These Slides are from Dr. V. Singh, CS Department, UW Madison
3D Perception

Human-Eye Separation (~6.5cm)

Many stereo slides from Michael Bleyer
3D Perception

Human-Eye Separation (~6.5 cm)

Left 2D Image  Right 2D Image
3D Perception

Human-Eye Separation (~6.5 cm)

Left 2D Image  Right 2D Image

Brain
3D Perception

Human-Eye Separation (~6.5 cm)

Left 2D Image  Right 2D Image  3D View
If we ensure that the left eye sees a 2D image and the right eye sees another one, our brain will try to overlay the images to generate a 3D impression.

How can we use this for watching 3D movies?
Anaglyphs

- Two images of complementary color are overlaid to generate one image.
- Glasses required (e.g. red/green)
- Red filter cancels out red image component, green filter cancels out green component
- Each eye gets one image => 3D impression
- Current 3D cinemas use this principle. However, use polarization filters
Shutter Glasses

• Display flickers between left and right image (i.e. each even frame shows left image, each odd frame shows right image)

• When left frame is shown, shutter glasses close right eye and vice versa

• Requires new displays of high frame rate
3D on YouTube
Computational Stereo
Computational Stereo

Replace human eyes with a pair of slightly displaced cameras.
Computational Stereo

Replace human eyes with a pair of slightly displaced cameras.

Displacement (Stereo Baseline)

Brain
Computational Stereo

Displacement (Stereo Baseline)

Left 2D Image

Right 2D Image

Brain
Computational Stereo
Computational Stereo

Displacement (Stereo Baseline)

Left 2D Image

Right 2D Image

Computer
Computational Stereo

Displacement (Stereo Baseline)

Left 2D Image

Right 2D Image

3D View

Computer
Computational Stereo

How can we accomplish a fully automatic 2D to 3D conversion?
What is Disparity?

• The amount to which a single pixel is displaced in the two images is called disparity.

• A pixel’s disparity is inversely proportional to its depth in the scene.
What is Disparity?

- The amount to which a single pixel is displaced in the two images is called disparity.
- A pixel’s disparity is inversely proportional to its depth in the scene (formally described shortly).
What is Disparity?

- The amount to which a single pixel is displaced in the two images is called disparity.
- A pixel’s disparity is inversely proportional to its depth in the scene.
What is Disparity?

- The amount to which a single pixel is displaced in the two images is called disparity.
- A pixel’s disparity is inversely proportional to its depth in the scene.
What is Disparity?

- The amount to which a single pixel is displaced in the two images is called disparity.
- A pixel’s disparity is inversely proportional to its depth in the scene.
What is Disparity?

- The amount to which a single pixel is displaced in the two images is called disparity.
- A pixel’s disparity is inversely proportional to its depth in the scene.
Disparity Encoding

- Disparity of each pixel is encoded by a gray value.
- High grey values represent high disparities (and low gray values small disparities).
- The resulting image is called disparity map.
The disparity map contains sufficient information for generating a 3D model.
The disparity map contains sufficient information for generating a 3D model. The challenging part is to compute the disparity map. This task is known as the stereo matching problem.

- The disparity map contains sufficient information for generating a 3D model.
Applications
(just a few examples)
3D Reconstruction from aerial images

- Stereo cameras are mounted on an airplane to obtain a terrain map.
- Images taken from http://www.robotic.de/Heiko.Hirschmueller/
3D Reconstruction of Cities

- City of Dubrovnik reconstructed from images taken from Flickr in a fully automatic way.
Driver Assistance / Autonomous driving cars

• For example, use stereo to measure distance to other cars.

The Mars Rover

- Reconstruct the surface of Mars using stereo vision
Human Motion Capture

- Fit a 3D model of the human body to the computed point cloud.
- [R. Plänkers and P. Fua, “Articulated Soft Objects for Multiview Shape and Motion Capture”, PAMI, 2003]
Bilayer Segmentation – Z-Keying

• Goal: Divide image into a foreground and a background region.
• Simple background subtraction will fail if there is motion in the background (see Jia’s paper in ICCV 13).
• Solution:
  • Compute depth map
  • If the depth of a pixel is larger than a predefined threshold, pixels belongs to the foreground
View Morphing

- Morph between pair of images using epipolar geometry [Seitz & Dyer, SIGGRAPH’96]
More technical version
Pinhole Camera

- Simplest model for describing the projection of a 3D scene onto a 2D image.
- Model is commonly used in computer vision.
• Let us assume we have a pinhole camera.
• The pinhole camera is characterized by its focal point $C_i$ and its image plane $L$. 
We also have a second pinhole camera \(<C_r, R>\).
We assume that the camera system is fully calibrated, i.e. the 3D positions of \(<C_l, L>\) and \(<C_r, R>\) are known.
Image Formation Process

- We have a 3D point $P$. 
• We compute the 2D projection $p_l$ of $P$ onto the image plane of the left camera $L$ by intersecting the ray from $C_l$ to $P$ with the plane $L$.
• This is what is happening when you take a 2D image of a 3D scene with your camera (image formation process).
We compute the 2D projection $p_l$ of $P$ onto the image plane of the left camera $L$ by intersecting the ray from $C_l$ to $P$ with the plane $L$.

This is what is happening when you take a 2D image of a 3D scene with your camera (image formation process).

Nice, but actually we want to do exactly the opposite: “We have a 2D image and want to make it 3D.”
3D Reconstruction

- Task: We have a 2D point $p_l$ and want to compute its 3D position $P$. 
3D Reconstruction

- $P$ has to lie on the ray of $C_l$ to $p_l$.
- Problem: It can lie anywhere on this ray.
Let us assume we also know the 2D projection $p_r$ of $P$ onto the right image plane $R$. 

3D Reconstruction
$P$ can now be reconstructed by intersecting the rays $C_l p_l$ and $C_r p_r$. 
3D Reconstruction

The challenging part is to find the pair of corresponding pixels $p_l$ and $p_r$ that are projections of the same 3D point $P = \text{Stereo Matching Problem}$.
3D Reconstruction

Problem: Given $p_l$, the corresponding pixel $p_r$ can lie at any x- and y-coordinate in the right image.

Can we make the search easier?
Epipolar Geometry

- We have stated that $P$ has to lie on the ray $C_ip_i$. 
Epipolar Geometry

- If we project each candidate 3D point onto the right image plane, we see that they all lie on a line in $R$. 
Epipolar Geometry

• If we project each candidate 3D point onto the right image plane, we see that they all lie on a line in $R$. 
Epipolar Geometry

- If we project each candidate 3D point onto the right image plane, we see that they all lie on a line in $R$. 
If we project each candidate 3D point onto the right image plane, we see that they all lie on a line in $R$. 

**Epipolar Geometry**
Epipolar Geometry

- This line is called epipolar line of $p_l$.
- This epipolar line is the projection of the ray $C_l p_l$ onto the right image plane $R$.
- The pixel $p_r$ is forced to lie on $p_l$’s epipolar line.
To find the corresponding pixel, we only have to search along the epipolar line (1D instead of 2D search).

This search space restriction is known as epipolar constraint.
Stereo correspondence

- Determine Pixel Correspondence
  - Pairs of points that correspond to same scene point

Epipolar Constraint
- Reduces correspondence problem to 1D search along *epipolar lines*
Stereo correspondence

• Determine Pixel Correspondence
  • Pairs of points that correspond to same scene point

Epipolar Constraint

• Reduces correspondence problem to 1D search along \textit{conjugate epipolar lines}
• Java demo: http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html
Stereo correspondence

- Determine Pixel Correspondence
  - Pairs of points that correspond to same scene point

Epipolar Constraint
- Reduces correspondence problem to 1D search along conjugate epipolar lines
Stereo correspondence

- Determine Pixel Correspondence
  - Pairs of points that correspond to same scene point

Epipolar Constraint
- Reduces correspondence problem to 1D search along \textit{conjugate epipolar lines}
- Java demo: http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html
Stereo correspondence

• Determine Pixel Correspondence
  • Pairs of points that correspond to same scene point

Epipolar Constraint

• Reduces correspondence problem to 1D search along *conjugate epipolar lines*

• Java demo:  [http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html](http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html)
Stereo correspondence

• Determine Pixel Correspondence
  • Pairs of points that correspond to same scene point

Epipolar Constraint

• Reduces correspondence problem to 1D search along *conjugate epipolar lines*
• Java demo:  [http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html](http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html)
Stereo correspondence

- Determine Pixel Correspondence
  - Pairs of points that correspond to same scene point

Epipolar Constraint
- Reduces correspondence problem to 1D search along *conjugate epipolar lines*
Epipolar Line Example
courtesy of Marc Pollefeys
Stereo Pipeline

- Left Image
- Right Image
- Epipolar Rectification
- Rectified Left Image
- Rectified Right Image
- Stereo Matching
- Disparity Map
- Depth via Triangulation
- 3D Scene Reconstruction
Stereo image rectification
Stereo image rectification
Stereo image rectification

- Reproject image planes onto a common plane parallel to the line between optical centers.
- Pixel motion is horizontal after this transformation.
- Two homographies, one for each input image reprojection.

Computing Rectifying Homographies for Stereo Vision. (CVPR 1999)
Epipolar Line Example

courtesy of Marc Pollefeys
Epipolar Line Example

courtesy of Marc Pollefeys
Epipolar Rectification

- Specifically interesting case:
  - Image plane $L$ and $R$ lie in a common plane.
  - $X$-axes are parallel to the baseline
  - Epipolar lines coincide with horizontal scanlines => corresponding pixels have the same y-coordinate
• Specifically interesting case:
  • Image plane $L$ and $R$ lie in a common plane.
  • X-axes are parallel to the baseline
• Epipolar lines coincide with horizontal scanlines => corresponding pixels have the same $y$-coordinate
Epipolar Rectification

- Specifically interesting case:
  - Image plane $L$ and $R$ lie in a common plane.
  - X-axes are parallel to the baseline
  - Epipolar lines coincide with horizontal scanlines => corresponding pixels have the same y-coordinate
Epipolar Rectification

- Specifically interesting case:
  - Image plane $L$ and $R$ lie in a common plane.
  - X-axes are parallel to the baseline
  - Epipolar lines coincide with horizontal scanlines => corresponding pixels have the same y-coordinate
Epipolar Rectification

To find the corresponding pixel, we only have to search along the horizontal scanline.

More convenient than tracing arbitrary epipolar lines.

The difference in x-coordinates of corresponding pixels is called disparity.
Epipolar Rectification

- This special case can be achieved by reprojecting left and right images onto virtual cameras.
- This process is known as epipolar rectification.
- After this, we can assume that images have been rectified.

Images taken from http://profs.sci.univr.it/~fusiello/rectif_cvol/node6.html
Summary

• 1D search is computationally faster than 2D search.
• Reduced search range lowers chance of finding a wrong match (Quality of depth maps).
• More or less the only constraint that will always be valid in stereo matching (unless there are calibration errors).
Depth via Triangulation

- $B$ and $X-B$
- $Z$ [meters]
- $x_l$, $x_r$
- $f$
- $x$ [pixels]
- $X$ [meters]
- $P$
- $C_l$, $C_r$
Depth via Triangulation

- **Z-value in 3D space**
- **X-coordinate in 2D image**
- **X-coordinate in 3D space**

Symbols:
- $Z$ [meters]
- $X$ [meters]
- $x$ [pixels]
- $f$
- $B$
- $X-B$
Depth via Triangulation

Baseline (Distance between focal points of the cameras)

All of these values are known from camera calibration
Depth via Triangulation

X-Coordinates of corresponding points
(correspondences known from stereo matching)
Depth via Triangulation

This is what we want to compute.

**X-coordinate in 3D space (unknown)**

**Z-coordinate (depth) in 3D space (unknown)**
Depth via Triangulation

\[ B \quad X-B \]

\[ Z \text{ [meters]} \]

\[ x_{l} \quad x_{r} \]

\[ p_{l} \quad p_{r} \]

\[ C_{l} \quad C_{r} \]

\[ X \text{ [meters]} \]

\[ x \text{ [pixels]} \]

\[ f \]

\[ Z \]
Depth via Triangulation

Similar Triangles:

\[
\frac{X}{Z} = \frac{x_l}{f}
\]
Depth via Triangulation

Similar Triangles:

\[ \frac{X - B}{Z} = \frac{x_r}{f} \]
Depth via Triangulation

- From similar triangles:

\[
\frac{X}{Z} = \frac{x_l}{f} \quad \frac{X - B}{Z} = \frac{x_r}{f}
\]
Depth via Triangulation

• From similar triangles:

\[
\frac{X}{Z} = \frac{x_l}{f} \quad \quad \frac{X - B}{Z} = \frac{x_r}{f}
\]

• Write X in explicit form:

\[
X = \frac{Z \cdot x_l}{f} \quad \quad X = \frac{Z \cdot x_r}{f} + B
\]
Depth via Triangulation

- From similar triangles:
  \[
  \frac{X}{Z} = \frac{x_l}{f} \quad \frac{X - B}{Z} = \frac{x_r}{f}
  \]

- Write X in explicit form:
  \[
  X = \frac{Z \cdot x_l}{f} \quad X = \frac{Z \cdot x_r}{f} + B
  \]

- Combine both equations:
  \[
  \frac{Z \cdot x_l}{f} = \frac{Z \cdot x_r}{f} + B
  \]
  \[
  Z \cdot x_l = Z \cdot x_r + B \cdot f
  \]
  \[
  Z \cdot (x_l - x_r) = B \cdot f
  \]

- Write Z in explicit form:
  \[
  Z = \frac{B \cdot f}{x_l - x_r} = \frac{B \cdot f}{d}
  \]

This is disparity
Depth via Triangulation

- From similar triangles:

  \[ \frac{X}{x_l} = \frac{Z}{x_r} \]

- Write \( X \) in explicit form:

  \[ X = \frac{f \cdot Z \cdot x_r}{x_l} \]

- Combine both equations:

  \[ \frac{X}{x_l} = \frac{Z \cdot x_r + B \cdot f}{x_l} \]

- Write \( Z \) in explicit form:

  \[ Z = \frac{B \cdot f}{x_l - x_r} \]

  \[ Z = \frac{B \cdot f}{d} \]

Disparity and depth are inversely proportional!

Therefore, disparity is commonly used synonymously with depth.

This is disparity
We will now focus on this problem.
Basic stereo algorithm

For each epipolar line (raster scan top to bottom)
For each pixel in the left image (left to right)
  • compare with every pixel on same epipolar line in right image
  • pick pixel with minimum match cost

Improvement: match windows
The similarity constraint

- Corresponding regions in two images should be similar in appearance
- and non-corresponding regions should be different
- When will the similarity constraint fail?
Limitations of similarity constraint

- Textureless surfaces
- Occlusions, repetition
- Non-Lambertian surfaces
Correspondence

(After calibration) all correspondences are along the same horizontal scan lines
Correspondence via Correlation

Rectified images

(Same as max-correlation / max-cosine for normalized image patch)
Image Normalization

• The cameras do not see exactly the same surfaces, so their overall light levels will differ: good idea to normalize the pixels in each window (accounts for changes in gain and sensitivity)

\[
\bar{I} = \frac{1}{|W_m(x,y)|} \sum_{(u,v)\in W_m(x,y)} I(u,v) \quad \text{Average pixel}
\]

\[
\|I\|_{W_m(x,y)} = \sqrt{\sum_{(u,v)\in W_m(x,y)} [I(u,v)]^2} \quad \text{Window magnitude}
\]

\[
\hat{I}(x,y) = \frac{I(x,y) - \bar{I}}{\|I - \bar{I}\|_{W_m(x,y)}} \quad \text{Normalized pixel}
\]
Images as Vectors

Each window is a vector in an $m^2$ dimensional vector space. Normalization makes them unit length.
Basic stereo algorithm

- For each disparity
  - For each pixel
    - For each pixel in window
      Compute difference
  - Find disparity with minimum SSD at each pixel
Incremental computation

- Given SSD of a window, at some disparity
Incremental computation

- Want: SSD at next location
Incremental computation

- Subtract contributions from leftmost column, add contributions from rightmost column
Incremental computation

- Efficient implementation with “sliding column”, Faugeras et. al.’1993 for window cost function

\[ \sum_{x, y \in \square} f(x, y) \]

- Running time is independent of window size
Window based approach

- Typical window cost function
- Location with the best cost wins

$$\sum_{x,y \in \square} f(x,y)$$
Selecting window size

- Small window: more detail, but more noise
- Large window: more robustness, less detail
- Example:
Selecting window size

3 pixel window

20 pixel window
Problems with Fixed Windows

small window

- better at boundaries
- noisy in low texture areas

large window

- better in low texture areas
- blurred boundaries
Non-square windows

• Compromise: have a large window, but higher weight near the center
• Example: Gaussian
Problems with window matching

• No guarantee that the matching is one-to-one
• Hard to balance window size and smoothness
A global approach

- Finding correspondence between a pair of epipolar lines for all pixels simultaneously
A global approach

- Finding correspondence between a pair of epipolar lines for all pixels simultaneously

Still challenging.
Why is Stereo Matching Challenging? (1)

• Color inconsistencies:
  • When solving the stereo matching problem, we typically assume that corresponding pixels have the same intensity/color (= Photo consistency assumption)
  • That does not need to be true due to:
    – Image noise
    – Different illumination conditions in left and right images
    – Specular reflections
Why is Stereo Matching Challenging? (2)

- Untextured regions (Matching ambiguities)
  - There needs to be a certain amount of intensity/color variation (i.e. texture) so that a pixel can be uniquely matched in the other view.
  - Can you (as a human) depict depth if you are standing in front of a wall that is completely white?

Left image (no texture in the background)  Right image  Computed disparity map (errors in background)
Why is Stereo Matching Challenging? (3)

- Occlusion Problem
  - There are pixels that are only visible in exactly one view.
  - We call this pixels occluded (or half-occluded)
  - It is difficult to estimate depth for these pixels.
Let’s consider a simple scene composed of a foreground and a background object.
The Occlusion Problem

- Regular case:
  - The white pixel $P_1$ can be seen by both camera.
The Occlusion Problem

- Occlusion in the right camera:
  - The left camera sees the grey pixel $P_2$.
  - The ray from the right camera to $P_2$ hits the white foreground object => $P_2$ cannot be seen by right camera.
The Occlusion Problem

- Occlusion in the left camera:
  - The right camera sees the grey pixel $P_3$.
  - The ray from the left camera to $P_3$ hits the white foreground object $\Rightarrow P_3$ cannot be seen by left camera.
The Occlusion Problem

- Occlusions occur in the proximity of disparity discontinuities.
The Occlusion Problem

Occlusions occur as a consequence of discontinuities in depth.

They occur close object/depth boundaries.

They occur in both frames
The Occlusion Problem

- In the left image, occlusions are located to the left of a disparity boundary.
- In the right image, occlusions are located to the right of a disparity boundary.
The Occlusion Problem

- It is difficult to find disparity if the matching point does not exist!
- Ignoring the occlusion problem leads to disparity artefacts near disparity borders.
Stereo

If x-shifts (disparities) are known for all pixels in the left (or right) image then we can visualize them as a disparity map: the image above makes sense globally.
Variable window size algorithms

- Correspondences are still found independently at each pixel
- All Window-based solution can be thought of as “local” solutions
  - Fast
- How to introduce spatial coherence
  - Introduce Energy function
  - Searching for “global” solutions
A global approach: Dynamic Programming

Define a global evaluation score for each configuration, choose the best matching configuration.

**Evaluation score:** the sum of corresponding pixel difference?
Correspondences

Match intensities sequentially between two scanlines
Correspondences

Match intensities sequentially between two scanlines
Correspondences

Match intensities sequentially between two scanlines
Correspondences

Match intensities sequentially between two scanlines
Correspondences

Match intensities sequentially between two scanlines
Correspondences

Match intensities sequentially between two scanlines
Correspondences

Left scanline

... Left occlusion

Match

Right occlusion

Match

Match

Right scanline
Search Over Correspondences

Three cases:

- Sequential – cost of match
- Left occluded – cost of no match
- Right occluded – cost of no match
Digression....

**Dynamic programming example**

Consider the matrix multiplication example: A1: 5x4, A2: 4x6, A3: 6x2

![Matrix Multiplication Diagram](image)

Fig. 7: Matrix Multiplication.

Note that although any legal parenthesization will lead to a valid result, not all involve the same number of operations. Consider the case of 3 matrices: \( A_1 \) be \( 5 \times 4 \), \( A_2 \) be \( 4 \times 6 \) and \( A_3 \) be \( 6 \times 2 \).

\[
\begin{align*}
\text{multCost}[(A_1A_2)A_3] &= (5 \cdot 4 \cdot 6) + (5 \cdot 6 \cdot 2) = 180, \\
\text{multCost}[A_1(A_2A_3)] &= (4 \cdot 6 \cdot 2) + (5 \cdot 4 \cdot 2) = 88.
\end{align*}
\]

Look for
- Optimal substructure
- Overlapping subproblems
Matrix Chain-Products

• **Matrix Chain-Product:**
  • Compute $A = A_0 \cdot A_1 \cdot \ldots \cdot A_{n-1}$
  • Problem: How to parenthesize?

• **Example**
  • B is $3 \times 100$
  • C is $100 \times 5$
  • D is $5 \times 5$
  • $(B \cdot C) \cdot D$ takes $1500 + 75 = 1575$ ops
  • $B \cdot (C \cdot D)$ takes $1500 + 2500 = 4000$ ops
## Dynamic Programming Algorithm

- $A_0$: 30 X 35; $A_1$: 35 X15; $A_2$: 15X5;
  $A_3$: 5X10;  $A_4$: 10X20;  $A_5$: 20 X 25

\[
N_{i,j} = \min_{i \leq k < j} \{N_{i,k} + N_{k+1,j} + d_id_{k+1}d_{j+1}\}
\]

\[
\begin{align*}
N_{1,4} &= \min\{ \\
N_{1,1} + N_{2,4} + d_1d_2d_5 &= 0 + 2500 + 35*15*20 = 13000, \\
N_{1,2} + N_{3,4} + d_1d_3d_5 &= 2625+1000 + 35*5*20 = 7125, \\
N_{1,3} + N_{4,4} + d_1d_4d_5 &= 4375 + 0 + 35*10*20 = 11375 \\
\} \\
&= 7125
\end{align*}
\]
Dynamic programming yields the optimal path through grid. This is the best set of matches that satisfy the ordering constraint.
Dynamic Programming Results

Left 2D Image

Right 2D Image

Disparity Map
Dynamic Programming Results
No global optimization

• Quite disappointing, why?
• We have posed the following task:
  • I have a red pixel. Find me a red pixel in the other image.
• Problem:
  • There are usually many red pixels in the other image (ambiguity)
• We need additional assumptions.
Note the horizontal streaks!
It doesn’t want to change its mind!
What is the most obvious problem?

- What is the most obvious difference between the correct and the computed disparity maps?
Smoothness Assumption (1)

- Observation:
  - A correct disparity map typically consists of regions of constant (or very similar) disparity. For example, lamp, head, table,
  - We can give this apriori knowledge to a stereo algorithm in the form of a smoothness assumption
Smoothness Assumption

- Smoothness assumption:
  - Spatially close pixels have the same (or similar) disparity.
  - (By spatially close I mean pixels of similar image coordinates.)
- Smoothness assumption typically holds true almost everywhere, except at disparity borders.
Revisit DP for scan-line stereo

Viterbi algorithm can be used to optimize the following energy of disparities $d = \{d_p \mid p \in S\}$ of pixels $p$ on a fixed scan-line $S_{\text{right}}$

$$E(d) = \sum_{p \in S} D_p(d_p) + \sum_{p \in S} V(d_p,d_{p+1}) = \sum_{\{p,q\} \in N} E(d_p,d_q)$$

$|I_p - I'_p|_{p\oplus d_p}$

photo consistency

$|d_p - d_{p+1}|$

spatial coherence

Viterbi can handle this on non-loopy graphs (e.g., scan-lines)
Other Global Methods: graph based

- Define a cost function to measure the quality of a disparity map:
  - High costs mean that the disparity map is bad.
  - Low costs mean it is good.
- Costs function is typically in the form of:

\[ E = E_{data} + E_{smooth} \]

where

- \( E_{data} \) measures photo consistency
- \( E_{smooth} \) measures smoothness
- Global methods express smoothness assumption in an explicit form
Coherent stereo on 2D grid?

• Scan-line stereo generates streaking artifacts
• Can’t use DP (or Dijkstra) to find spatially coherent disparities/correspondances on a 2D grid

• Now let’s talk about graph cuts algorithm
**Graph cuts** for spatially coherence on grids

We will globally minimize the same energy of disparities $\mathbf{d} = \{d_p \mid p \in G\}$ for pixels $p$ on a grid $G$

$$E(\mathbf{d}) = \sum_{p \in G} D_p(d_p) + \sum_{\{p,q\} \in N} V(d_p, d_q)$$

- $|I_p - I'_{p \oplus d_p}|$  photo consistency
- $w_{pq} \cdot |d_p - d_q|$  spatial coherence

Consider a 2D grid $G$ corresponding to (right) image pixels.

Weights $w_{pq}$ of neighborhood edges, **n-links**, may reflect (right) image intensity gradients (**static cues**). 

**static cues**
Multi-scan-line stereo with graph cuts
(Roy&Cox’98)
Multi-scan-line stereo with graph cuts
(Roy & Cox’98)
Concentrate on one pair of neighboring pixels \( \{p, q\} \in N \)

\[
E(d_p, d_q) = D_p(2) + D_q(5) + \ldots + w_{pq} \cdot |3| + \ldots
\]

Disparity labels

-5
-4
-3
-2
-1
0

Cost of vertical edges

Cost of horizontal edges
Concentrate on one pair of neighboring pixels \( \{p, q\} \in N \)

\[
E(d_p, d_q) = D_p(d_p) + D_q(d_q) + \ldots + w_{pq} \cdot |d_p - d_q| + \ldots
\]

- cost of vertical edges
- cost of horizontal edges
The combined energy over the entire grid $G$ is

$$E(d) = \sum_{p \in G} D_p(d_p) + \sum_{\{p,q\} \in N} w_{pq} \cdot |d_p - d_q|$$

(\textit{photo consistency})

(cost of vertical edges)

(cost of horizontal edges)

(\textit{spatial consistency})
Multi-scan-line stereo with graph cuts
(Roy&Cox’98)

Dynamic Programming
(single scan line optimization)

s-t Graph Cuts
(multi-scan-line optimization)
Graph Cuts Basics (see CLRS)

Simple 2D example

Goal: divide the graph into two parts separating red and blue nodes

A graph with two terminals S and T

- Cut cost is a sum of severed edge weights
- Minimum cost s-t cut can be found in polynomial time
Graph cut is an old problem

From Harris & Ross [1955]
Graph cut is an old problem

From Harris & Ross [1955]
“Augmenting Paths”

- Find a path from S to T along non-saturated edges
- Increase flow along this path until some edge saturates

A graph with two terminals
“Augmenting Paths”

- Find a path from $S$ to $T$ along non-saturated edges

Increase flow along this path until some edge saturates

- Find next path
- Increase flow
“Augmenting Paths”

- Find a path from S to T along non-saturated edges
  - Increase flow along this path until some edge saturates

A graph with two terminals

Iterate until … all paths from S to T have at least one saturated edge

MAX FLOW $\Leftrightarrow$ MIN CUT
Some results from Roy & Cox

- Minimum cost s/t cut
- Multi scan line stereo
- Single scan line stereo (DP)
Some results from Roy & Cox

multi scan line stereo
single scan line stereo

minimum cost s/t cut
Stereo results

- Data from University of Tsukuba

http://cat.middlebury.edu/stereo/
Cancellation of Flow

\[
\begin{array}{c}
\text{v}_1 \\
8/10 \rightarrow 3/4 \rightarrow \text{v}_2 \\
\text{v}_2 \\
\end{array}
\quad = \quad
\begin{array}{c}
\text{v}_1 \\
5/10 \rightarrow 0/4 \rightarrow \text{v}_2 \\
\text{v}_2 \\
\end{array}
\]
Definition of Network Flow

flow $f: (E \cup E^R) \rightarrow R^+$

(1) $f(e) = -f(e^R)$ for all $e \in E$

(2) $f(e) \leq c(e)$

(3) $\sum_{(v,u) \in E} f(v,u) = 0$ for all $v \in V - \{s, t\}$
Results with window correlation

normalized correlation (best window size)  

ground truth
Results with data + smooth

so-called “graph cuts”  ground truth
Middlebury Stereo Benchmark
For stereo, ground truth data is available on the Middlebury Stereo Evaluation website [http://vision.middlebury.edu/stereo/](http://vision.middlebury.edu/stereo/).

The Middlebury set is widely adopted in the stereo community.
The Middlebury Set

<table>
<thead>
<tr>
<th>2003 Sets</th>
<th>2005 Sets</th>
<th>2006 Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tsukuba</td>
<td>Art</td>
<td>Aloe</td>
</tr>
<tr>
<td>Venus</td>
<td>Books</td>
<td>Baby1</td>
</tr>
<tr>
<td>Teddy</td>
<td>Dolls</td>
<td>Baby2</td>
</tr>
<tr>
<td>Cones</td>
<td>Laundry</td>
<td>Baby3</td>
</tr>
<tr>
<td></td>
<td>Moebius</td>
<td>Rocks1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rocks2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wood1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bowling2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lampshade1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flowerpots</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Push Flow of 2

Total Flow pushed: 2
Residual Graph

Total Flow pushed: 2
Residual Graph

Total Flow pushed: 3
Residual Graph

Total Flow pushed: 3
Residual Graph

Total Flow pushed: 5
Max flow Min Cut

Total Flow pushed: 5
Min Cut

Total Flow pushed: 5