"Video Mosaics for Virtual Environments," R. Szeliski

Review by:

Christopher Rasmussen

Announcements

- Homework due by midnight
- Next homework will be assigned Tuesday, due following Tuesday. It will involve tracking, specifically the Kalman filter, so look at Chap. 18-18.2 and handout for head start
- Some calendar changes
  - Project proposal due Oct. 10—details soon
  - Second paper presentations pushed back slightly
- Software for grabbing images
  - OS-based screen dump -> Crop
  - Software: SnagIt (Windows), scrot (Linux), etc.

Main Contributions

- Automatically construct image mosaics of…
  - Planar surfaces under general camera motion
  - General scene when camera motion is rotation about camera center
- Recover dense 3-D depth map (up to projective ambiguity)

Mosaicing: Big Issues

- Estimating homography without manually selecting point correspondences
  - Find displacement that best "registers" one image with the other
- Drawing overlapping images in a visually pleasing way

Primary Techniques

- **Nonlinear minimization** of SSD error function to estimate homography parameters
- **Fourier phase correlation** to estimate large camera motions
- **Multiple planes/cylindrical coordinates** to counteract large angle distortion for panoramic mosaicing
- **Projective depth variables** estimated for parallax motions

What We Want to Minimize

- Let **u**′ = **H u** represent homography transformation of a point **u** (Szeliski uses **M** for **H**). As shorthand, write transformation of all image points as **H I** = **I**′
- This is a linear system of equations; HW 1 was about solving the minimal form of it for known correspondences
- If we don't know any correspondences, what can we do? Idea: hypothesize **H**, transform image (using bilinear interpolation), see how good match is
- Szeliski uses SSD image similarity measure

$$E = \sum \left[I'(x', y') - I(x, y) \right]^2 = \sum e^2$$

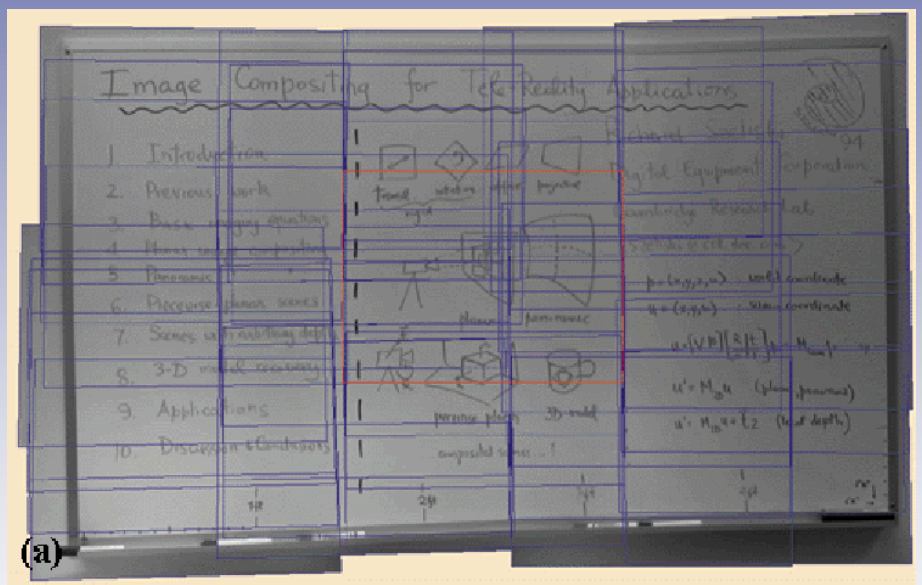
Nonlinear Minimization

- The SSD error function is nonlinear, so we can't use linear least-squares (e.g., SVD from homework) to solve for **H**
- Approach: Nonlinear least-squares to find the parameters of **H** which minimize the error function
- **Levenberg-Marquardt** is a standard technique (see Chap. 15.5 of Numerical Recipes for details)
- Initial guess: Identity homography
- Matlab: Pass error function to `lsqnonlin` after `optimset('LevenbergMarquardt','on')`

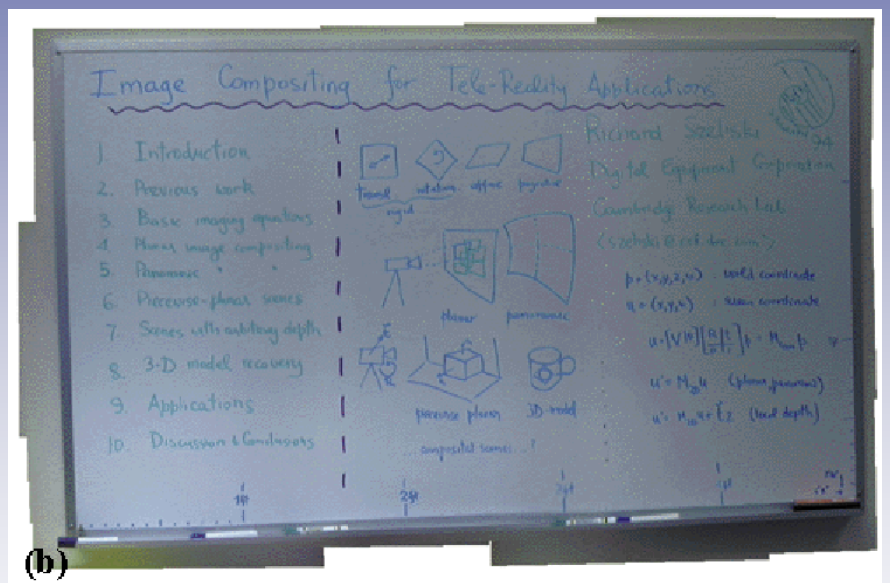
Image Compositing

- With homography computed, how to render combined image?
- Simply putting one image on top of the other, even with bilinear interpolation, may result in a "seam" due to different brightness levels
  - Auto-iris can change overall lightness of images
  - Vignetting can make image edges darker
- Bilinear blending: Use "hat" function *w* indicating weight of contributions of both images to mosaic
  - *w* maximal at source image center, falls linearly to 0 at edges

Planar Mosaic: Whiteboard



Raw images



Final mosaic

Dealing with Large Displacements

- For small overlaps, gradient descent/LM have problem with local minima in SSD error function. Possible solutions:
  - Hierarchical matching (i.e., image pyramid)
  - Phase correlation (for even less overlap)—uses Fourier transform
- Also, geometry of panoramic mosaicing invalidates single plane assumption over large angular changes

Fourier Transform

- Given a function $f(x, y)$, its Fourier transform $F(u, v)$ is defined by:

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-i(ux+vy)}dxdy$$

- Invertible decomposition of function (image) into waves of different frequencies (sines & cosines)
- Takes function from *spatial* domain to (complex) *frequency* domain

Fourier Theorem

- The convolution of two functions is the same as the product of their Fourier transforms
  - Given $g(x,y) = f(x,y) * h(x,y)$, we have that $G(u,v) = F(u,v)H(u,v)$
- Helpful way to convolve efficiently (less so for small kernels)
- Fast Fourier Transform (FFT): Numerical method for computing Fourier transform; computational complexity = $n \log n$.
  - Matlab: `fft2`; also, C library at `fftw.org`

What We Want To Know

- Need to estimate shift (x_0, y_0) between two images. This is like computing cross-correlation and finding peak
- Cross-power spectrum: Fourier transform of cross-correlation function

Fourier Shift Theorem

- Suppose $f_2(x,y) = f_1(x + x_0, y + y_0)$ and $F_1(u,v) = F_1(x,y)$. Then:

$$F_2(u,v) = e^{-2\pi i(ux_0 + vy_0)} F_1(u,v)$$

Estimating Shift

- Equivalently:

$$\frac{F_1(u,v)F_2^*(u,v)}{|F_1(u,v)F_2^*(u,v)|} = e^{+2\pi i(ux_0+vy_0)}$$

  where the LHS is the cross-power spectrum
- The inverse Fourier transform of the RHS is $\delta(x_0, y_0)$, so find the location of the maximum and we are done

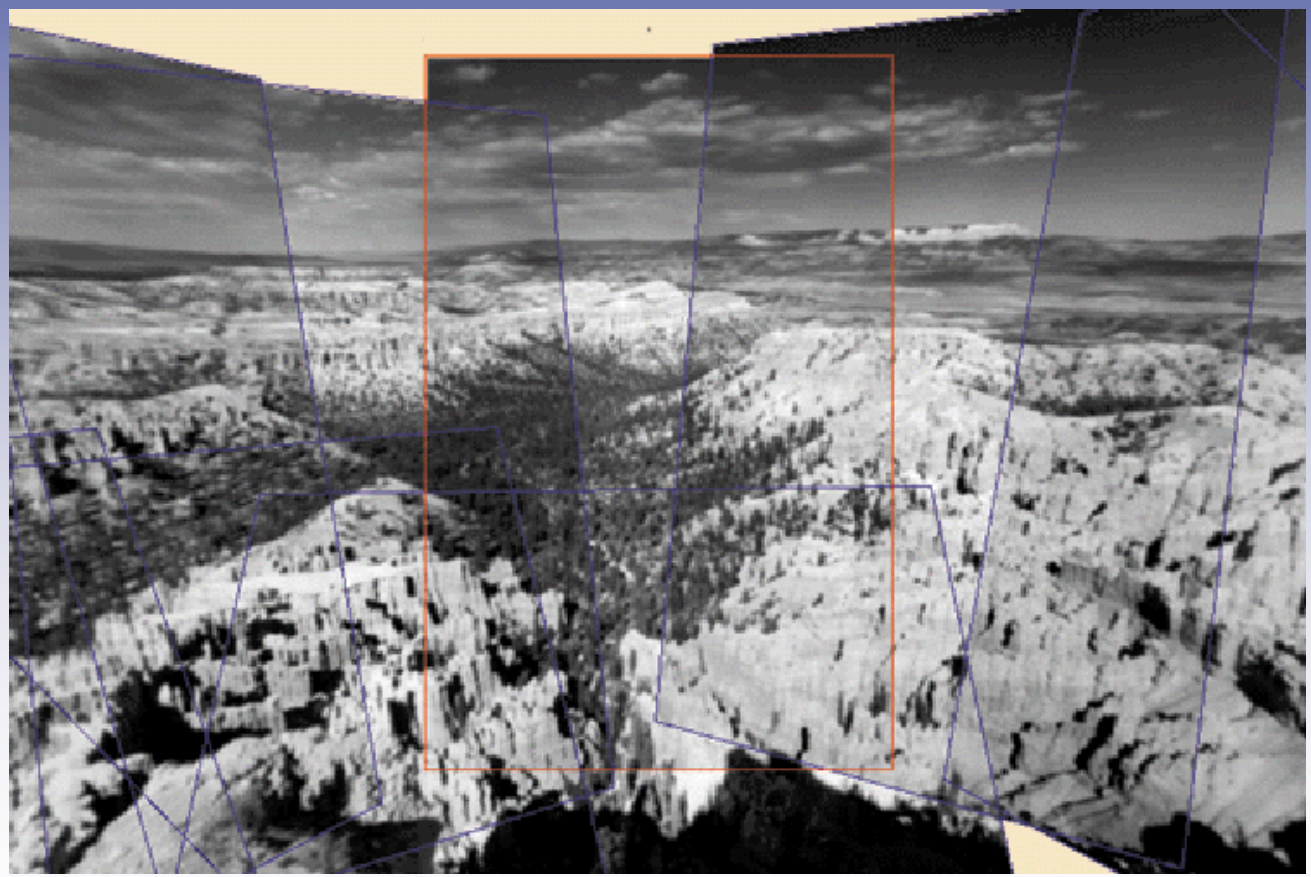
Extensions

- The preceding analysis can be extended to handle rotation and scaling
  - See B. Reddy and B. Chatterji, "An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration," IEEE Trans. Image Processing, Vol. 5, pp. 1266-1271, August, 1996.

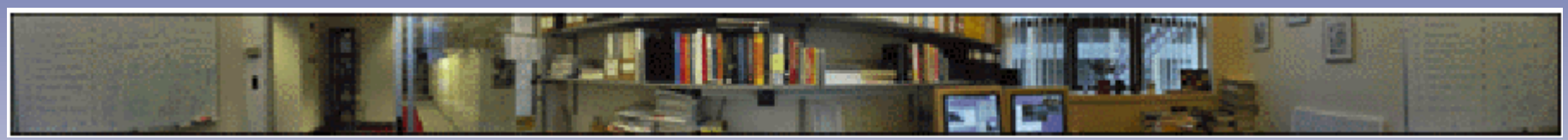
Panoramic Mosaics

- Rotation around camera center does not induce parallax $\rightarrow$ Locally, images are coplanar
- However, just one plane becomes bad approximation over large angles
- Techniques
  - Tile sphere with planes, register images to local plane
  - Register images to base frame; choose new base frame periodically
  - Map to cylindrical coordinates before registering

Results: Multiple Base Frames



Results: Cylindrical Coordinates



Office

River

Depth Recovery

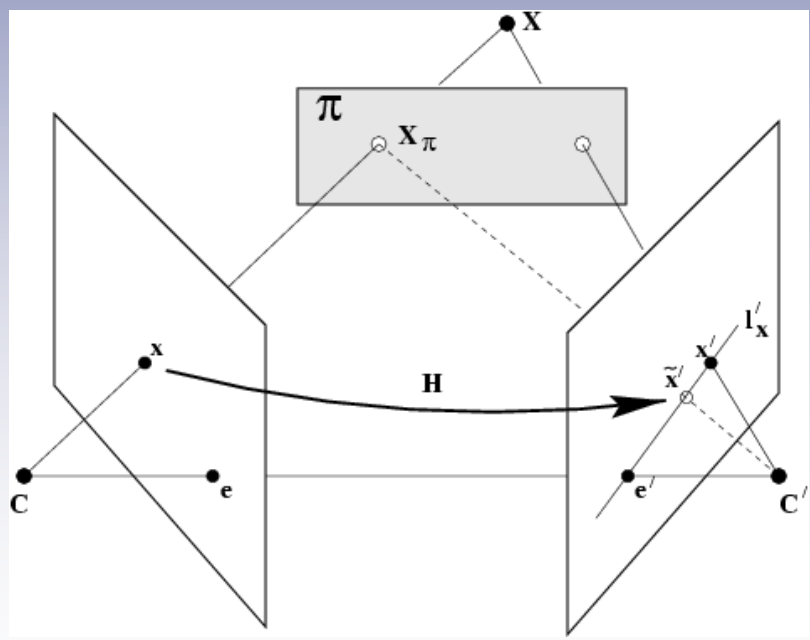
- Want to estimate Z (depth) values for scene points
- Benefits of 3-D structure
  - Obstacle detection without recognition
  - More information for recognition
- Approaches
  - Piecewise planar: suggested but not explained
  - Dense (per pixel)

Dense 3-D Depth Maps

- Calculating depths
  - Known camera motion $\rightarrow$ Stereo reconstruction
  - Unknown camera motion $\rightarrow$ Structure from motion
- Projective depth
  - Definition: *Parallax*, or image shift, of point between two views after accounting for homography. Proportional to relative depth from scene plane inducing homography
  - Not the same as traditional Euclidean depth

Projective Depth: Details

- Approach: Decompose motion of scene points into two parts:
  - 2-D homography (as if all points coplanar)
  - Parallax: Discrepancy proportional to distance from plane
- Apply nonlinear minimiz-ation as before to estimate new variables
- More next week in "Motion" lecture

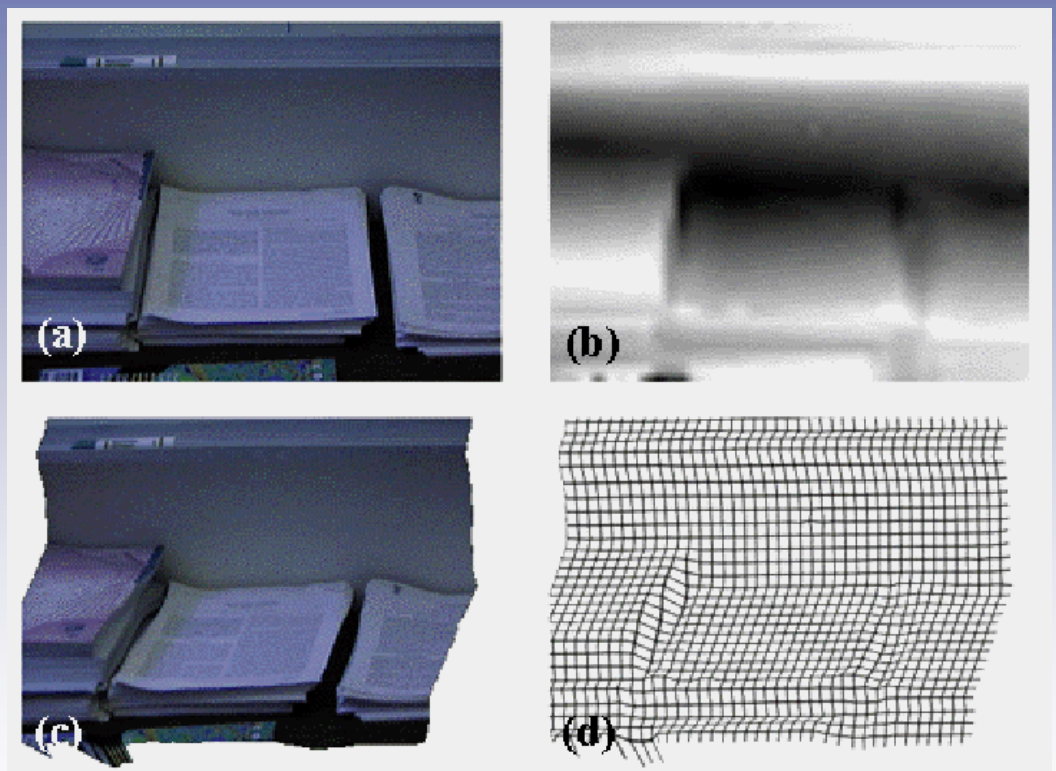


from Hartley & Zisserman

Results:
Desktop Depth Recovery

Raw image

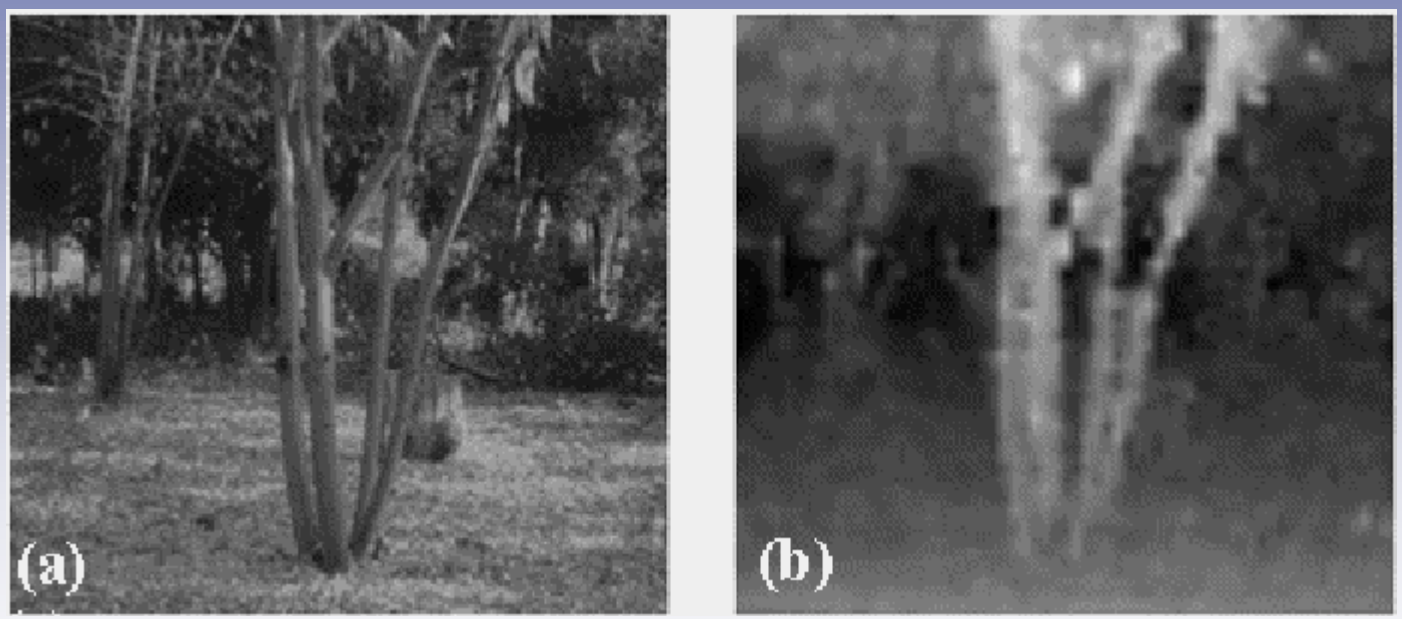
Estimated depths



Synthesized view (texture-mapped)

Synthesized view (wire-frame)

Results:
Tree Depth Recovery



Raw image

Estimated depths

Results

- Looks good qualitatively, but no quantitative analysis
- Could mosaic large, known poster to compare pixel by pixel accuracy
- Projective depth recovery could have imaged known scene to measure accuracy

Limitations

- Need scene to be highly-textured for accurate registration, projective depth estimation
- Sensitive to illumination variations, departures from pinhole model (i.e., radial distortion)
- Hat function for blending is ad-hoc—could we actually calibrate images photometrically (i.e., for exposure, vignetting) and compensate precisely?

More Limitations

- What if something in the scene is moving?
- Images only adjusted pairwise $\rightarrow$ errors propagate. How about a global algorithm estimates alignments of all images simultaneously?

Connections

- Techniques have wide applicability
  - Image registration useful for tracking, recognition
  - Fourier phase correlation can be used for audio localization—i.e., shift is in time of sound's arrival at different microphones, which is proportional to distance, allowing triangulation, etc.
- Related things we're reading…
  - Making panoramic mosaics for recognition (instead of using omnidirectional camera)
  - Mosaicing of sea floor by submersible
  - Building ceiling mosaic for museum robot

Robotics Applications/ Possible improvements

- Tracking planar patterns (e.g., signs, building facades, the ground from a UAV) over time
- Super-resolution for higher fidelity pictures—Can we make face recognition, OCR, etc. more accurate?
- Acquiring piecewise-planar 3-D building models automatically