STEREEO VISION.
Thank you for the slides.
They come mostly from the following sources.

**Martial Hebert** CMU

**Dan Huttenlocher** Cornell U.

**David Lowe** U. of British Columbia

**Svetlana Lazebnik** U. of North Carolina at Chapel Hill
Stereo Reconstruction

The Stereo Problem
- Shape from two (or more) images
- Biological motivation

Why do we have two eyes?

Cyclope vs. TA
Random dot stereograms

Julesz: showed that recognition is not needed for stereo.

e.g. http://www.inf.ufsc.br/~otuyama/eng/stereogram/basic/index.html
Depth from disparity

\[ \text{disparity} = x - x' = \frac{\text{baseline} \cdot f}{z} \]


Stereo reconstruction pipeline

Steps
- Calibrate cameras
- Rectify images
- Compute disparity
- Estimate depth

What will cause errors?
- Camera calibration errors
- Poor image resolution
- Occlusions
- Violations of brightness constancy (specular reflections)
- Large motions
- Low-contrast image regions
Stereo vision

Triangulate on two images of the same point to recover depth.
- Feature matching across views
- Calibrated cameras

Matching correlation windows across scan lines
The epipolar constraint

- Epipolar Constraint
  - Matching points lie along corresponding epipolar lines
  - Reduces correspondence problem to 1D search along *conjugate epipolar lines*
  - Greatly reduces cost and ambiguity of matching

Slide credit: Steve Seitz
Simplest Case: Rectified Images

- Image planes of cameras are parallel.
- Focal points are at same height.
- Focal lengths same.
- Then, epipolar lines fall along the horizontal scan lines of the images

- We will assume images have been *rectified* so that epipolar lines correspond to scan lines
  - Simplifies algorithms
  - Improves efficiency
We can always achieve this geometry with image rectification

- Image Reprojection
  - reproject image planes onto common plane parallel to line between optical centers
- Notice, only focal point of camera really matters

(Seitz)
We know that, given a plane $P$ in space, there exists two homographies $H_l$ and $H_r$ that map each image plane onto $P$. That is, if $p_l$ is a point in the left image, then the corresponding point in $P$ is $H_p$ (in homogeneous coordinates). If we map both images to a common plane $P$ such that $P$ is parallel to the line $C_lC_r$, then the pair of virtual (rectified) images is such that the epipolar lines are parallel. With proper choice of the coordinate system, the epipolar lines are parallel to the rows of the image.

The algorithm for rectification is then:

• Select a plane $P$ parallel to $C_lC_r$
• Define the left and right image coordinate systems on $P$
• Construct the rectification matrices $H_l$ and $H_r$ from $P$ and the virtual image’s coordinate systems.
Rectification

(a) Original image pair overlayed with several epipolar lines.

(b) Image pair transformed by the specialized projective mapping $H_p$ and $H_p'$. Note that the epipolar lines are now parallel to each other in each image.

BAD!

tilt around the y-axis removed but this is not enough

Cornell University
Rectification

epipolar lines o.k. - similarity transformations

(c) Image pair transformed by the similarity $H_2$ and $H'_2$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).

(d) Final image rectification after shearing transform $H_s$ and $H'_s$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

GOOD!

rescale the second image without change the orientation
Choosing the Baseline

What’s the optimal baseline?
- Too small: large depth error
- Too large: difficult search problem
Basic Stereo Derivations

The two images already aligned

Baseline $B$

$P_L = (X,Y,Z)$

$$ (u_L, v_L) = \left( f \frac{X}{Z}, f \frac{Y}{Z} \right) $$

$$ (u_R, v_R) = \left( f \frac{X - B}{Z}, f \frac{Y}{Z} \right) $$

Disparity:

$$ d = u_L - u_R = f \frac{B}{Z} $$

$$ Z = f \frac{B}{d} $$
Correspondence

- It is fundamentally ambiguous, even with stereo constraints

Ordering constraint... ...and its failure
Correspondence: What should we match?

- Objects?
- Edges?
- Pixels?
- Collections of pixels?

...each for the methods has its pros and cons...
Correspondence: Photometric constraint

- Same world point has same intensity in both images.
  - True for Lambertian surfaces
    - A Lambertian surface has a brightness that is independent of viewing angle
  - Violations:
    - Noise
    - Specularity
    - Non-Lambertian materials
    - Pixels that contain multiple surfaces
Finding Correspondences

- Apply feature matching criterion (e.g., correlation) at all pixels simultaneously
- Search only over epipolar lines (many fewer candidate positions)
Pixel matching

For each epipolar line
  For each pixel in the left image
    • compare with every pixel on same epipolar line in right image
    • pick pixel with minimum match cost

This leaves too much ambiguity, so:

Improvement: match *windows*
Correspondence: Epipolar constraint.

The epipolar constraint helps, but much ambiguity remains.
Image Normalization

- Even when the cameras are identical models, there can be differences in gain and sensitivity.
- For these reasons and more, it is a good idea to normalize the pixels in each window:

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

Average pixel

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

Window magnitude

\[
\hat{I}(x, y) = \frac{I(x, y) - \bar{I}}{\|I - \bar{I}\|_{W_m(x,y)}}
\]

Normalized pixel

\[\text{an mxm area in both images}\]
Images as Vectors

“Unwrap” image to form vector, using raster scan order

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

$W_L$ $W_R$

Row 1 $m$
Row 2 $m$
Row 3 $m$


**Image Metrics**

(Normalized) Sum of Squared Differences

\[ C_{SSD}(d) = \sum_{(u,v) \in W_m(x,y)} [\hat{I}_L(u,v) - \hat{I}_R(u - d, v)]^2 \]

\[ = \|w_L - w_R(d)\|^2 \]

Normalized Correlation

\[ C_{NC}(d) = \sum_{(u,v) \in W_m(x,y)} \hat{I}_L(u,v)\hat{I}_R(u - d, v) \]

\[ = w_L \cdot w_R(d) = \cos \theta \]

\[ d^* = \arg \min_d \|w_L - w_R(d)\|^2 = \arg \max_d w_L \cdot w_R(d) \]
Correspondence Using Correlation

Left

Right

scanline

SSD error

disparity
Block Based Matching

- How to determine correspondences?
  - *Block matching* or *SSD* (sum squared differences)
    
    $$E(x, y; d) = \sum_{(x', y') \in N(x, y)} [I_L(x' + d, y') - I_R(x', y')]^2$$
    
    $d$ is the *disparity* (horizontal motion)

- How big should neighborhood be?
Effects of Block Size

- Smaller neighborhood: more details
- Larger neighborhood: fewer isolated mistakes

\[ w = 3 \quad \text{vs.} \quad w = 20 \]
Computing Correspondence

- Another approach is to match *edges* rather than windows of pixels:

[Marr, Poggio and Grimson, 1979-81]

- Edges tend to fail in dense texture (outdoors)
- Correlation tends to fail in smooth featureless areas
Feature-based stereo matching

Feature-based stereo matching

• Pros
  • Robust to clutter and occlusion
  • Only find matches at reliable points
  • Can use invariant local features to deal with foreshortening, scale change, wide baselines

• Cons
  • You only get a sparse cloud of points (or oriented patches), not a dense depth map or a complete surface
From feature matching to dense stereo

1. Extract features
2. Get a sparse set of initial matches
3. Iteratively expand matches to nearby locations
4. Use visibility constraints to filter out false matches
5. Perform surface reconstruction

From feature matching to dense stereo

http://www.cs.washington.edu/homes/furukawa/gallery/

Volumetric stereo

- In plane sweep stereo, the sampling of the scene still depends on the reference view.
- We can use a voxel volume to get a view-independent representation.
Volumetric stereo

Scene Volume: $V$

Input Images (Calibrated)

Goal: Determine occupancy, “color” of points in $V$
Discrete formulation: Voxel Coloring

Goal: Assign RGB values to voxels in $V$ photo-consistent with images
Photo-consistency

- A *photo-consistent scene* is a scene that exactly reproduces your input images from the same camera viewpoints.
- You can’t use your input cameras and images to tell the difference between a photo-consistent scene and the true scene.

Does not take advantage of epipolar geometry!
Stereo testing and comparisons


Scene

Ground truth
Decreasing order of overall performance (Fig. 17 plus an other row at the end)

Scharstein and Szeliski
Results with window correlation

Window-based matching  
(best window size)  
(Seitz)  

Ground truth
Graph Cuts

- Solution technique for general 2D problem

\[
E_{\text{total}}(d) = E_{\text{data}}(d) + \lambda E_{\text{smoothness}}(d)
\]

\[
E_{\text{data}}(d) = \sum_{x,y} f_{x,y}(d_{x,y})
\]

\[
E_{\text{smoothness}}(d) = \sum_{x,y} \rho(d_{x,y} - d_{x-1,y})
\]

\[
+ \sum_{x,y} \rho(d_{x,y} - d_{x,y-1})
\]

(a) original image  (b) observed image  (c) local min w.r.t. standard moves  (d) local min w.r.t. \(\alpha\)-expansion moves
Results with better method

State of the art method: Graph cuts

Ground truth

(Seitz)
Segmentation-based Stereo

Hai Tao and Harpreet W. Sawhney

Another Example
Stereo Correspondences
Stereo Correspondences

Left scanline

Right scanline

Match

Match

Match

Occlusion

Disocclusion
Search Over Correspondences

Three cases:
- Sequential – add cost of match (small if intensities agree)
- Occluded – add cost of no match (large cost)
- Disoccluded – add cost of no match (large cost)
Stereo Matching with Dynamic Programming

Dynamic programming yields the optimal path through grid. This is the best set of matches that satisfy the ordering constraint.
Dynamic Programming

- Efficient algorithm for solving sequential decision (optimal path) problems.

How many paths through this trellis? $3^T$
Dynamic Programming

Suppose cost can be decomposed into stages:

\[ \Pi_{ij} = \text{Cost of going from state } i \text{ to state } j \]
**Dynamic Programming**

\[ C_t(j) = \min_i (\Pi_{ij} + C_{t-1}(i)) \]

*Principle of Optimality* for an n-stage assignment problem:

- \( i = 1 \)
  - 1
  - \( \Pi_{12} \)
  - 1
  - 1
- \( i = 2 \)
  - 2
  - \( \Pi_{22} \)
  - \( \Pi_{32} \)
  - 2
  - 2
- \( i = 3 \)
  - 3
  - 3
  - 3

---

**cost from i to j**  
**minimum cost still state i**
Stereo Matching with Dynamic Programming

Scan across grid computing optimal cost for each node given its upper-left neighbors. Backtrack from the terminal to get the optimal path.
Stereo Matching with Dynamic Programming

Scan across grid computing optimal cost for each node given its upper-left neighbors. Backtrack from the terminal to get the optimal path.
Stereo Matching with Dynamic Programming

Scan across grid computing optimal cost for each node given its upper-left neighbors. Backtrack from the terminal to get the optimal path.

Occluded Pixels

Left scanline

Dis-occluded Pixels

Right scanline

Terminal

backtracing give the optimal path between the two sequences
Dynamic Programming

- Can we apply this trick in 2D as well?

No: $d_{x,y-1}$ and $d_{x-1,y}$ may depend on different values of $d_{x-1,y-1}$

Enforces ordering constraint. For narrow foreground object NOT the case since ordering can change.
Sub-Pixel Disparity:

The disparity is computed by moving a window one pixel at a time. As a result, the disparity is known only up to one pixel. This limitation on the resolution of the disparity translates into a severe limitation on the accuracy of the recovered 3-D coordinates. One effective way to get this problem is to recover the disparity at a finer resolution by interpolating between the pixel disparities using quadratic interpolation.

Suppose that the best disparity at a pixel is obtained at \( d_0 \) with a matching value (for example SSD) of \( S(d_0) \). We can obtain a second order approximation of the (unknown) function \( S(d) \) by approximating \( S \) by a parabola. At the position \( d_{\text{opt}} \) corresponding to the bottom of the parabola, we have \( S(d_{\text{opt}}) \leq S(d_0) \). Therefore, \( d_{\text{opt}} \) is a better estimate of the disparity than \( d_0 \).

The question remains as to how to find this approximating parabola. Let us first translate all the disparity values so that \( d_0 = 0 \). The equation of a general parabola is: \( S(d) = ad^2 + bd + c \). To recover the 3 parameters of the parabola we need 3 equations which we obtain by writing that the parabola passes through the point of disparity 0, -1, and +1:

\[
S(0) = c \quad S(1) = a + b + c \quad S(-1) = a - b + c
\]

Solving this, we obtain:

\[
c = S(0) \quad a = ( S(1) + S(-1) - 2 S(0))/2 \quad b = ( S(1) - S(-1))/2
\]

The bottom of the parabola is obtained at \( d_{\text{opt}} \) such that \( S'(d) = 2ad + b = 0 \). Therefore, the optimal disparity is obtained as:

\[
d_{\text{opt}} = \frac{(S(-1) - S(1))}{2(S(1) + S(-1) - 2S(0))}
\]

In practice, estimating disparity with a fraction of a pixel resolution is possible using this interpolation approach.

Note that the denominator is close to 0 if the function \( S \) is mostly flat, in which case there is no valid estimate of the disparity.
Laplacian of Gaussian (LOG):

A better way of eliminating smooth variations across the images is to take the second derivative, because the second derivative of a linear function of the form \( bx+c \) is 0, and therefore eliminates this component.

More generally, smoothly varying parts of the image do not carry much information for matching. The useful information is contained in higher-frequency variations of intensity. For example, in those graphs, only the pixels at the indicated locations carry useful information for matching. The smoothly varying part of the profile constitute the majority of the points and would confuse the matcher. The information that we really want to match is the fact that the intensity bumps at A and B have similar shape.

So we want to eliminate the slowly-varying parts of the image (low-frequency), and this can be done by using a second derivatives. Also, as usual, we want to eliminate the high-frequency components which correspond to noise in the image, which suggests blurring with a Gaussian filter. The combination of the two suggests the use of the Laplacian of Gaussian (LOG) previously defined in the context of edge detection. This filter is technically a band-pass filter (removes both high and low frequencies.) In practice, the images are first convoluted with the LOG filter:

\[
\nabla^2 G_\sigma * I_l \quad \text{and} \quad \nabla^2 G_\sigma * I_r
\]

Any of the definition of \( \psi \) can be used directly on the Laplacian image.
Laplacian Pyramids:

The effect of $\sigma$ is similar to the trade-off in edge detection:

Large $\sigma$: A lot of the noise is eliminated, the matching curves are smoother, and matching is less ambiguous. Therefore, we have better performance in terms of detection of matches.

Small $\sigma$: The image details are not blurred as much so the localization is better.

Just like for Gaussian smoothing, it is beneficial to perform stereo matching at different resolutions (i.e., different values of $\sigma$.) The algorithm would look like:

- Find the best disparity $d_{\sigma_k}(x,y)$ of every pixel at the current level of the image pyramid corresponding to $\sigma_k$ by searching over the interval $[d_{\min}(x,y), d_{\max}(x,y)]$.
- For every pixel, set $d_{\min}(x,y) = d_{\sigma_k}(x,y) - \delta$ and $d_{\max}(x,y) = d_{\sigma_k}(x,y) + \delta$.
- Repeat at the next pyramid level for $\sigma_{k+1} < \sigma_k$.

This is interesting for two reasons:

Computation: Matching is expensive. In this approach, the matching at coarse levels requires search over small intervals of disparity (because the image is small). At finer resolution, the search is still limited because we predict a small search interval from the previous level in the pyramid. Therefore, we never have to search the entire disparity interval in the full-resolution image.
Another way to view the use of >2 cameras is the “multibaseline” approach. Suppose that the three cameras are now aligned. A point P corresponds to three points in the images, aligned along the horizontal epipolar line. The disparities of those points $d_{12}$ and $d_{13}$ between the first image and images 2 and 3 are in general different because the baselines are different:

$$d_{12} = \frac{B_{12}}{Z}, \quad d_{13} = \frac{B_{13}}{Z}$$

However, the depth (Z) of the physical point in space is the same. Therefore:

$$\frac{1}{Z} = \frac{d_{12}}{B_{12}} = \frac{d_{13}}{B_{13}}$$

The implication is that: if we plot the matching curves ($S_{12}(d')$) not as a function of the disparity but at a function of $d' = d/B$, assuming that the matches are correct, all the curves should have the same minimum at $d' = 1/Z$.

In fact, instead of using $S_{12}$ and $S_{13}$ separately, we can just combine them into a single matching function: $S(d') = S_{12} + S_{13}$

and find the minimum of $S(d')$. Such an approach is called multi-baseline stereo.
Multiple-baseline stereo

- Pick a reference image, and slide the corresponding window along the corresponding epipolar lines of all other images, using inverse depth relative to the first image as the search parameter.

Multiple-baseline stereo

- For larger baselines, must search larger area in second image
Using a multibaseline approach combines the advantages of both short and long baseline:

The short baseline will make the matching easier but leads to poorer localization in space. The longer baseline leads to higher precision localization but more difficult matching. Furthermore, ambiguous matches would not generally appear at multiple baselines, thus, by combining the matching function into a single function, we get a sharper, unique peak of the matching function.

Occlusions become an issue for large baseline.
Multiple-baseline stereo

Use the sum of SSD scores to rank matches

Fig. 5. SSD values versus inverse distance: (a) $B = b$; (b) $B = 2b$; (c) $B = 3b$; (d) $B = 4b$; (e) $B = 5b$; (f) $B = 6b$; (g) $B = 7b$; (h) $B = 8b$. The horizontal axis is normalized such that $\mathcal{S}P = 1$.

Fig. 7. Combining multiple baseline stereo pairs.
Multiple-baseline stereo results

Summary of different stereo methods

• **Constraints:**
  – Geometry, epipolar constraint.
  – Photometric: Brightness constancy, only partly true.
  – Ordering: only partly true.
  – Smoothness of objects: only partly true.

• **Algorithms:**
  – What you compare: points, regions, features?

• **How you optimize:**
  – Local greedy matches.
  – 1D search.
  – 2D search.