Image Processing

Cosimo Distante

Lecture: Alignment
Today: Alignment

- Homographies
- Rotational Panoramas
- RANSAC
- Global alignment
- Warping
- Blending
Motivation: Recognition

Figures from David Lowe
Motivation: medical image registration
Motivation: Mosaics

• Getting the whole picture
  – Consumer camera: 50° x 35°
Motivation: Mosaics

• Getting the whole picture
  – Consumer camera: 50° x 35°
  – Human Vision: 176° x 135°
Motivation: Mosaics

• Getting the whole picture
  – Consumer camera: 50° x 35°
  – Human Vision: 176° x 135°

• Panoramic Mosaic = up to 360 x 180°

Slide from Brown & Lowe 2003
Motion models
Motion models

- What happens when we take two images with a camera and try to align them?
- translation?
- rotation?
- scale?
- affine?
- perspective?
- ... see interactive demo (VideoMosaic)
Image Warping
Image Warping

- image filtering: change *range* of image
  - \( g(x) = h(f(x)) \)

- image warping: change *domain* of image
  - \( g(x) = f(h(x)) \)
Image Warping

- image filtering: change range of image
  \[ g(x) = h(f(x)) \]

- image warping: change domain of image
  \[ g(x) = f(h(x)) \]
Parametric (global) warping

- Examples of parametric warps:
  - translation
  - rotation
  - aspect
  - affine
  - perspective
  - cylindrical
Image Warping

• Given a coordinate transform $x' = h(x)$ and a source image $f(x)$, how do we compute a transformed image $g(x') = f(h(x))$?
Forward Warping

• Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
• What if pixel lands “between” two pixels?
Forward Warping

• Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
• What if pixel lands “between” two pixels?
• Answer: add “contribution” to several pixels, normalize later (splatting)
Inverse Warping

• Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$

• What if pixel comes from “between” two pixels?
Inverse warping

Get each pixel \( g(x',y') \) from its corresponding location \( (x,y) = T^{-1}(x',y') \) in the first image.

Q: what if pixel comes from “between” two pixels?

A: Interpolate color value from neighbors
   - nearest neighbor, bilinear…

>> help interp2
Bilinear interpolation

Sampling at $f(x,y)$:

$$f(x, y) = (1 - a)(1 - b) \ f[i, j] + a(1 - b) \ f[i + 1, j] + ab \ f[i + 1, j + 1] + (1 - a)b \ f[i, j + 1]$$
Interpolation

• Possible interpolation filters:
  – nearest neighbor
  – bilinear
  – bicubic (interpolating)
  – sinc / FIR

• Needed to prevent “jaggies” and “texture crawl”
Prefiltering

• Essential for \textit{downsampling} (\textit{decimation}) to prevent \textit{aliasing}

• MIP-mapping [Williams’83]:
  1. build pyramid (but what decimation filter?):
     • block averaging
     • Burt & Adelson (5-tap binomial)
     • 7-tap wavelet-based filter (better)
  2. \textit{trilinear} interpolation
     • bilinear within each 2 adjacent levels
     • linear blend \textit{between} levels (determined by pixel size)
2D coordinate transformations

- translation: \( x' = x + t \) \( x = (x,y) \)
- rotation: \( x' = R x + t \)
- similarity: \( x' = s R x + t \)
- affine: \( x' = A x + t \)
- perspective: \( x' \cong H x \) \( x = (x,y,1) \)
  
  (\( x \) is a homogeneous coordinate)

- These all form a nested group (closed w/ inv.)
Basic 2D Transformations

Basic 2D transformations as 3x3 matrices

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Translate

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  \cos \Theta & -\sin \Theta & 0 \\
  \sin \Theta & \cos \Theta & 0 \\
  0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Rotate

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & 0 \\
  0 & s_x & 0 \\
  0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Scale

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} = \begin{bmatrix}
  1 & s_{hx} & 0 \\
  s_{hy} & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Shear

Source: Alyosha Efros
2D Affine Transformations

\[
\begin{bmatrix}
  x' \\
  y' \\
  w
\end{bmatrix} = \begin{bmatrix}
  a & b & c \\
  d & e & f \\
  0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  w
\end{bmatrix}
\]

Affine transformations are combinations of …

• Linear transformations, and

• Translations

Parallel lines remain parallel
Projective Transformations

\[
\begin{bmatrix}
  x' \\
  y' \\
  w'
\end{bmatrix} = \begin{bmatrix}
  a & b & c \\
  d & e & f \\
  g & h & i
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  w
\end{bmatrix}
\]

Projective transformations:
- Affine transformations, and
- Projective warps

Figure 3.44
Image warping involves modifying the domain of an image function rather than its range.

Figure 3.45
Basic set of 2D geometric image transformations.

Here we look at functions that transform the domain, \( g(x) = f(h(x)) \) (3.88) (see Figure 3.44).

We begin by studying the global parametric 2D transformation first introduced in Section 2.1.2. (Such a transformation is called parametric because it is controlled by a small number of parameters.) We then turn our attention to more local general deformations such as those defined on meshes (Section 3.6.2). Finally, we show how image warps can be combined with cross-dissolves to create interesting morphs (in-between animations) in Section 3.6.3.

For readers interested in more details on these topics, there is an excellent survey by Heckbert (1986) as well as very accessible textbooks by Wolberg (1990), Gomes, Darsa, Costa et al. (1999) and Akenine-Möller and Haines (2002). Note that Heckbert's survey is on texture mapping, which is how the computer graphics community refers to the topic of warping images onto surfaces.

3.6.1 Parametric transformations
Parametric transformations apply a global deformation to an image, where the behavior of the transformation is controlled by a small number of parameters.
• Two broad approaches:
  – Direct (pixel-based) alignment
    • Search for alignment where most pixels agree
  – Feature-based alignment
    • Search for alignment where extracted features agree
    • Can be verified using pixel-based alignment

Source: L. Lazebnik
Fitting an affine transformation

Affine model approximates perspective projection of planar objects.

Figures from David Lowe, ICCV 1999
Fitting an affine transformation

- Assuming we know the correspondences, how do we get the transformation?

\[
\begin{bmatrix}
    x_i' \\
    y_i'
\end{bmatrix}
= \begin{bmatrix}
    m_1 & m_2 \\
    m_3 & m_4
\end{bmatrix}
\begin{bmatrix}
    x_i \\
    y_i
\end{bmatrix}
+ \begin{bmatrix}
    t_1 \\
    t_2
\end{bmatrix}
\]
Fitting an affine transformation

- Assuming we know the correspondences, how do we get the transformation?

\[
\begin{bmatrix}
  x_i' \\
  y_i'
\end{bmatrix} = \begin{bmatrix}
  m_1 & m_2 \\
  m_3 & m_4
\end{bmatrix} \begin{bmatrix}
  x_i \\
  y_i
\end{bmatrix} + \begin{bmatrix}
  t_1 \\
  t_2
\end{bmatrix}
\]
Fitting an affine transformation

\[
\begin{bmatrix}
  x_i & y_i & 0 & 0 & 1 & 0 \\
  0 & 0 & x_i & y_i & 0 & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\begin{bmatrix}
  m_1 \\
  m_2 \\
  m_3 \\
  m_4 \\
  t_1 \\
  t_2 \\
\end{bmatrix}
= \begin{bmatrix}
  x'_i \\
  y'_i \\
  \vdots \\
\end{bmatrix}
\]

- How many matches (correspondence pairs) do we need to solve for the transformation parameters?
- Once we have solved for the parameters, how do we compute the coordinates of the corresponding point for \( (x_{new}, y_{new}) \)?
Panoramas

Obtain a wider angle view by combining multiple images.

image from S. Seitz
How to stitch together a panorama?

• Basic Procedure
  – Take a sequence of images from the same position
    • Rotate the camera about its optical center
  – Compute transformation between second image and first
  – Transform the second image to overlap with the first
  – Blend the two together to create a mosaic
  – (If there are more images, repeat)

• …but wait, why should this work at all?
  – What about the 3D geometry of the scene?
  – Why aren’t we using it?

Source: Steve Seitz
Panoramas: generating synthetic views

Can generate any synthetic camera view as long as it has the same center of projection!

Source: Alyosha Efros
The mosaic has a natural interpretation in 3D

- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*

Source: Steve Seitz
Homography

How to relate two images from the same camera center?
- how to map a pixel from PP1 to PP2?

Think of it as a 2D **image warp** from one image to another.

A projective transform is a mapping between any two PPs with the same center of projection
- rectangle should map to arbitrary quadrilateral
- parallel lines aren’t
- but must preserve straight lines

Called **Homography**

\[
\begin{bmatrix}
w x' \\
w y' \\
w \\
p'
\end{bmatrix} = \begin{bmatrix}
* & * & * & x \\
* & * & * & y \\
* & * & * & 1 \\
1 & 1 & 1 & 1
\end{bmatrix} \begin{bmatrix}
p \\
p
\end{bmatrix}
\]

Source: Alyosha Efros
To apply a given homography $H$

- Compute $p' = Hp$ (regular matrix multiply)
- Convert $p'$ from homogeneous to image coordinates

$$\begin{bmatrix}
wx' \\
wv' \\
w \\
p'
\end{bmatrix} = \begin{bmatrix}
* & * & * \\
* & * & * \\
* & * & * \\
1 & 0 & 0
\end{bmatrix} \begin{bmatrix}
x \\
y \\
w \\
p
\end{bmatrix}$$
To compute the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of $H$ are the unknowns...
Solving for homographies

\[ p' = Hp \]
\[
\begin{bmatrix}
wx' \\
wz' \\
w
\end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\]

Can set scale factor \( i = 1 \). So, there are 8 unknowns.
Set up a system of linear equations:
\[ Ah = b \]
where vector of unknowns \( h = [a, b, c, d, e, f, g, h]^T \)
Need at least 8 eqs, but the more the better…
Solve for \( h \). If overconstrained, solve using least-squares:
\[
\min \| Ah - b \|^2
\]

>> help lmddivde
Recap: How to stitch together a panorama?

• Basic Procedure
  – Take a sequence of images from the same position
    • Rotate the camera about its optical center
  – Compute transformation between second image and first
  – Transform the second image to overlap with the first
  – Blend the two together to create a mosaic
  – (If there are more images, repeat)
Image warping with homographies

image plane in front

black area where no pixel maps to

Source: Steve Seitz
Image rectification
Analysing patterns and shapes

What is the shape of the b/w floor pattern?

The floor (enlarged)

Automatically rectified floor

Slide from Criminisi
Analysing patterns and shapes

From Martin Kemp *The Science of Art* (manual reconstruction)

Slide from Criminisi
Automatically rectified floor

Analysing patterns and shapes

What is the (complicated) shape of the floor pattern?

St. Lucy Altarpiece, D. Veneziano

Slide from Criminisi
Analysing patterns and shapes

Automatic rectification

From Martin Kemp, *The Science of Art* (manual reconstruction)

Slide from Criminisi
changing camera center

Does it still work?

Source: Alyosha Efros
Planar scene (or far away)

PP3 is a projection plane of both centers of projection, so we are OK!

This is how big aerial photographs are made

Source: Alyosha Efros
Outliers

- **Outliers** can hurt the quality of our parameter estimates, e.g.,
  - an erroneous pair of matching points from two images
  - an edge point that is noise, or doesn’t belong to the line we are fitting.
Example: least squares line fitting

• Assuming all the points that belong to a particular line are known
Outliers affect least squares fit
Outliers affect least squares fit
RANSAC

• RANdom Sample Consensus

• Approach: we want to avoid the impact of outliers, so let’s look for “inliers”, and use those only.

• Intuition: if an outlier is chosen to compute the current fit, then the resulting line won’t have much support from rest of the points.
RANSAC

- **RANSAC loop:**
  1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
  2. Compute transformation from seed group
  3. Find *inliers* to this transformation
  4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers
RANSAC Line Fitting Example

Task:
Estimate best line

Slide credit: Jinxiang Chai, CMU
RANSAC Line Fitting Example

Sample two points
RANSAC Line Fitting Example
RANSAC Line Fitting Example

Total number of points within a threshold of line.
RANSAC Line Fitting Example

Repeat, until get a good result
RANSAC Line Fitting Example

Repeat, until get a good result
RANSAC Line Fitting Example

Repeat, until get a good result
RANSAC example: Translation

Putative matches

Source: Rick Szeliski
RANSAC example: Translation

Select one match, count inliers
RANSAC example: Translation

Select one match, count inliers
RANSAC example: Translation

Find “average” translation vector
Feature-based alignment outline

Source: L. Lazebnik
Feature-based alignment outline

- Extract features

Source: L. Lazebnik
Feature-based alignment outline

- Extract features
- Compute *putative matches*
Feature-based alignment outline

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation $T$ (small group of putative matches that are related by $T$)
Feature-based alignment outline

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation $T$ (small group of putative matches that are related by $T$)
  - *Verify* transformation (search for other matches consistent with $T$)

Source: L. Lazebnik
Feature-based alignment outline

• Extract features
• Compute *putative matches*
• Loop:
  • *Hypothesize* transformation $T$ (small group of putative matches that are related by $T$)
  • *Verify* transformation (search for other matches consistent with $T$)

Source: L. Lazebnik
Towards large-scale mosaics...
Motion models

- Translation: 2 unknowns
- Affine: 6 unknowns
- Perspective: 8 unknowns
- 3D rotation: 3 unknowns
Plane perspective mosaics

- 8-parameter homographies
- **Limitations:**
  - local minima
  - slow convergence
  - difficult to control interactively
Rotational mosaics

– Directly optimize rotation and focal length

– Advantages:
  • ability to build full-view panoramas
  • easier to control interactively
  • more stable and accurate estimates
3D → 2D Perspective Projection

\[
\begin{bmatrix}
X_c \\
Y_c \\
Z_c
\end{bmatrix} = 
\begin{bmatrix}
R
\end{bmatrix}_{3 \times 3}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} + t
\]

\[
\begin{bmatrix}
u \\
v \\
1
\end{bmatrix} \sim 
\begin{bmatrix}
U \\
V \\
W
\end{bmatrix} = 
\begin{bmatrix}
f & 0 & u_c \\
0 & f & v_c \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
X_c \\
Y_c \\
Z_c
\end{bmatrix}
\]
Rotational mosaic

• Projection equations

1. Project from image to 3D ray
• \((x_0, y_0, z_0) = (u_0-u_c, v_0-v_c, f)\)

2. Rotate the ray by camera motion
• \((x_1, y_1, z_1) = R_{01} (x_0, y_0, z_0)\)

3. Project back into new (source) image
• \((u_1, v_1) = (fx_1/z_1+u_c, fy_1/z_1+v_c)\)
Establishing correspondences

1. ‘Direct’ method: \textit{(more next week)}
   - Use generalization of affine motion model
     [Szeliski & Shum ’97]

2. Feature-based method
   - Extract features, match, find consistent \textit{inliers}
   - Compute $R$ from correspondences
     (absolute orientation)
Absolute orientation

[Arun et al., PAMI 1987] [Horn et al., JOSA A 1988]
Procrustes Algorithm [Golub & VanLoan]

Given two sets of matching points, compute R

- \( p_i' = R p_i \)  
  
  3D rays
- \( A = \sum_i p_i p_i^T = \sum_i p_i p_i^T R^T = U S V^T = (U S U^T) R^T \)
- \( V^T = U^T R^T \)
- \( R = V U^T \)
Stitching demo
Panoramas

• What if you want a 360° field of view?

mosaic Projection Cylinder
Cylindrical panoramas

• Steps
  – Reproject each image onto a cylinder
  – Blend
  – Output the resulting mosaic
Cylindrical Panoramas

- Map image to cylindrical or spherical coordinates
  - need *known* focal length

Image 384x300  \( f = 180 \) (pixels)  \( f = 280 \)  \( f = 380 \)
Determining the focal length

1. Initialize from homography $H$
   (see text or [SzSh’97])
2. Use camera’s EXIF tags (approx.)
3. Use a tape measure

4. Ask your instructor
Cylindrical projection

\[(X, Y, Z) \rightarrow \text{Map 3D point (X,Y,Z) onto cylinder}\]

\[
(\tilde{x}, \tilde{y}, \tilde{z}) = \frac{1}{\sqrt{X^2 + Z^2}}(X, Y, Z)
\]

- Convert to cylindrical coordinates

\[
(sin\theta, h, cos\theta) = (\tilde{x}, \tilde{y}, \tilde{z})
\]

- Convert to cylindrical image coordinates

\[
(\tilde{x}, \tilde{y}) = (s\theta, sh) + (\tilde{x}_c, \tilde{y}_c)
\]

- \(s\) defines size of the final image
Cylindrical warping

- Given focal length $f$ and image center $(x_c, y_c)$

\[
\begin{align*}
\theta &= \frac{(x_{cyl} - x_c)}{f} \\
h &= \frac{(y_{cyl} - y_c)}{f} \\
\hat{x} &= \sin \theta \\
\hat{y} &= h \\
\hat{z} &= \cos \theta \\
x &= f\hat{x}/\hat{z} + x_c \\
y &= f\hat{y}/\hat{z} + y_c
\end{align*}
\]
Spherical warping

• Given focal length $f$ and image center $(x_c, y_c)$

\[
\begin{align*}
\theta &= \left( x_{cyl} - x_c \right) / f \\
\varphi &= \left( y_{cyl} - y_c \right) / f \\
\hat{x} &= \sin \theta \cos \varphi \\
\hat{y} &= \sin \varphi \\
\hat{z} &= \cos \varphi \cos \varphi \\
x &= f \hat{x} / \hat{z} + x_c \\
y &= f \hat{y} / \hat{z} + y_c
\end{align*}
\]
3D rotation

- Rotate image before placing on unrolled sphere

\[
\begin{align*}
\theta &= \frac{x_{cyl} - x_c}{f} \\
\varphi &= \frac{y_{cyl} - y_c}{f} \\
\hat{x} &= \sin \theta \cos \varphi \\
\hat{y} &= \sin \varphi \\
\hat{z} &= \cos \theta \cos \varphi \\
x &= f\hat{x}/\hat{z} + x_c \\
y &= f\hat{y}/\hat{z} + y_c \\
p &= R p
\end{align*}
\]
Radial distortion

- Correct for “bending” in wide field of view lenses

\[
\begin{align*}
\hat{r}^2 &= \hat{x}^2 + \hat{y}^2 \\
\hat{x}' &= \hat{x} / (1 + \kappa_1 \hat{r}^2 + \kappa_2 \hat{r}^4) \\
\hat{y}' &= \hat{y} / (1 + \kappa_1 \hat{r}^2 + \kappa_2 \hat{r}^4) \\
x &= f \hat{x}' / \hat{z} + x_c \\
y &= f \hat{y}' / \hat{z} + y_c
\end{align*}
\]
Radial distortion

• Correct for “bending” in wide field of view lenses

Figure 6.8 Calibration patterns: (a) a three-dimensional target (Quan and Lan 1999) © 1999 IEEE; (b) a two-dimensional target (Zhang 2000) © 2000 IEEE. Note that radial distortion needs to be removed from such images before the feature points can be used for calibration.
Fisheye lens

- Extreme “bending” in ultra-wide fields of view

\[ \hat{r}^2 = \hat{x}^2 + \hat{y}^2 \]

\[( \cos \theta \sin \phi, \sin \theta \sin \phi, \cos \phi ) = s (x, y, z)\]

Equations become

\[ x' = s\phi \cos \theta = s \frac{x}{r} \tan^{-1} \frac{r}{z}, \]

\[ y' = s\phi \sin \theta = s \frac{y}{r} \tan^{-1} \frac{r}{z}, \]
Image Stitching

1. Align the images over each other
   - camera pan ↔ translation on cylinder
2. Blend the images together
Assembling the panorama

- Stitch pairs together, blend, then crop
Problem: Drift

- Error accumulation
  - small (vertical) errors accumulate over time
  - apply correction so that sum = 0 (for 360° pan.)
Problem: Drift

• Solution
  – add another copy of first image at the end
  – this gives a constraint: \( y_n = y_1 \)
  – there are a bunch of ways to solve this problem
    • add displacement of \((y_1 - y_n)/(n -1)\) to each image after the first
    • compute a global warp: \( y' = y + ax \)
    • run a big optimization problem, incorporating this constraint
      – best solution, but more complicated
      – known as “bundle adjustment”
Full-view (360° spherical) panoramas
Full-view Panorama
Texture Mapped Model
Global alignment

- Register *all* pairwise overlapping images
- Use a 3D rotation model (one R per image)
- Use direct alignment (patch centers) or feature based
- *Infer* overlaps based on previous matches (incremental)
- Optionally *discover* which images overlap other images using feature selection (RANSAC)
Bundle adjustment formulations

All pairs optimization:

\[ E_{\text{all-pairs-2D}} = \sum_{i} \sum_{jk} c_{ij} c_{ik} \| \tilde{x}_{ik} \left( \tilde{x}_{ij}; R_j, f_j, R_k, f_k \right) - \hat{x}_{ik} \|^2, \quad (9.29) \]

Full bundle adjustment, using 3-D point positions \( \{x_i\} \)

\[ E_{\text{BA-2D}} = \sum_{i} \sum_{j} c_{ij} \| \tilde{x}_{ij} \left( x_i; R_j, f_j \right) - \hat{x}_{ij} \|^2, \quad (9.30) \]

Bundle adjustment using 3-D ray:

\[ E_{\text{BA-3D}} = \sum_{i} \sum_{j} c_{ij} \| \tilde{x}_{ij} \left( \tilde{x}_{ij}; R_j, f_j \right) - x_i \|^2, \quad (9.31) \]

All-pairs 3-D ray formulation:

\[ E_{\text{all-pairs-3D}} = \sum_{i} \sum_{jk} c_{ij} c_{ik} \| \tilde{x}_{ij} \left( \tilde{x}_{ij}; R_j, f_j \right) - \tilde{x}_{ik} \left( \tilde{x}_{ik}; R_k, f_k \right) \|^2. \quad (9.32) \]
Recognizing Panoramas

Matthew Brown & David Lowe

ICCV’2003
Recognizing Panoramas

[Brown & Lowe, ICCV'03]
Finding the panoramas
Finding the panoramas
Finding the panoramas
Finding the panoramas
Fully automated 2D stitching demo

Windows Live Photo Gallery
Easily manage and share your photos and videos

Get it free

Overview | Features | System Requirements

Easily share your photos
The “Publish” button makes it simple to share your photos and videos online. Or you can easily e-mail as many photos as you'd like to friends and family. You can also display your photos with cool screen savers and slideshows.

Quickly find and organize your photos and videos
Import your photos from your digital camera; the Windows Live Photo Gallery will automatically organize them based on date and time. Keep your images organized by name, date, rating, and type. Locate similar photos with tags you add.

Enhance your photos
Create a cool panoramic view by combining multiple photos. Capture the moment by adding captions. Enhance your photos by adjusting things like color and exposure. Improve your photos with simple crop and red-eye fixes.

http://get.live.com/photogallery/overview
Rec.pano.: system components

1. Feature detection and description
   – more uniform point density
2. Fast matching (hash table)
3. RANSAC filtering of matches
4. Intensity-based verification
5. Incremental bundle adjustment

Multi-Scale Oriented Patches

• Interest points
  – Multi-scale Harris corners
  – Orientation from blurred gradient
  – Geometrically invariant to similarity transforms

• Descriptor vector
  – Bias/gain normalized sampling of local patch (8x8)
  – Photometrically invariant to affine changes in intensity
Features

• Distribute points evenly over the image
Descriptor Vector

• Orientation = blurred gradient
• Similarity Invariant Frame
  – Scale-space position \((x, y, s)\) + orientation \((\theta)\)
Probabilistic Feature Matching
RANSAC motion model
RANSAC motion model
RANSAC motion model
Probabilistic model for verification
How well does this work?

Test on 100s of examples...
How well does this work?

Test on 100s of examples...

...still too many failures (5-10%) for consumer application

Szeliski
Matching Mistakes: False Positive
Matching Mistakes: False Positive
Matching Mistake: False Negative

• Moving objects: large areas of disagreement
Matching Mistakes

• Accidental alignment
  – repeated / similar regions
• Failed alignments
  – moving objects / parallax
  – low overlap
  – “feature-less” regions
    (more variety?)
• No 100% reliable algorithm?
How can we fix these?

• Tune the feature detector
• Tune the feature matcher (cost metric)
• Tune the RANSAC stage (motion model)
• Tune the verification stage
• Use “higher-level” knowledge
  – e.g., typical camera motions

• Sounds like a big “learning” problem
  – Need a large training/test data set (panoramas)
Image Blending
Image feathering

- Weight each image proportional to its distance from the edge
  (distance map [Danielsson, CVGIP 1980]

1. Generate weight map for each image
2. Sum up all of the weights and divide by sum: weights sum up to 1:
   \[ w_i' = w_i / (\sum_i w_i) \]
Image Feathering
Feathering

Szeliski
Effect of window size

Szeliski
Effect of window size
Good window size

“Optimal” window: smooth but not ghosted
  • Doesn’t always work...
Pyramid Blending

Burt, P. J. and Adelson, E. H.,
A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.
Laplacian image blend

1. Compute Laplacian pyramid
2. Compute Gaussian pyramid on weight image (can put this in A channel)
3. Blend Laplacians using Gaussian blurred weights
4. Reconstruct the final image
   • Q: How do we compute the original weights?
   • A: For horizontal panorama, use *mid-lines*
   • Q: How about for a general “3D” panorama?
Weight selection (3D panorama)

- Idea: use original feather weights to select strongest contributing image

- Can be implemented using L-\(\infty\) norm: \((p = 10)\)
  
  \[
  w_i' = \left[ \frac{w_i^p}{\left( \sum_i w_i^p \right)} \right]^{1/p}
  \]
Poisson Image Editing

• Blend the gradients of the two images, then integrate
• For more info: Perez et al, SIGGRAPH 2003
De-Ghosting
Local alignment (deghosting)

• Use local optic flow to compensate for small motions [Shum & Szeliski, ICCV’98]

Figure 3: Deghosting a mosaic with motion parallax: (a) with parallax; (b) after single deghosting step (patch size 32); (c) multiple steps (sizes 32, 16 and 8).
Local alignment (deghosting)

• Use local optic flow to compensate for radial distortion [Shum & Szeliski, ICCV’98]

Figure 4: Deghosting a mosaic with optical distortion: (a) with distortion; (b) after multiple steps.
Region-based de-ghosting

• Select only one image in regions-of-difference using weighted vertex cover [Uyttendaele et al., CVPR’01]

Figure 5 – (A) Ghosted mosaic. (B) Result of de-ghosting algorithm.
Region-based de-ghosting

- Select only one image in regions-of-difference using weighted vertex cover [Uyttendaele et al., CVPR’01]