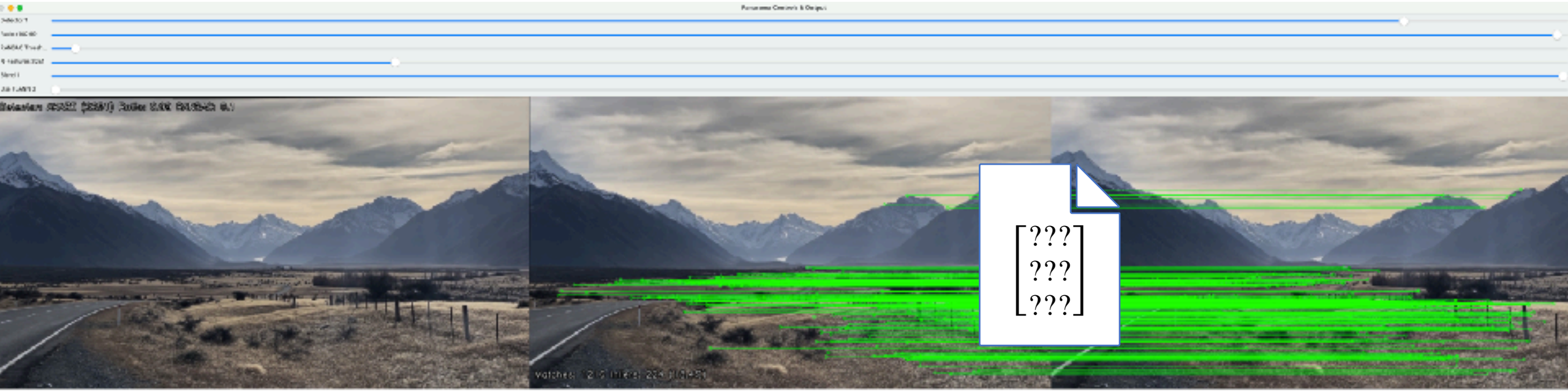
Visual Computing I:

Interactive Computer Graphics and Vision



Homographies, RANSAC, Blending

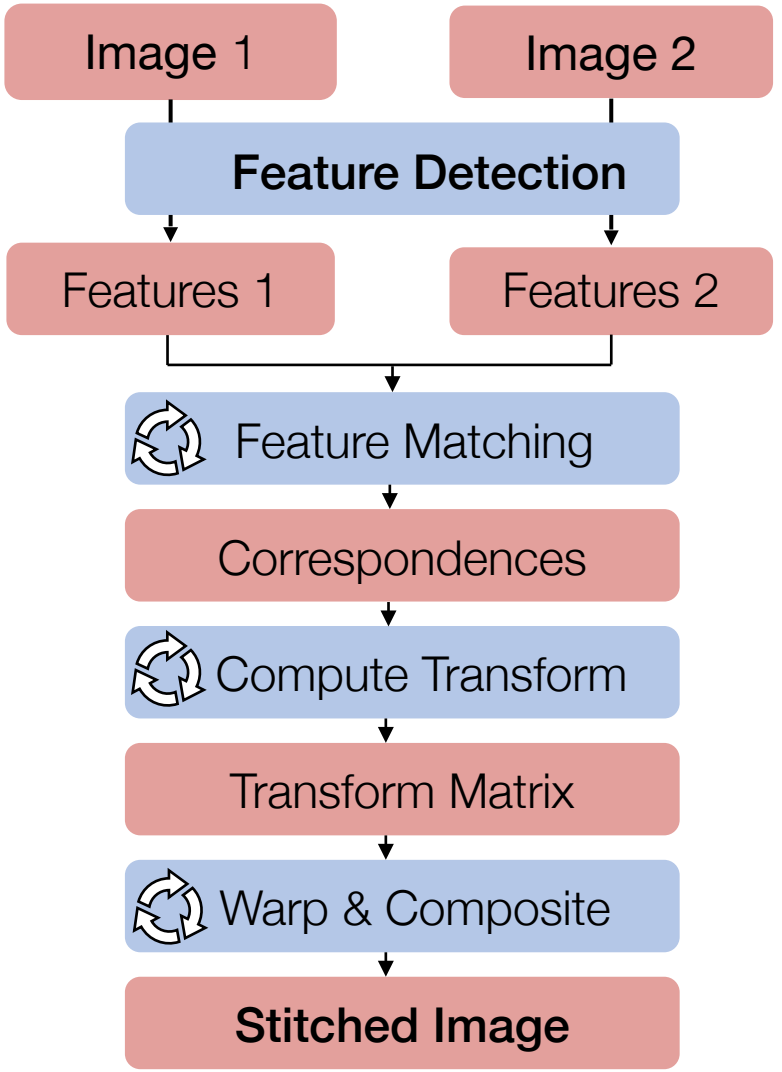
Stefanie Zollmann and Tobias Langlotz

Last time..

How to find correspondences?

Image Stitching Process

- Four main stages:
  1. Detect features in each image
  2. Match features between images
  3. Estimate a transform
  4. Warp one image to the other
- Stitching only works if:
  - The scene is (almost) planar, or
  - The camera only (mostly) rotates



Homographies and Image Stitching

- To stitch an image:
  - Need to warp images to align
  - This warping is a homography

- How to find the homography?
  - If a feature at (u,v) in one image matches to (u',v') in the other, then

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Demo…

Solving from Point Correspondences

- Suppose we have n matching points between two images
  $(u_1, v_1) \leftrightarrow (u_1', v_1'), (u_2, v_2) \leftrightarrow (u_2', v_2'), \ldots (u_n, v_n) \leftrightarrow (u_n', v_n')$
- Each pair gives us information about the homography, H
  $x_i' \equiv \mathrm{H} x_i$
- The equivalence makes things difficult:
  - Everything has an unknown scale factor associated with it
  - Can't solve things uniquely – if H is a solution, so is
    $k\mathrm{H}, \ \forall k \neq 0$

Solving from Point Correspondences

- Remember: We want to use the relationship: $x' = \lambda Hx$ where
  - x,x′ are homogeneous 2D points (3×1),
  - H is the homography (3×3),
  - λ is an unknown scale factor
- The problem: λ is different for each correspondence, so it's not something we can just solve for globally
- The trick: Equation $x' = \lambda Hx$ means the two vectors x′ and Hx are parallel in 3D (homogeneous coordinate space) -> In other words, one is a scaled version of the other
  - This means their cross product is 0: $x' \times (Hx) = 0$
  - No λ -> no worry -> Each correspondence provides 2 independent linear equations in the entries of H
  - Need at least 4 correspondences to solve for H (8 dof)

Lets build the equation system

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \times \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \times \begin{bmatrix} h_1u + h_2v + h_3 \\ h_4u + h_5v + h_6 \\ h_7u + h_8v + h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Remember: Matrix/Vector Multiplication

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \end{bmatrix}$$

Remember: cross product

$$\mathbf{x} \times \mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \times \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_2y_3 - x_3y_2 \\ x_3y_1 - x_1y_3 \\ x_1y_2 - x_2y_1 \end{bmatrix}$$

$$v'(h_7u + h_8v + h_9) - (h_4 + h_5 + h_6) = 0$$

$$(h_1u + h_2v + h_3) - u'(h_7y + h_8v + h_9) = 0$$

$$u'(h_4u + h_5v + h_6) - v'(h_1u + h_2v + h_3) = 0$$

$$\begin{bmatrix} v'(h_7u + h_8v + h_9) - (h_4 + h_5 + h_6) \\ (h_1u + h_2v + h_3) - u'(h_7y + h_8v + h9) \\ u'(h_4u + h_5v + h_6) - v'(h_1u + h_2v + h_3) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Finding the Homography

$$\begin{bmatrix} v_i'(h_7u_i + h_8v_i + h_9) - (h_4u_i + h_5v_i + h_6) \\ (h_1u_i + h_2v_i + h_3) - u_i'(h_7u_i + h_8v_i + h_9) \\ u_i'(h_4u_i + h_5v_i + h_6) - v_i'(h_1u_i + h_2v_i + h_3) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

- This looks like three equations, but not independent
- Use the first two (for symmetry) and rewrite as

$$\begin{bmatrix} 0 & 0 & 0 & -u_i & -v_i & -1 & u_iv_i' & v_iv_i' & v_i' \\ u_i & v_i & 1 & 0 & 0 & 0 & -u_iu_i' & -v_iu_i' & -u_i' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Finding the Homography

- We can now stack up all the points in one equation, $\mathbf{Ah} = \mathbf{0}$

$$\begin{bmatrix} 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1v'_1 & v_1v'_1 & v'_1 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1u'_1 & -v_1u'_1 & -u'_1 \\ 0 & 0 & 0 & -u_2 & -v_2 & -1 & u_2v'_2 & v_2v'_2 & v'_2 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2u'_2 & -v_2u'_2 & -u'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_nv'_n & v_nv'_n & v'_n \\ u_n & v_n & 1 & 0 & 0 & 0 & -u_nu'_n & -v_nu'_n & -u'_n \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Direct Linear Transform (DLT)

- One solution is $\mathbf{h} = \mathbf{0}$
  - This is the trivial solution
  - It is not useful to us
- Eigenvectors help us here

  $Ax = \lambda x$

- Applied to our problem:

  $Ah = 0 = 0h$

When you multiply A by x, it doesn't change the direction of x; it only stretches or squashes it by a factor of λ. If λ = 0, x gets flattened to 0.

We also say that "h lies in the null space of A"

- So we're looking for an eigenvector with a zero eigenvalue

Direct Linear Transform (DLT)

- Four points is sufficient
  - H has 9 values, up to a scale, so 8 degrees of freedom
  - More points gives a least-squares solution
- Eigenvector with smallest eigenvalue (why not zero?)
- Minimises $\|\mathbf{Ah}\|$
- Is this meaningful?

Do we still have a problem?

- For each point we get constraints of the form:

$$\begin{bmatrix} 0 & 0 & 0 & -u_i & -v_i & -1 & u_i v_i' & v_i v_i' & v_i' \\ u_i & v_i & 1 & 0 & 0 & 0 & -u_i u_i' & -v_i u_i' & -u_i' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

  - us and vs are pixel values – on the order of 100 or 1000
  - So errors in h_3 and h_6 get multiplied by 1
  - Errors in h_7 and h_8 get multiplied by 10,000 to 1,000,000

- This makes the solution very unstable

The Solution

- Transform the points before computing H
  - Make it so that on average their size is about 1
  - Subtract average of the (u_i, v_i)s – so points are around the origin
  - Then scale so that the average distance from the origin is $\sqrt{2}$
  - Why $\sqrt{2}$? Because if $u = v = 1$, then $\left|(u, v)\right| = \sqrt{1^2 + 1^2} = \sqrt{2}$
- Now all the values are on the order of 1 (ish)
  - Changing all of the h_is has an equal effect
  - This changes the homography so need to undo the transform to get the final result

Algorithm: Normalized DLT

Input: $n \geq 4$ correspondences $\mathbf{u}_i \leftrightarrow \mathbf{u}_i'$

Output: Homography, H, such that $\mathbf{u}_i' = \mathrm{H}\mathbf{u}_i$

1. **Normalisation:** Find T and T' to centre $\tilde{\mathbf{u}}_i = \mathrm{T}\mathbf{u}_i$ and $\tilde{\mathbf{u}}_i' = \mathrm{T}'\mathbf{u}_i'$ on the origin with average length $\sqrt{2}$

2. **Direct Linear Transform:**
   1. Form the matrix A from $\tilde{\mathbf{u}}_i$ and $\tilde{\mathbf{u}}_i'$
   2. Compute the SVD of A, and find the smallest eigenvector, $\tilde{\mathbf{h}}$
   3. Reshape $\tilde{\mathbf{h}}$ to give the 3×3 homography matrix $\tilde{\mathrm{H}}$

3. **Denormalisation:** Final solution is $\mathrm{H} = \mathrm{T}'^{-1}\tilde{\mathrm{H}}\mathrm{T}$

RANSAC

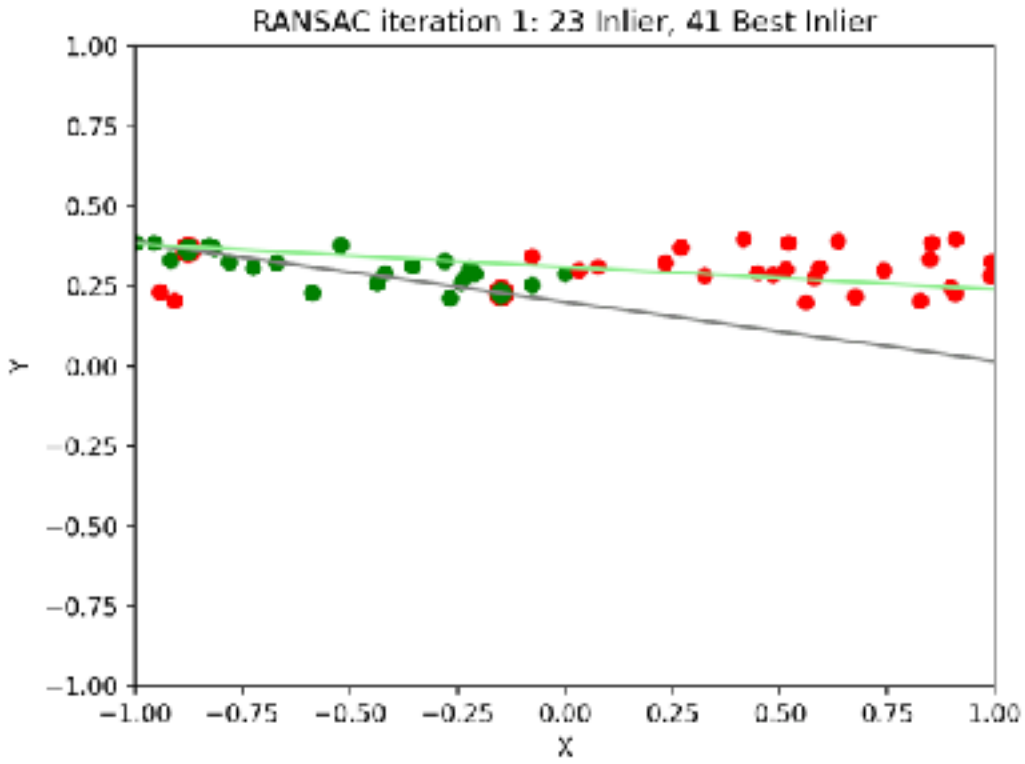
Errors vs Outliers

- Errors (uncertainty)
  - All measurements of the world are uncertain
- For the stitching process, we measure feature locations
  - How accurate are these?
  - 1 pixel? More? Less?
- Least squares solutions
  - DLT minimises $\left|\mathbf{Ah}\right|^2$

- Outliers (mistakes)
- Some measures are wrong
- E.g. recording 1.2 m as .12 m
- Some of these matches are just wrong
- Very bad for least squares

RANSAC for Lines

- Fitting a line to points
  - Some points are outliers
  - Least squares fit skewed away from the main trend
- RANSAC:
  - Randomly pick two points
  - Draw a line through them
  - Count points close to the line
  - Repeat until find a good one



RANSAC for Lines

- **function** RANSAC (n points, $\mathbf{p}_i$; numTrials, threshold)

```
consensus = {}
for trial ← 1 to numTrials do
    s1 = rand(1,n); s2 = rand(1,n); thisConsensus = {}
    L = fitLine(p_s1, p_s2)
    for i ← 1 to n do
        if (distance(L, p_i) < threshold) then
            thisConsensus = thisConsensus + p_i
if |thisConsensus| > |consensus| then
    consensus = thisConsensus
return fitLine(consensus)
```

RANSAC for Lines

```
const auto& p1 = points[idx1];
const auto& p2 = points[idx2];

// Fit a line model (y = mx + b).
double m = (p2.y - p1.y) / (p2.x - p1.x);
double b = p1.y - m * p2.x;

// Find and count the inliers.
int current_inlier_count = 0;
for (const auto& point : points) {
    double dist = std::abs(point.y - (m * point.x + b));
    if (dist < inlier_threshold) {
        current_inlier_count++;
    }
}

// If the current model is better, update the best one.
if (current_inlier_count > best_inlier_count) {
    best_inlier_count = current_inlier_count;
    best_model = {m, b};
}
```

RANSAC for General Model Fitting

3 steps

1. Sample subset of data points
2. Compute a model based on sample data points
3. Test all data points and compute a score for all data points

Repeat

RANSAC for General Model Fitting

- We need:
  - A set of n points or items, p_i
  - To be able to fit a model, M, to $k \ll n$ points
  - To find the distance, $d(M, p_i)$
- RANSAC has parameters:
  - A threshold for acceptance
  - A number of trials (see later)
- RANSAC then proceeds as for the line fitting, except:
  - We select k items at random
  - We fit our model, M, to them

RANSAC for Homographies

- Homography estimation:
  - Items are point matches
  - We have $k = 4$
  - Model is a homography

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \equiv \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & -u_i & -v_i & -1 & u_i v_i' & v_i v_i' & v_i' \\ u_i & v_i & 1 & 0 & 0 & 0 & -u_i u_i' & -v_i u_i' & -u_i' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

RANSAC for Homographies

- Threshold for acceptance
  - Can often be interpreted in terms of the task
  - E.g. for how well does $\mathbf{H}$ align the corresponding points?
  - Set too low -> smaller consensus sets
  - Set too high -> outliers give less accurate models
- Number of trials
  - Resource limits
    - E.g. tracking in 25 fps video, do as many trials as you can in 0.04s
  - Can also be interpreted in a probabilistic manner
    - Replace number of trials with probability of success

How Many Trials?

- What's the chance we get an outlier in the final solution?
  - Some proportion, ϕ, of items are inliers
  - Probability of picking k good points is ϕ^k
  - Probability of one trial going wrong is $(1-\phi^k)$
  - Probability of t trials going wrong is $\left(1-\phi^k\right)^t$
  - Set this to some small probability, p, and solve for t

$$t = \frac{\log(p)}{\log\left(1-\phi^k\right)}$$

e.g. 0.7 for 70% inliers in feature matching

e.g. 0.7^4 =0.2401 for 4 points)

1-0.2401 = 0.7599

$$p = \left(1-\phi^k\right)^t$$

$$\log(p) = \log\left(1-\phi^k\right)^t$$

$$\log(p) = t\log\left(1-\phi^k\right)$$

How Many Trials?

- Homography from SIFT:
  - $k = 4$, choose $p = 0.001$
  - With filtering, $\phi \approx 0.7$

$$t = \frac{\log(0.001)}{\log\left(1 - 0.7^4\right)} \approx 25$$

  - Without filtering, $\phi \approx 0.3$

$$t = \frac{\log(0.001)}{\log\left(1 - 0.3^4\right)} \approx 850$$

- Can estimate ϕ as we go
  - For n points, initialise $\phi = \frac{k}{n}$
  - If we find a consensus set, C

$$\phi = \max\left(\phi, \frac{\|C\|}{n}\right)$$

- As we make more trials
  - ϕ increases, t decreases
  - Eventually $t <$ trials done

Updated Method

Input: Two images, I_1 and I_2; **Output:** a homography, H

1. **Compute and describe** features in each image
2. **Find correspondences** between images, and filter them
3. **RANSAC** estimation via the **normalised DLT:**
   1. Repeatedly select four correspondences, and estimate an H*
   2. See how many other points agree with each H*, and keep the best
4. **Least squares** estimation: using all inliers, estimate H using the **normalised DLT**

Optimisation and Cost Functions

Algorithm: Normalized DLT

Input: $n \geq 4$ correspondences $\mathbf{u}_i \leftrightarrow \mathbf{u}_i'$

Output: Homography, H, such that $\mathbf{u}_i' = \mathrm{H}\mathbf{u}_i$

1. **Normalisation:** Find T and T' to centre $\tilde{\mathbf{u}}_i = \mathrm{T}\mathbf{u}_i$ and $\tilde{\mathbf{u}}_i' = \mathrm{T}'\mathbf{u}_i'$ on the origin with average length $\sqrt{2}$

2. **Direct Linear Transform:**
   1. Form the matrix A from $\tilde{\mathbf{u}}_i$ and $\tilde{\mathbf{u}}_i'$
   2. Compute the SVD of A, and find the smallest eigenvector, $\tilde{\mathbf{h}}$
   3. Reshape $\tilde{\mathbf{h}}$ to give the 3×3 homography matrix $\tilde{\mathrm{H}}$

3. **Denormalisation:** Final solution is $\mathrm{H} = \mathrm{T}'^{-1}\tilde{\mathrm{H}}\mathrm{T}$

RANSAC + DLT for Homography

Input: Two images, I_1 and I_2; **Output:** a homography, H

1. **Compute and describe** features in each image
2. **Find correspondences** between images, and filter them
3. **RANSAC** estimation via the **normalised DLT:**
   1. Repeatedly select four correspondences, and estimate an H*
   2. See how many other points agree with each H*, and keep the best
4. **Least squares** estimation: using all inliers, estimate H using the **normalised DLT**

Direct Linear Transform

- Solve the linear system of equations, $\mathbf{Ah} = \mathbf{0}$

$$\begin{bmatrix} 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1v'_1 & v_1v'_1 & v'_1 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1u'_1 & -v_1u'_1 & -u'_1 \\ 0 & 0 & 0 & -u_2 & -v_2 & -1 & u_2v'_2 & v_2v'_2 & v'_2 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2u'_2 & -v_2u'_2 & -u'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_nv'_n & v_nv'_n & v'_n \\ u_n & v_n & 1 & 0 & 0 & 0 & -u_nu'_n & -v_nu'_n & -u'_n \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

Cost Functions and Error Terms

- With more than 4 points we are minimising $\|\mathbf{Ah}\|^2$
  - This is the algebraic distance
  - It is computationally ‘easy’ to minimise
  - It lacks a clear geometric or statistical interpretation
  - We want to minimise a more meaningful error measure

- Geometric distance
  - If we assume only errors in second image, minimise the transfer error
    $$\sum_i \|\mathbf{u}_i - \mathbf{H}\mathbf{u}_i'\|^2$$
  - A symmetric form is
    $$\sum_i \|\mathbf{u}_i - \mathbf{H}\mathbf{u}_i'\|^2 + \|\mathbf{u}_i' - \mathbf{H}^{-1}\mathbf{u}_i\|^2$$

Cost Functions and Error Terms

- With error in both images
  - $\mathbf{u}_i$ and $\mathbf{u}_i'$ are noisy estimates of true values, $\hat{\mathbf{u}}_i$ and $\hat{\mathbf{u}}_i^{'}$
  - These are perfectly matched by the true homography $\hat{\mathbf{u}}_i \equiv \hat{\mathbf{H}}\hat{\mathbf{u}}_i^{'}$
  - $\mathbf{H}$, $\mathbf{u}_i$, and $\mathbf{u}_i'$ from the DLT are approximations of the true values, $\hat{\mathbf{H}}$, $\hat{\mathbf{u}}_i$, and $\hat{\mathbf{u}}_i^{'}$

- Reprojection error:

$$\sum_i \left\|\mathbf{u}_i - \hat{\mathbf{u}}_i\right\|^2 + \left\|\mathbf{u}_i' - \hat{\mathbf{u}}_i^{'}\right\|^2$$

  - Subject to $\hat{\mathbf{u}}_i = \hat{\mathbf{H}}\hat{\mathbf{u}}_i^{'}, \quad \forall i$
- This is hard to minimise
  - Nonlinear least-squares
  - Iterative from an initial estimate of $\hat{\mathbf{H}}$, $\hat{\mathbf{u}}_i$, and $\hat{\mathbf{u}}_i^{'}$

Blending

Blending in image stitching

- Handling seams and exposure differences in panoramas
- Naïve stitching = visible seams

$$I(x) = \begin{cases} I_A(x), & x \in A \setminus B \\ I_B(x), & x \in B \end{cases}$$

- Issues:
  - Different exposures
  - Lighting changes
  - Parallax
- Goal: seamless panorama

Naive alpha blending

- Simple averaging across overlap

$$I(\mathbf{x}) = \frac{1}{2} I_A(\mathbf{x}) + \frac{1}{2} I_B(\mathbf{x})$$

- Pros: simple, fast

- Cons: avoids a hard seam, but if the images are misaligned or have different exposures, it causes ghosting (double edges, blurred details)

Feathering (Alpha Blending)

- Weighted averaging across overlap for two images:

$$I(\mathbf{x}) = w_A(\mathbf{x})\, I_A(\mathbf{x}) + w_B(\mathbf{x})\, I_B(\mathbf{x})$$

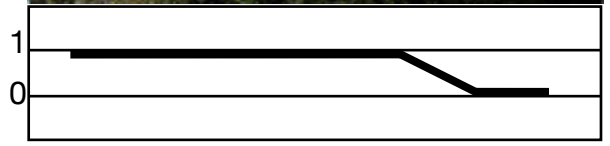
- Distance-based weighting
- Pros: smooth transitions, less visible seams
- Cons: still some ghosting if strong parallax or moving objects
- Multiple images:

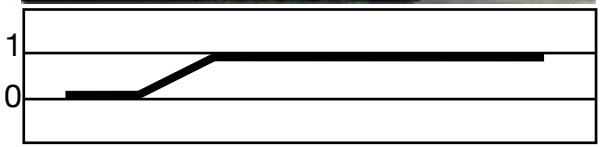
$$I(\mathbf{x}) = \sum_{i=1}^{N} w_i(\mathbf{x})\, I_i(\mathbf{x}) \sum_{i=1}^{N} w_i(\mathbf{x}) = 1$$

$$w_i(\mathbf{x}) = \frac{d_i(\mathbf{x})}{\sum_{j=1}^{N} d_j(\mathbf{x})}, \qquad i = 1,2,\ldots,N$$

Left Weighted

Right Weighted





Multi-band Blending

- Blend low frequencies smoothly, high frequencies sharply
- Low frequencies:
  - Smooth background, illumination, color differences
  - Obtained by Gaussian blur:
    $L_A = G_\sigma * I_A, \quad L_B = G_\sigma * I_B$
- High frequencies:
  - Edges, details, textures
  - Computed as: $H_A = I_A - L_A, \quad H_B = I_B - L_B$
- Better at handling illumination/exposure differences
- Often used in panorama software

International Journal of Computer Vision 74(1), 59–73, 2007
© 2007 Springer Science + Business Media, LLC. Manufactured in the United States.
DOI: 10.1007/s11263-006-0002-3

Automatic Panoramic Image Stitching using Invariant Features

MATTHEW BROWN* AND DAVID G. LOWE
Department of Computer Science, University of British Columbia, Vancouver, Canada
mbrown@cs.ubc.ca
lowe@cs.ubc.ca

Received July 29, 2005; Accepted August 3, 2006

First online version published in December 2006

Abstract. This paper concerns the problem of fully automated panoramic image stitching. Though the 2D problem (single axis of rotation) is well studied, 3D or multi-row stitching is more difficult. Previous approaches have used human input or restrictions on the image sequence in order to establish matching images. In this work, we formulate stitching as a multi-image matching problem, and use invariant local features to find matches between all of the images. Because of this our method is insensitive to the ordering, orientation, scale and illumination of the input images. It is also insensitive to noise images that are not part of a panorama, and can recognise multiple panoramas in an unordered image dataset. In addition to providing more detail, this paper extends our previous work in the area (Brown and Lowe, 2003) by introducing gain compensation and automatic straightening steps.

Keywords: multi-image matching, stitching, recognition

1. Introduction

Panoramic image stitching has an extensive research literature (Szeliski, 2004; Milgram, 1975; Brown and Lowe, 2003) and several commercial applications (Chen, 1995; Realviz, http://www.realviz.com; http://www.microsoft.com/products/imaging). The basic geometry of the problem is well understood, and consists of estimating a 3 × 3 camera matrix or homography for each image (Hartley and Zisserman, 2004; Szeliski and Shum, 1997). This estimation process needs an initialisation, which is typically provided by user input to approximately align the images, or a fixed image ordering. For example, the PhotoStitch software bundled with Canon digital cameras requires a horizontal or vertical sweep, or a square matrix of images. REALVIZ Stitcher version 4 (http://www.realviz.com) has a user interface to roughly position the images with a mouse, before automatic registration proceeds. Our work is novel in that we require no such initialisation to be provided.

In the research literature methods for automatic image alignment and stitching fall broadly into two categories—direct (Szeliski and Kang, 1995; Irani and Anandan, 1999; Sawhney and Kumar, 1999; Shum and Szeliski, 2000) and feature based (Zoghlami et al., 1997; Capel and Zisserman, 1998; McLauchlan and Jaenicke, 2002). Direct methods have the advantage that they use all of the available image data and hence can provide very accurate registration, but they require a close initialisation. Feature based registration does not require initialisation, but traditional feature matching methods (e.g., correlation of image patches around Harris corners (Harris, 1992; Shi and Tomasi, 1994)) lack the invariance properties needed to enable reliable matching of arbitrary panoramic image sequences.

In this paper we describe an invariant feature based approach to fully automatic panoramic image stitching. This has several advantages over previous approaches. Firstly, our use of invariant features enables reliable matching of panoramic image sequences despite rotation, zoom and illumination change in the input images. Secondly, by viewing image stitching as a multi-image matching problem, we can automatically discover the matching relationships between the images, and recognise panoramas in unordered datasets. Thirdly, we generate high-quality results using multi-band blending

*Corresponding author.

Brown, M., Lowe, D.G. Automatic Panoramic Image Stitching using Invariant Features. Int J Comput Vision 74, 59–73 (2007). https://doi.org/10.1007/s11263-006-0002-3

Multi-band Blending

- Blend low frequencies smoothly, high frequencies sharply
- Final Image computed as:

$$I(\mathbf{x}) = L(\mathbf{x}) + H(\mathbf{x})$$

- Where:

$$L(\mathbf{x}) = \alpha(\mathbf{x})L_A(\mathbf{x}) + \Big(1 - \alpha(\mathbf{x})\Big)L_B(\mathbf{x})$$

$$H(\mathbf{x}) = M(\mathbf{x})H_A(\mathbf{x}) + \Big(1 - M(\mathbf{x})\Big)H_B(\mathbf{x})$$

α(x) is the smooth (feathering) mask used for low-frequency blending.

M(x) s the binary mask (hard cut) used for high-frequency blending.

(a) Linear blending (b) Multi-band blending

Brown, M., Lowe, D.G. Automatic Panoramic Image Stitching using Invariant Features. Int J Comput Vision 74, 59–73 (2007). https://doi.org/10.1007/s11263-006-0002-3

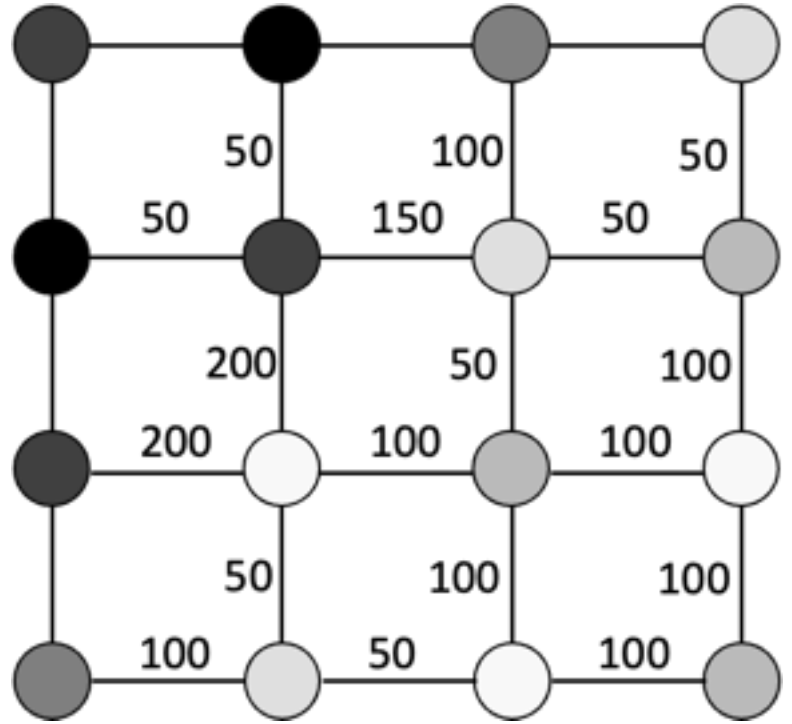
Gradient-Domain (Poisson) Blending

- Match gradients across boundaries
  - Goal: preserve the local variations of the source image while aligning with the target image's boundary region
  - Instead of directly copying pixel values, method ensures the rate of intensity change (gradient) is consistent
- Solve Poisson equation for smooth transition:
  - Find an image f inside a region Ω
  - $\nabla^2 f = \mathrm{div}(\mathbf{g})$ in Ω,
  - Solution involves solving a large sparse system of equations (a discretized Poisson equation)
- Produces seamless color consistency
- Computationally expensive

Optimal Seam Finding

- Place seam in least noticeable area
- Energy = gradient magnitude (avoid cutting across strong edges)
- Methods: Graph cuts, dynamic programming
- Often combined with multi-band blending

50	0	100	200
0	50	200	150
50	250	150	250
100	200	250	150



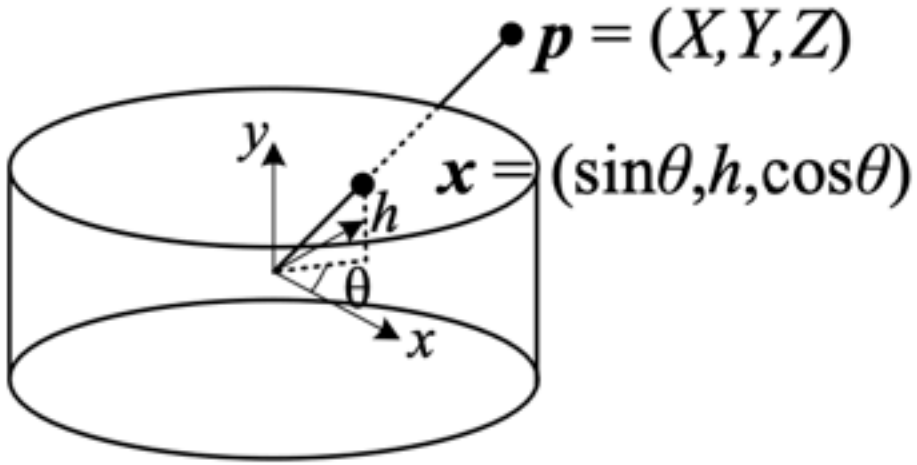
Practical Considerations

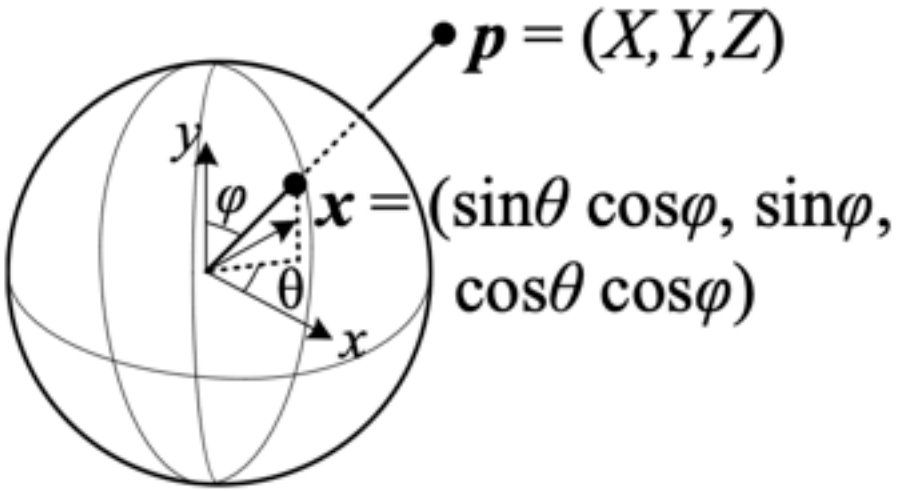
- Trade-offs: quality vs. speed
- Common in panorama tools:
- Seam finding + multi-band blending
- Handling moving objects → need additional strategies (e.g., exposure compensation, local warping)

Projections

Projections

- Panorama stitching maps multiple images to a common coordinate system.
- Common projections:
  - Cylindrical – wraps images on a cylinder.
  - Spherical – wraps images on a sphere.
  - Equirectangular – maps a spherical surface to a rectangle
- Choice depends on field of view and application (e.g., wide horizontal vs 360°)





http://szeliski.org/Book/

Demo: https://threejs.org/examples/webgl_panorama_equirectangular.html

Next time:
3D Geometry, Cameras

The end!