziegler and Stiller: StereoScan: Dense 3d reconstruction in real-time. IV, 2011.

Epipolar Geometry

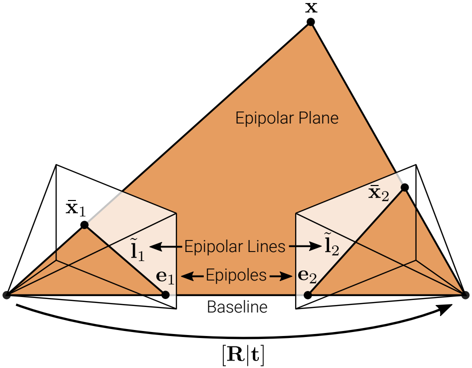
Epipolar Geometry

Epipolar Geometry:

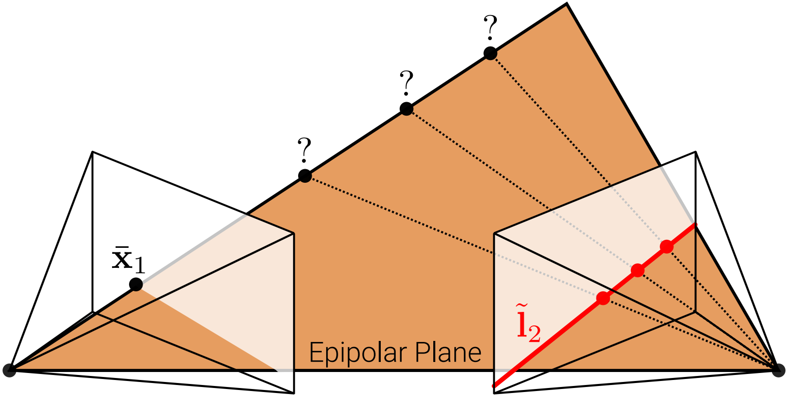
- 3D point $\mathbf{x}$ is projected to both images as $\bar{\mathbf{x}}_1 \propto \mathbf{K}_1\,\mathbf{x}$ and $\bar{\mathbf{x}}_2 \propto \mathbf{K}_2\,(\mathbf{R}\mathbf{x} + \mathbf{t})$
- Image correspondences are related by

$$\tilde{\mathbf{x}}_2^\top \tilde{\mathbf{E}}\,\tilde{\mathbf{x}}_1 = 0 \quad \text{with} \quad \tilde{\mathbf{x}}_i = \mathbf{K}_i^{-1}\,\bar{\mathbf{x}}_i$$

- The correspondence of pixel $\bar{\mathbf{x}}_1$ is located on the **epipolar line** $\tilde{\mathbf{l}}_2 = \tilde{\mathbf{E}}\,\tilde{\mathbf{x}}_1$
- The correspondence of pixel $\bar{\mathbf{x}}_2$ is located on the **epipolar line** $\tilde{\mathbf{l}}_1 = \tilde{\mathbf{E}}^\top\,\tilde{\mathbf{x}}_2$
- Epipolar lines intersect at the **epipoles**



Epipolar Geometry

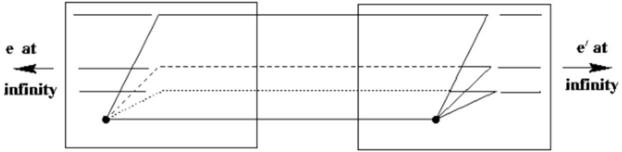


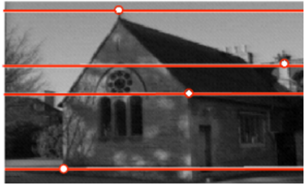
- A point $\bar{\mathbf{x}}_1$ in the left image must be located on the epipolar line $\tilde{\mathbf{l}}_2$
- This reduces correspondence search to a (much simpler) **1D problem**
- For VGA images: $\sim$640 instead of $\sim$300k hypotheses (factor 480 less)

Image Rectification

Image Rectification

What if both cameras face exactly the same direction?





- Image planes are co-planar $\Rightarrow$ Epipoles at infinity, epipolar lines parallel
- Correspondences search along **horizontal scanlines** (simplifies implementation)

Image Rectification

What if the images are not in the required setup?

- There is a trick: We can rewarp them through **rotation,** mapping both image planes to a common plane parallel to the baseline, this is called **rectification**
- For this rotation around the camera center, the 3D structure must not be known

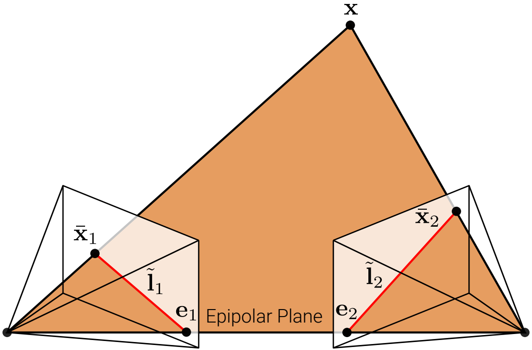


Image Rectification

What if the images are not in the required setup?

- There is a trick: We can rewarp them through **rotation,** mapping both image planes to a common plane parallel to the baseline, this is called **rectification**
- For this rotation around the camera center, the 3D structure must not be known

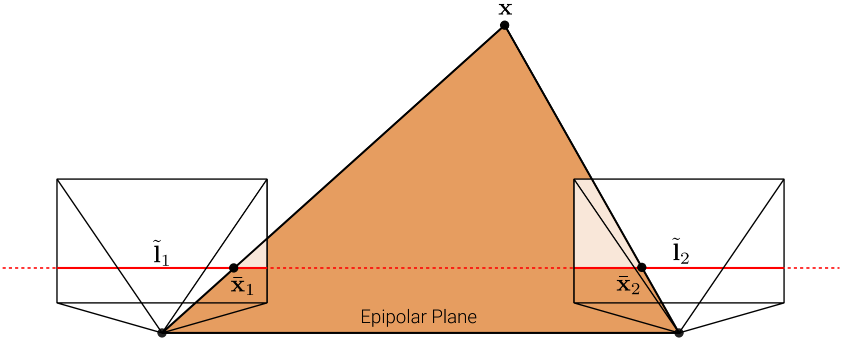


Image Rectification

How can we make epipolar lines horizontal?

Let $\mathbf{K}_1 = \mathbf{K}_2 = \mathbf{R} = \mathbf{I}$ and $\mathbf{t} = (t, 0, 0)^\top$. In this case, the essential matrix is given by:

$$\tilde{\mathbf{E}} = [\mathbf{t}]_\times \mathbf{R} = [\mathbf{t}]_\times = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t \\ 0 & t & 0 \end{bmatrix}$$

As $\mathbf{K}_i = \mathbf{I}$, two corresponding points $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ are related by the **epipolar constraint:**

$$\bar{\mathbf{x}}_2^\top \tilde{\mathbf{E}} \, \bar{\mathbf{x}}_1 = \bar{\mathbf{x}}_2^\top \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t \\ 0 & t & 0 \end{bmatrix} \bar{\mathbf{x}}_1 = \bar{\mathbf{x}}_2^\top \begin{pmatrix} 0 \\ -t \\ t\, y_1 \end{pmatrix} = t\, y_1 - t\, y_2 = 0$$

where y_i denotes the second element of $\bar{\mathbf{x}}_i$

Thus, $y_1 = y_2$, i.e., the two images of 3D point $\mathbf{x}$ are on the **same horizontal line.**

Image Rectification

In order to rectify an image, we calculate a **rectifying rotation** $\mathbf{R}_{\text{rect}} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)^\top$, with $\mathbf{r}_1 = \mathbf{t}/\|\mathbf{t}\|_2$, $\mathbf{r}_2 = [(0,0,1)^\top]_\times \mathbf{r}_1$ and $\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$. As the epipole in the first image is in the direction of $\mathbf{r}_1$, it is easy to see that the rotated epipole is ideal: $\mathbf{R}_{\text{rect}}\,\mathbf{r}_1 = (1,0,0)^\top$. Thus, applying $\mathbf{R}_{\text{rect}}$ to the first camera leads to parallel and **horizontal epipolar lines.**

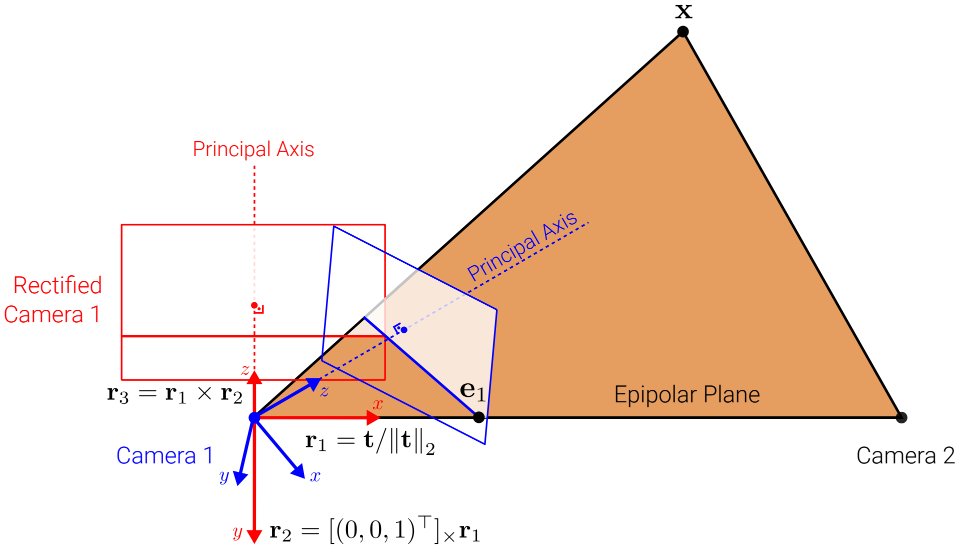
Rectification Algorithm:

1. Estimate $\tilde{\mathbf{E}}$, decompose into $\mathbf{t}$ and $\mathbf{R}$, and construct $\mathbf{R}_{rect}$ as above
2. Warp pixels in the first image as follows: $\tilde{\mathbf{x}}_1' = \mathbf{K}\,\mathbf{R}_{\text{rect}}\,\mathbf{K}_1^{-1}\,\bar{\mathbf{x}}_1$
3. Warp pixels in the second image as follows: $\tilde{\mathbf{x}}_2' = \mathbf{K}\,\mathbf{R}\,\mathbf{R}_{\text{rect}}\,\mathbf{K}_2^{-1}\,\bar{\mathbf{x}}_2$

Remarks:

- $\mathbf{K}$ is a shared projection matrix that can be chosen arbitrarily (e.g., $\mathbf{K} = \mathbf{K}_1$)
- In practice, the inverse transformation is used for warping (i.e., query the source)

Calculating the Rectifying Rotation Matrix



Rectification Example

- Correspondences are located on the **same image row** as the query point

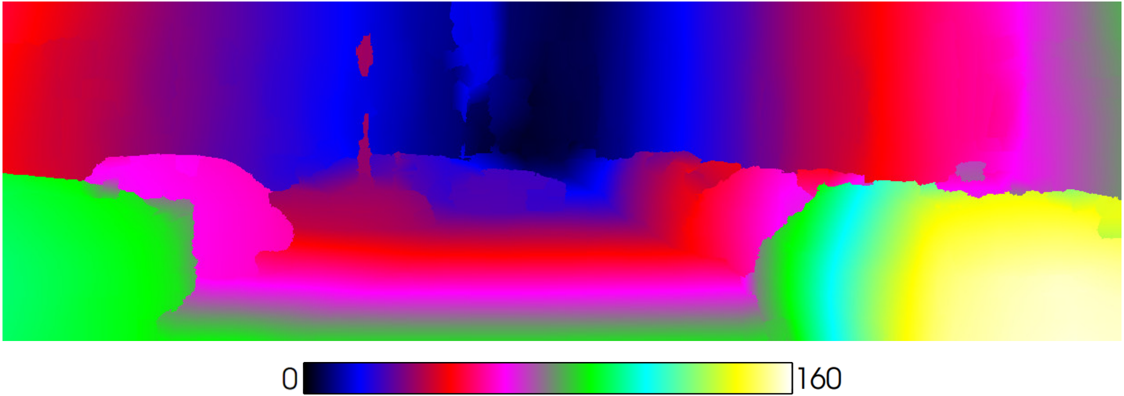
Disparity Estimation Example

- The relative horizontal displacement is called **disparity** (here: 0-160 px to the left)
- Disparity is anti-proportional to depth (far points move less than closer points)

Disparity Estimation Example

- The relative horizontal displacement is called **disparity** (here: 0-160 px to the left)
- Disparity is anti-proportional to depth (far points move less than closer points)

Disparity Estimation Example



- The relative horizontal displacement is called **disparity** (here: 0-160 px to the left)
- Disparity is anti-proportional to depth (far points move less than closer points)

Disparity to Depth

Disparity to Depth

How can we recover depth from the estimated disparity?

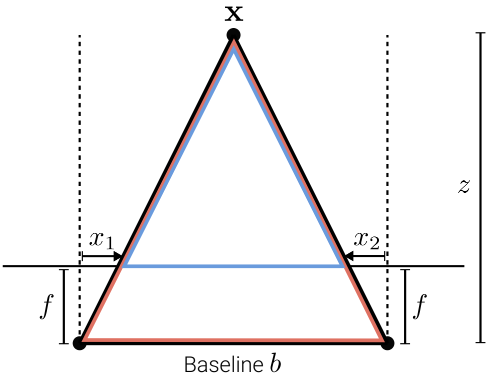
- The left and right ray must intersect as both lie in the epipolar plane
- Assuming disparity $d = x_1 - x_2$, we have:

$$\frac{z-f}{b-d} = \frac{z}{b}$$

$$zb - fb = zb - zd$$

$$z = \frac{fb}{d}$$

- The depth Δz error grows quadratically with the depth z (left as exercise)



4.2
Block Matching

Correspondence Ambiguity

How to determine if two image points correspond?

- Assume that they look “similar”, but what does similar mean?
- A single pixel does not reveal the local structure (ambiguities)
- Therefore, we should compare a small image region/patch!
- But even then the task is difficult:

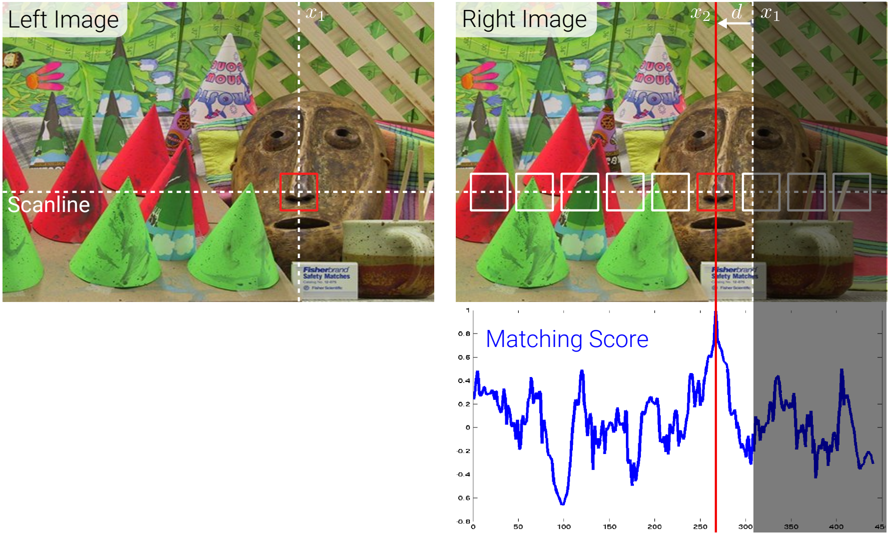
Similarity Metrics

Scharstein and Szeliski: High-Accuracy Stereo Depth Maps Using Structured Light. CVPR, 2003.

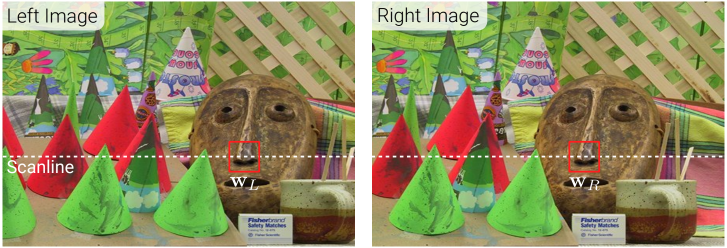
Similarity Metrics

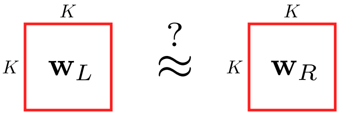
Scharstein and Szeliski: High-Accuracy Stereo Depth Maps Using Structured Light. CVPR, 2003.

Similarity Metrics



Similarity Metrics





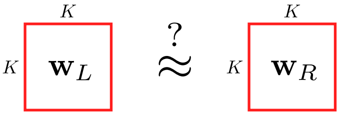
- Consider two $K \times K$ windows of pixels flattened to vectors $\mathbf{w}_L, \mathbf{w}_R \in \mathbb{R}^{K^2}$
- Zero Normalized Cross-Correlation (ZNCC), $\bar{w}$ denotes the mean of the vector $\mathbf{w}$:

$$\text{ZNCC}(x, y, d) = \frac{(\mathbf{w}_L(x,y) - \bar{w}_L(x,y))^T \, (\mathbf{w}_R(x-d,y) - \bar{w}_R(x-d,y))}{\|\mathbf{w}_L(x,y) - \bar{w}_L(x,y)\|_2 \, \|\mathbf{w}_R(x-d,y) - \bar{w}_R(x-d,y)\|_2}$$

Numerous other similarity metrics exist, see, e.g., Szeliski, Chapter 12.3.

Hirschmüller and Scharstein: Evaluation of Stereo Matching Costs on Images with Radiometric Differences. TPAMI, 2009.

Similarity Metrics



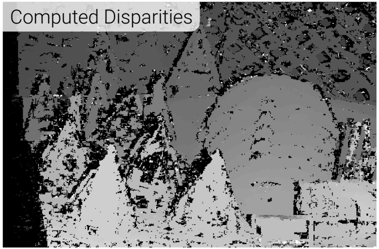
- Consider two $K \times K$ windows of pixels flattened to vectors $\mathbf{w}_L, \mathbf{w}_R \in \mathbb{R}^{K^2}$
- Sum of squared differences (SSD):

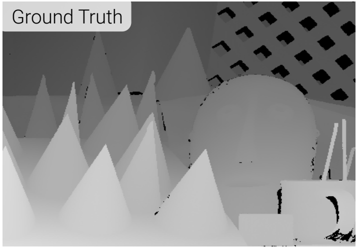
$$\text{SSD}(x, y, d) = \|\mathbf{w}_L(x, y) - \mathbf{w}_R(x - d, y)\|_2^2$$

Numerous other similarity metrics exist, see, e.g., Szeliski, Chapter 12.3.

Hirschmüller and Scharstein: Evaluation of Stereo Matching Costs on Images with Radiometric Differences. TPAMI, 2009.

Block Matching

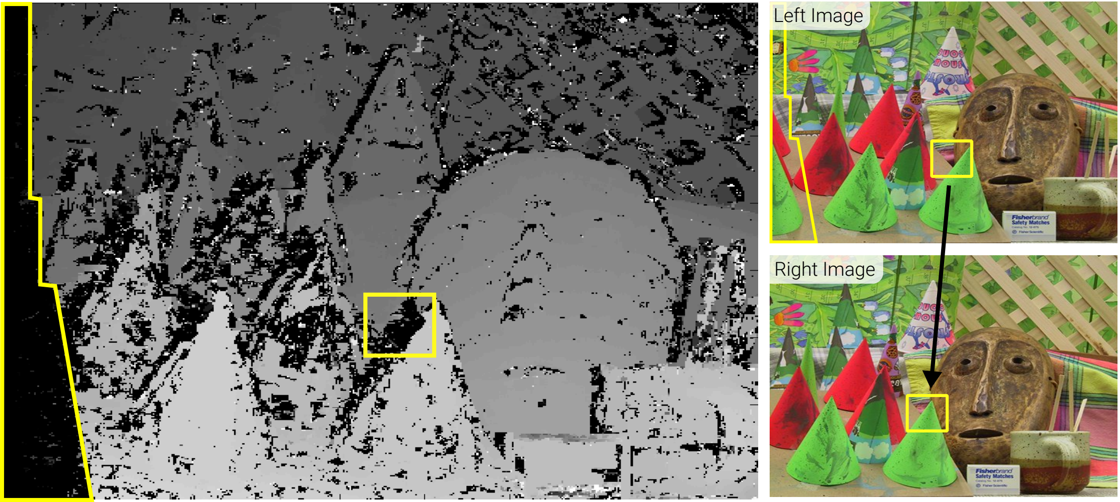




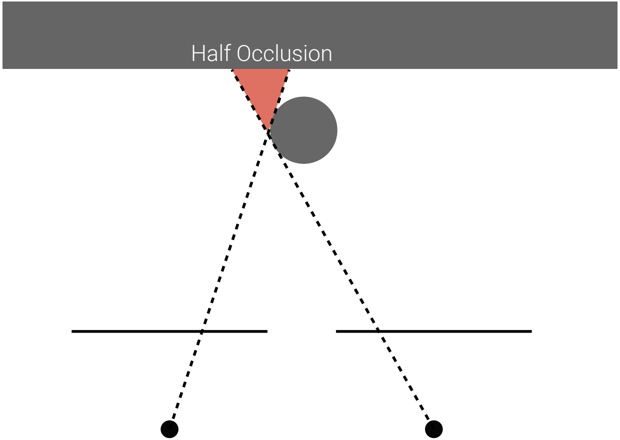
Block Matching:

- Choose disparity range $[0, D]$
- For all pixels $\mathbf{x} = (x, y)$ compute the best disparity $\Rightarrow$ winner-takes-all (WTA)
- Do this for both images and apply left-right consistency check to remove outliers

Block Matching: Half Occlusions



Block Matching: Half Occlusions



- The red area is visible in the left image, but not in the right image

Block Matching: Assumption Violations

Assumption Violations:

- Block matching assumes that all pixels inside the window are displaced by d
- This is called the **fronto-parallel assumption** which is often invalid
- **Slanted surfaces** deform perspectively when the viewpoint changes

Block Matching: Assumption Violations

Assumption Violations:

- Block matching assumes that all pixels inside the window are displaced by d
- This is called the **fronto-parallel assumption** which is often invalid
- **Slanted surfaces** deform perspectively when the viewpoint changes

Block Matching: Assumption Violations

Assumption Violations:

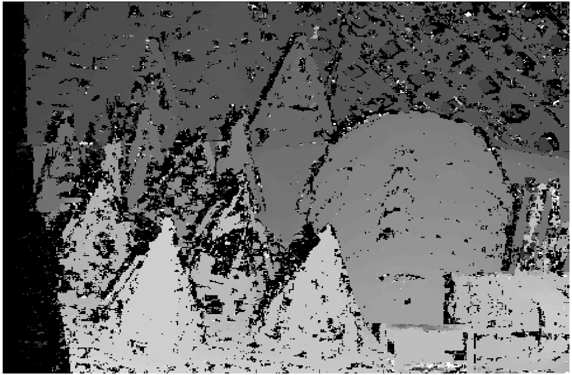
- Block matching assumes that all pixels inside the window are displaced by d
- This is called the **fronto-parallel assumption** which is often invalid
- The window content changes differently at **disparity discontinuities**

Block Matching: Assumption Violations

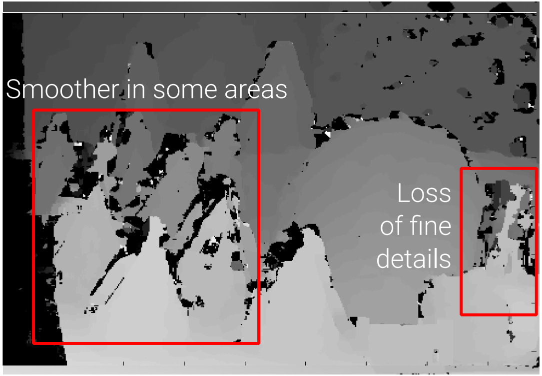
Assumption Violations:

- Block matching assumes that all pixels inside the window are displaced by d
- This is called the **fronto-parallel assumption** which is often invalid
- The window content changes differently at **disparity discontinuities**

Effect of Window Size



Window Size: 5x5 Pixels

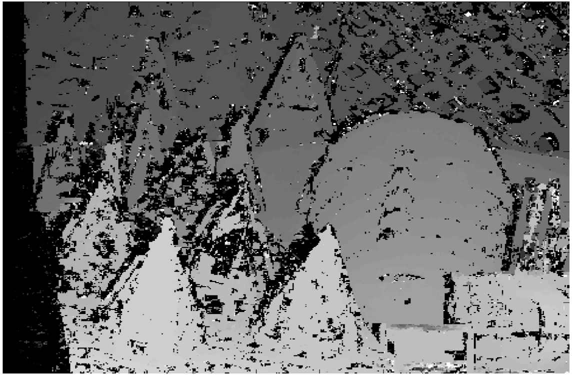


Window Size: 11x11 Pixels

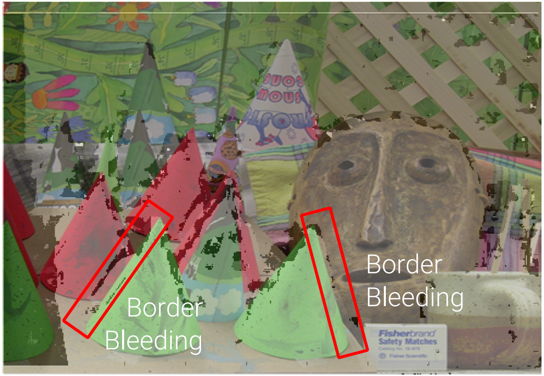
Tradeoff:

- Small windows lead to matching ambiguities and noise in the disparity maps
- Larger windows lead to smoother results, but loss of details and border bleeding

Effect of Window Size



Window Size: 5x5 Pixels

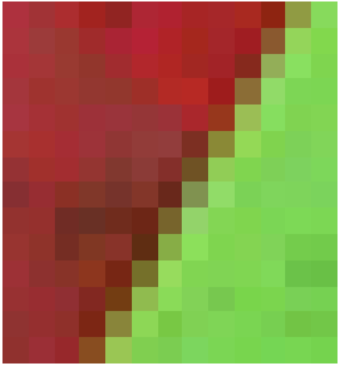
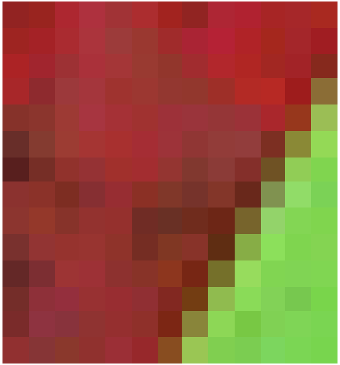


Window Size: 11x11 Pixels

Tradeoff:

- Small windows lead to matching ambiguities and noise in the disparity maps
- Larger windows lead to smoother results, but loss of details and border bleeding

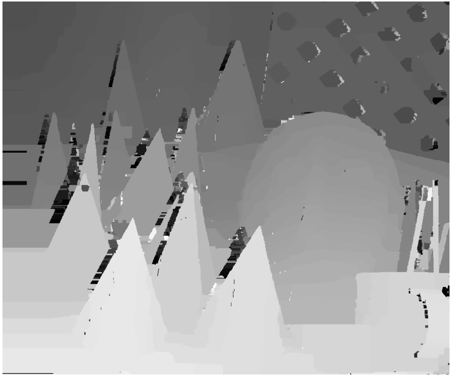
Effect of Window Size

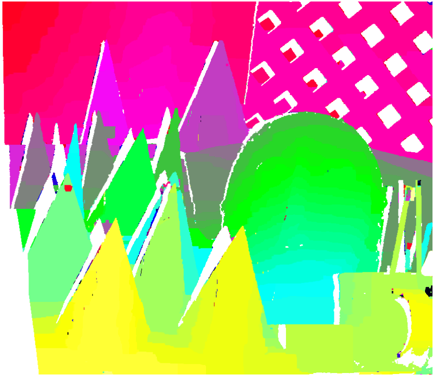
 

Border Bleeding:

- The left and right patch lead to the same disparity due to the dominant edge
- The red background does not provide enough gradients to overcome this

Left-Right Consistency Test





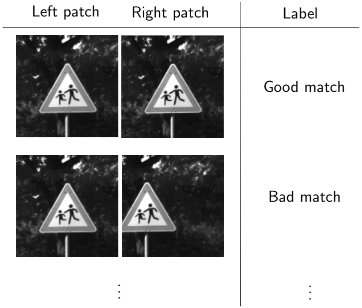
Left-Right Consistency Test:

- Outliers and half-occlusions can be detected via a left-right consistency test
- Compute disparity map for both images, verify if they map to each other (cycle)

4.3
Siamese Networks

Siamese Networks for Stereo Matching

- **Hand crafted features** and similarity metrics do not take into consideration all relevant geometric and radiometric invariances or occlusion patterns
- The world is too complex to specify this by hand
- Matching cost computation can be treated as **image classification problem:**

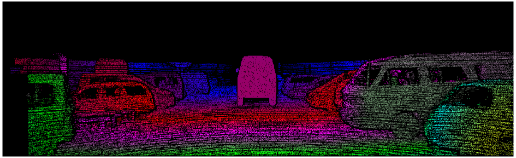


Zbontar and LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. JMLR, 2016.

Siamese Networks for Stereo Matching

Method Overview:

- Train CNN patch-wise based on images with ground truth disparity maps (e.g., KITTI Stereo 2015: `http://www.cvlibs.net/datasets/kitti/`)

- Calculate features for each of the two images using learned model
- Correlate features between both images (dot product)
- Find maximum for every pixel (winner-takes-all) or run global optimization

Zbontar and LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. JMLR, 2016.

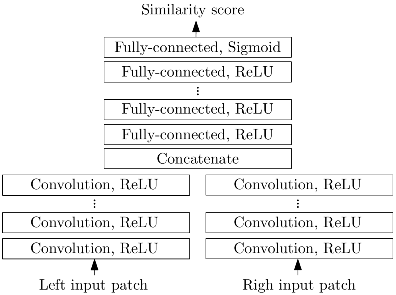
Siamese Network Architecture

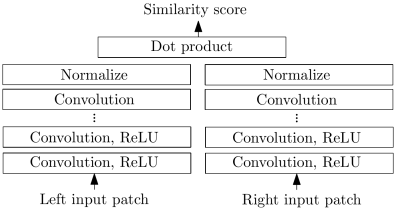
Learned Similarity:

- Learn features & sim. metric
- Potentially more expressive
- Slow (WxHxD MLP evaluations)

Cosine Similarity:

- Learn features & apply dot-product
- Features must do the heavy lifting
- Fast matching (no network eval.)





Zbontar and LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. JMLR, 2016.

Training

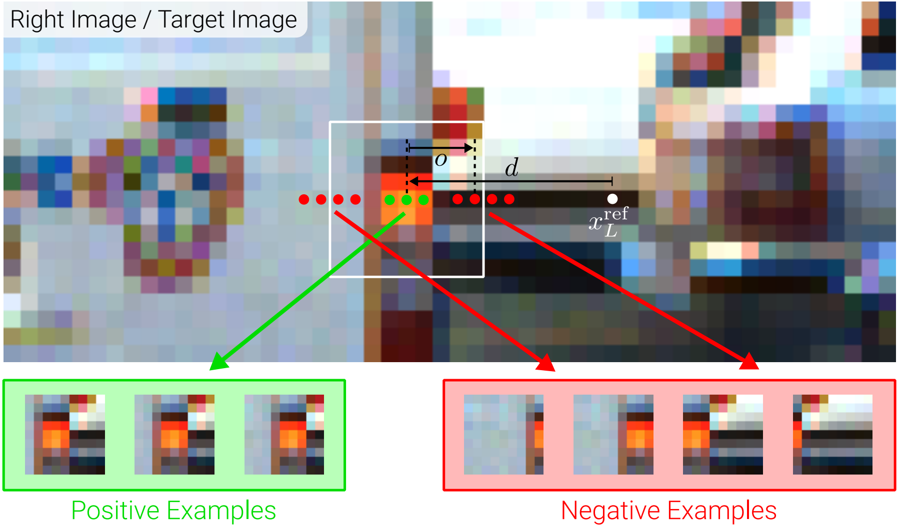
The training set is composed of patch triplets:

$$\left(\mathbf{w}_L(\mathbf{x}_L^{\text{ref}}),\ \mathbf{w}_R(\mathbf{x}_R^{\text{neg}}),\ \mathbf{w}_R(\mathbf{x}_R^{\text{pos}})\right)$$

- $\mathbf{w}_L(\mathbf{x}_L)$ is an image patch from the left image centered at $\mathbf{x}_L = (x_L, y_L)$
- $\mathbf{w}_R(\mathbf{x}_R)$ is an image patch from the right image centered at $\mathbf{x}_R = (x_R, y_R)$
- Negative example: $\mathbf{x}_R^{\text{neg}} = (x_L^{\text{ref}} - d + o_{\text{neg}}, y_L^{\text{ref}})$
  - Offsets o_{neg} drawn from $\mathcal{U}(\{-N_{\text{hi}}, \dots, -N_{\text{lo}}, N_{\text{lo}}, \dots, N_{\text{hi}}\})$
- Positive example: $\mathbf{x}_R^{\text{pos}} = (x_L^{\text{ref}} - d + o_{\text{pos}}, y_L^{\text{ref}})$
  - Offsets o_{pos} drawn from $\mathcal{U}(\{-P_{\text{hi}}, \dots, P_{\text{hi}}\})$
- Here, d denotes the true disparity for a pixel (provided as ground truth)
- Typically, $P_{\text{hi}} = 1$, $N_{\text{low}} = 3$ and $N_{\text{hi}} = 6 \Rightarrow$ hard negative mining

Zbontar and LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. JMLR, 2016.

Training



Zbontar and LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. JMLR, 2016.

Loss Function

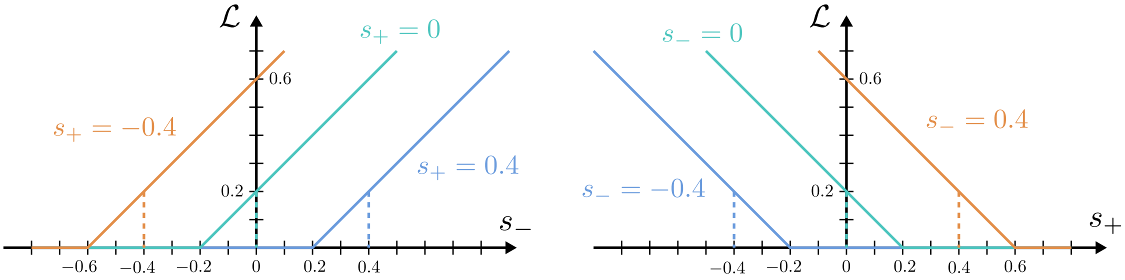
Hinge Loss:

$$\mathcal{L} = \max(0, m + s_- - s_+)$$

- s_- / s_+ is the score of the network for the negative/positive example
- The loss is zero when the similarity of the positive example is greater than the similarity of the negative example by at least margin m
- This prevents separation of pos/neg features that are already well separated
- Gives the model the capacity to focus on the hard cases
- The margin m is a tunable hyperparameter (the paper uses $m = 0.2$)

Zbontar and LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. JMLR, 2016.

Loss Function

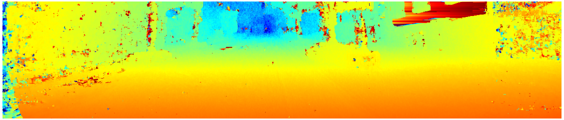


$$\mathcal{L} = \max(0, m + s_- - s_+)$$

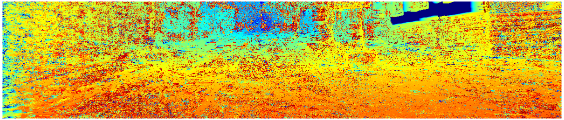
Zbontar and LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. JMLR, 2016.

Winner-takes-All Results

Left Input Image



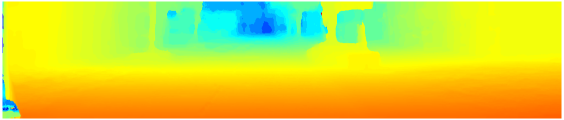
Siamese Network



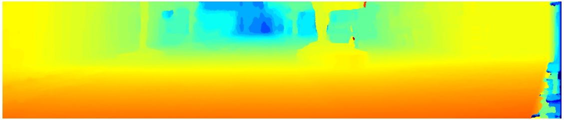
Standard Block Matching

Zbontar and LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. JMLR, 2016.

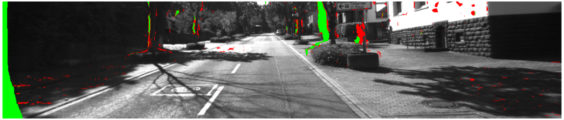
Semiglobal Matching Results



Left Disparity Map



Right Disparity Map



Left-Right Consistency Test

Zbontar and LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. JMLR, 2016.

Runtime

- Original version implemented in CUDA and Lua/Torch7
- Run on Nvidia GTX Titan GPU
- Training takes 5 hours
  - 45 million training examples
  - 16 epochs
  - Stochastic gradient descent with batch size of 128
- Inference for a single pair of images takes 6 seconds / 100 seconds
  - 1 second / 95 seconds for the neural network (depending on architecture)
  - 3 seconds for semiglobal matching
  - 1 second for cost aggregation

Zbontar and LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. JMLR, 2016.

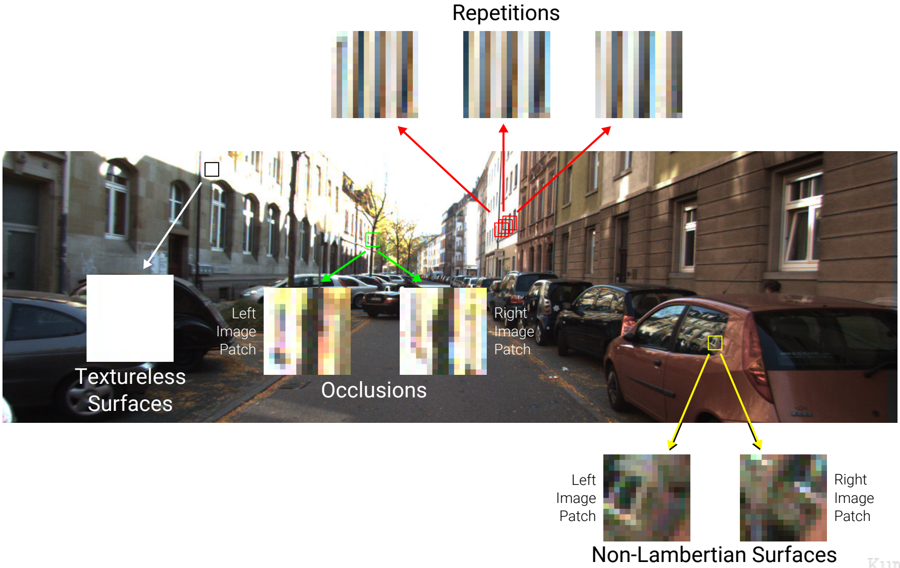
4.4
Spatial Regularization

When will local matching fail?

The Underlying Assumption

- Corresponding regions in both images should look similar
- Non-corresponding regions should look different
- When will this similarity constraint fail?

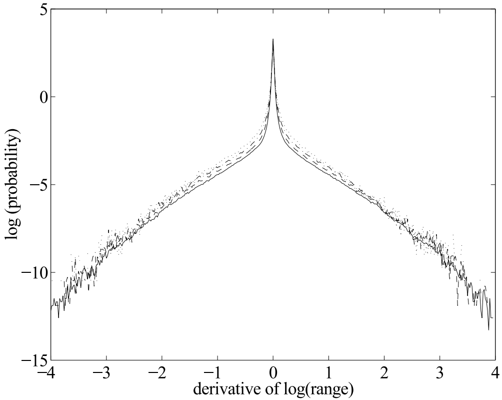
Similarity Constraint: Failure Cases



Spatial Regularization

How does the real world look like?

- Leverage real-world statistics, e.g., Brown range image database
- Conclusion: Depth varies slowly except at object discontinuities which are sparse

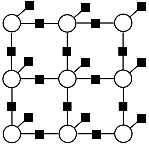


Huang, Lee and Mumford: Statistics of Range Images. CVPR, 2000.

Stereo MRF

- Specify a loopy Markov Random Field (MRF, Lecture 5) on a grid and solve for the whole disparity map $\mathbf{D}$ at once. MAP solution = minimum of energy.

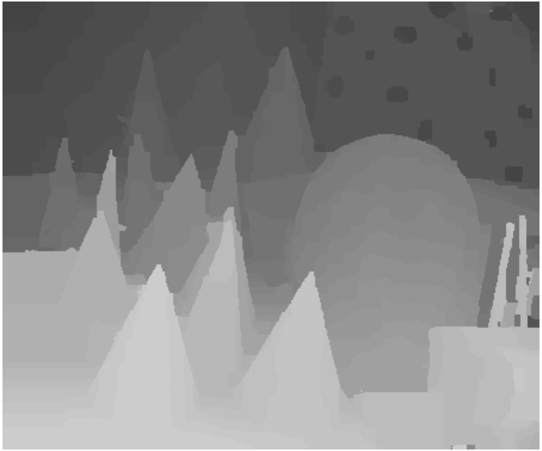
$$p(\mathbf{D}) \propto \exp\left\{ -\sum_i \psi_{data}(d_i) - \lambda \sum_{i \sim j} \psi_{smooth}(d_i, d_j) \right\}$$



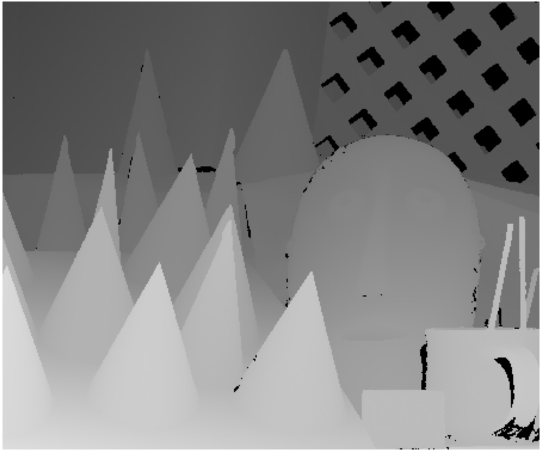
- $i \sim j$: neighboring pixels on a 4-connected grid
- Unary terms: Matching cost $\psi_{data}(d)$
- Pairwise terms: Smoothness between adjacent pixels, e.g.:
  - Potts: $\psi_{smooth}(d, d') = [d \neq d']$
  - Truncated l_1: $\psi_{smooth}(d, d') = \min(|d - d'|, \tau)$
- Solve MRF approximately using belief propagation (Lecture 5-7)

Huang, Lee and Mumford: Statistics of Range Images. CVPR, 2000.

Stereo MRF – Results



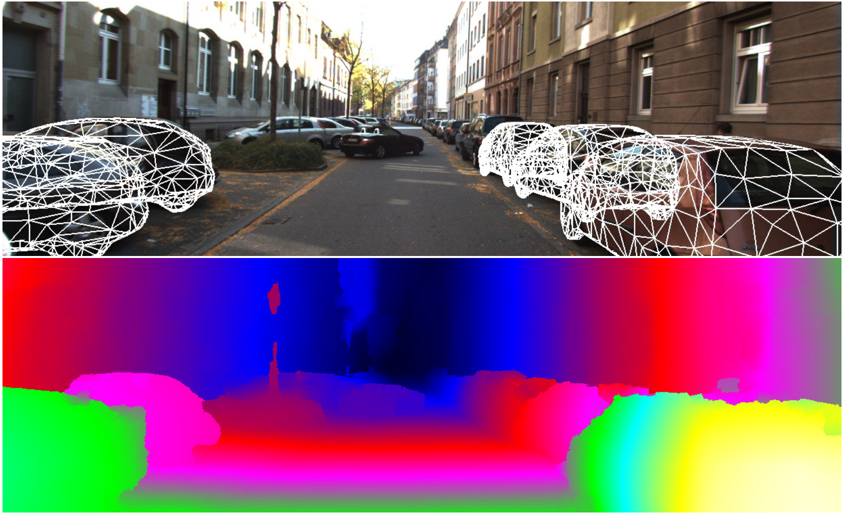
Inference Results



Ground Truth

Huang, Lee and Mumford: Statistics of Range Images. CVPR, 2000.

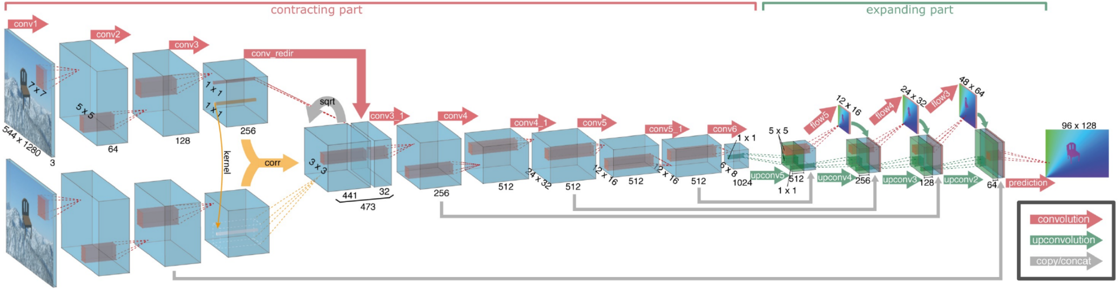
Stereo MRF – Results



Güney and Geiger: Displets: Resolving stereo ambiguities using object knowledge. CVPR, 2015.

4.5
End-to-End Learning

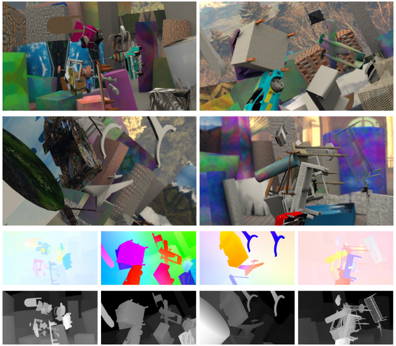
DispNet



- DispNet was the first end-to-end trained deep neural network for stereo
- It used a U-Net like architecture with skip-Connections to retain details
- Correlation layer (40px displacement corresponding to 160px in input image)
- Multi-scale loss (disparity error in pixels), curriculum learning (learn easy-to-hard)

Mayer et al.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR, 2016.

Synthetic Datasets



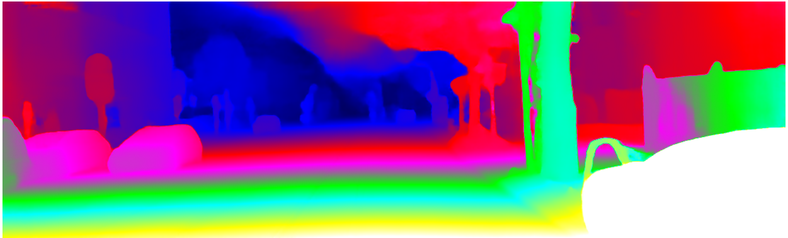
Flying Things

Monkaa

- Pretraining on large synthetic datasets (cheap annotations)
- Finetuning on little real data (expensive annotations)

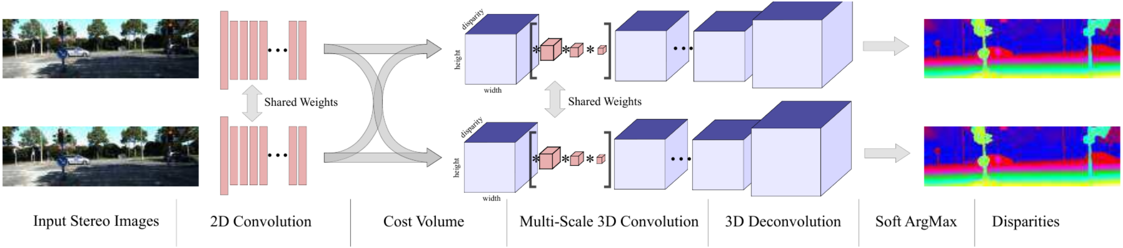
Mayer et al.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR, 2016.

DispNet Results on KITTI Dataset



Mayer et al.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR, 2016.

GC-Net



- Key idea: calculate disparity cost volume and apply 3D convolutions on it
- Convert the learned matching cost $c_\theta(d)$ to disparity via the expectation:

$$d^* = \mathbb{E}[d] = \sum_{d=0}^{D} \underbrace{\text{softmax}(-c_\theta(d))}_{p(d)} \cdot d$$

- Slightly better performance but large memory requirements (3D feature volume)

Kendall, Martirosyan, Dasgupta and Henry: End-to-End Learning of Geometry and Context for Deep Stereo Regression. ICCV, 2017.

Stereo Mixture Density Networks (SMD-Nets)

- Stereo Mixture Density Networks predict sharper boundaries at higher resolution

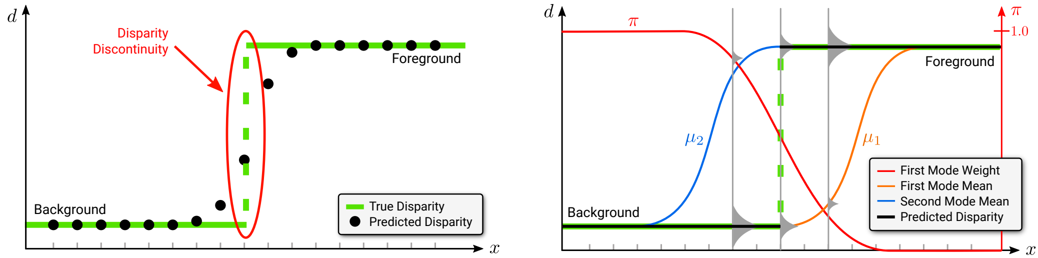
Tosi, Liao, Schmitt and A. Geiger: SMD-Nets: Stereo Mixture Density Networks. CVPR, 2021

Stereo Mixture Density Networks (SMD-Nets)

- Stereo Mixture Density Networks predict sharper boundaries at higher resolution

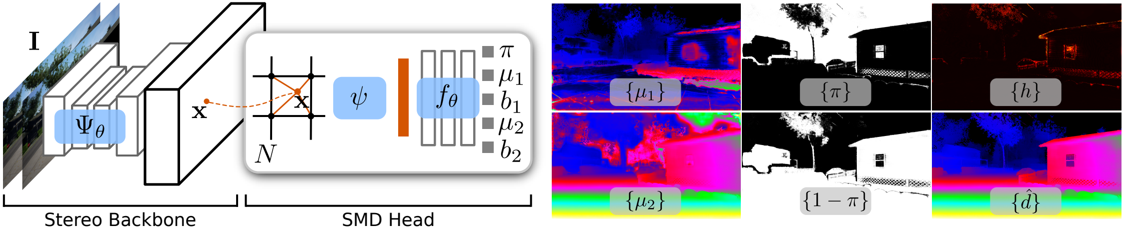
Tosi, Liao, Schmitt and A. Geiger: SMD-Nets: Stereo Mixture Density Networks. CVPR, 2021

Stereo Mixture Density Networks (SMD-Nets)



- Left: Classical deep networks for stereo regression suffer from smoothness bias and hence continuously interpolate object boundaries
- Right: SMD-Nets predict a bimodal (Laplacian) mixture distribution (gray) which allows to accurately capture uncertainty close to depth discontinuities

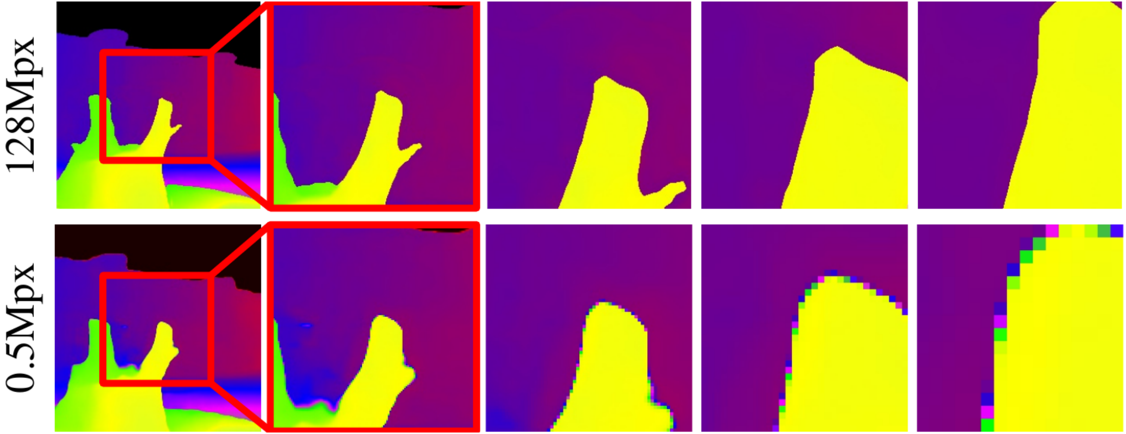
Stereo Mixture Density Networks (SMD-Nets)



- SMD-Nets use a 2D or 3D convolutional backbone in combination with a shallow MLP head that regresses the distribution parameters from interpolated features
- This enables training and inference at arbitrary spatial resolution

Tosi, Liao, Schmitt and A. Geiger: SMD-Nets: Stereo Mixture Density Networks. CVPR, 2021

Stereo Mixture Density Networks (SMD-Nets)



- Top: SMD-Net, Bottom: Traditional stereo regression network

Tosi, Liao, Schmitt and A. Geiger: SMD-Nets: Stereo Mixture Density Networks. CVPR, 2021