Applied Machine Vision
Estimation

Matthew Dailey

Information and Communication Technologies
Asian Institute of Technology

Readings

Readings for these lecture notes:

- Hartley, R., and Zisserman, A. *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2004, Chapter 4.
- Tomasi, C. *Mathematical Modeling of Continuous Systems*, online lecture notes from Duke University, 2004.

If you are unfamiliar with the mathematics used in these lecture notes, study Carlo Tomasi's beautiful lecture notes. You'll find the notes on the course Web site under "Readings."

Outline

Introduction
Estimation Problems

In vision, we are frequently confronted with estimation problems in which parameters of some function must be estimated from measurements.

Some examples of important estimation problems:

- 2D homography: Given a set of points $\mathbf{x}_i$ in $\mathbb{P}^2$ and corresponding points $\mathbf{x}'_i$ in $\mathbb{P}^2$, find a homography taking each $\mathbf{x}_i$ to $\mathbf{x}'_i$.
- 3D to 2D camera projection: Given a set of points $\mathbf{X}_i$ in 3D space and corresponding points $\mathbf{x}_i$ in an image, find the 3D to 2D projective mapping taking each $\mathbf{X}_i$ to $\mathbf{x}_i$.
- Fundamental matrix computation: Given a set of points $\mathbf{x}_i$ in one image and a set of corresponding points $\mathbf{x}'_i$ in another image, find the fundamental matrix $\mathtt{F}$ relating the two images.
- Trifocal tensor computation: Given a set of point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i \leftrightarrow \mathbf{x}''_i$ across three images, compute the trifocal tensor $\mathcal{T}_i^{jk}$ relating points or lines in three views.

Introduction
Homography estimation

First we'll consider homography estimation.

We have a set of points $\mathbf{x}_i$ and corresponding points $\mathbf{x}'_i$. We want to compute $\mathtt{H}$ such that $\forall i, \mathtt{H}\mathbf{x}_i = \mathbf{x}'_i$.

How many points do we need?

- We already saw that each point correspondence gives us 2 constraints (equations), one for the x component and one for the y component.
- Since $\mathtt{H}$ has 8 degrees of freedom we need at least 4 correspondences.

Introduction
Cost functions

We know that 4 correspondences yields an exact solution.

Due to measurement error and correspondence error we should get more than 4 correspondences then find the homography $\mathtt{H}$ minimizing some cost function.

Gold Standard algorithm (Hartley and Zisserman, 2004)

An estimation algorithm minimizing the cost function that is the best possible cost function under certain assumptions.

All algorithms should be evaluated with respect to the Gold Standard.

Outline

Direct Linear Transform (DLT)

Another view of the homography estimation problem

Now we'll look at another way to derive the exact linear estimate of $\mathtt{H}$ from 4 points.

For corresponding points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ we want $\mathbf{x}'_i = k_i \mathtt{H} \mathbf{x}_i$ for some nonzero scaling factor k.

Thus we can say that $\mathbf{x}'_i$ and $\mathtt{H}\mathbf{x}_i$ must be collinear.

Recall that the cross product of two collinear vectors is the 0 vector.

This means we can write this constraint in the form $\mathbf{x}'_i \times \mathtt{H}\mathbf{x}_i = \mathbf{0}$.

Direct Linear Transform (DLT)

Deriving the linear system

Let's use $\mathbf{h}^j$ to denote the j-th row of $\mathtt{H}$ written as a vector. Then we have

$$\mathtt{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1T}\mathbf{x}_i \\ \mathbf{h}^{2T}\mathbf{x}_i \\ \mathbf{h}^{3T}\mathbf{x}_i \end{pmatrix}.$$

Now if $\mathbf{x}'_i = (x'_i, y'_i, w'_i)^T$, the cross product can be written

$$\mathbf{x}'_i \times \mathtt{H}\mathbf{x}_i = \begin{pmatrix} y'_i\mathbf{h}^{3T}\mathbf{x}_i - w'_i\mathbf{h}^{2T}\mathbf{x}_i \\ w'_i\mathbf{h}^{1T}\mathbf{x}_i - x'_i\mathbf{h}^{3T}\mathbf{x}_i \\ x'_i\mathbf{h}^{2T}\mathbf{x}_i - y'_i\mathbf{h}^{1T}\mathbf{x}_i \end{pmatrix}.$$

Direct Linear Transform (DLT)

Deriving the linear system

Since we want the cross product to be the zero vector, we can write the linear system

$$\begin{bmatrix} \mathbf{0}^T & -w'_i \mathbf{x}_i^T & y'_i \mathbf{x}_i^T \\ w'_i \mathbf{x}_i^T & \mathbf{0}^T & -x'_i \mathbf{x}_i^T \\ -y'_i \mathbf{x}_i^T & x'_i \mathbf{x}_i^T & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}.$$

We have three linear equations in 9 unknowns. The third equation is actually a linear combination of the first two.[1] So we can drop it (see next slide)...

[1] To convince yourself of this, use the factor $-x'_i / w'_i$ for the first equation and $-y'_i / w'_i$ for the second equation.

Direct Linear Transform (DLT)

Deriving the linear system

Dropping the redundant third equation in the parameters of $\mathtt{H}$, we have:

$$\mathtt{A}_i\mathbf{h} = \begin{bmatrix} \mathbf{0}^T & -w_i'\mathbf{x}_i^T & y_i'\mathbf{x}_i^T \\ w_i'\mathbf{x}_i^T & \mathbf{0}^T & -x_i'\mathbf{x}_i^T \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}. \qquad (1)$$

However, if $w_i' = 0$ we have an ideal point, and the first two equations become linearly dependent. In this case we should use the third equation and the first or second equation, or just use all three equations all the time.

Direct Linear Transform (DLT)

Deriving the linear system

With 8 equations in 9 unknowns, $\mathtt{A}$ is 8×9 and has rank 8. It has a one-dimensional null space and we obtain $\mathbf{h} = \mathcal{N}(\mathtt{A})$.

If we have more than 4 correspondences the system $\mathtt{A}\mathbf{h} = \mathbf{0}$ will be over-determined:

- With perfect measurement, the rank of $\mathtt{A}$ would still be 8 and it would still have a one-dimensional null space.
- But measurement noise means no exact solution.
- So we try to find the vector $\mathbf{h}$ minimizing some cost function.

Since we want $\mathtt{A}\mathbf{h}$ to be as close as possible to $\mathbf{0}$, a sensible cost function is the norm of $\mathtt{A}\mathbf{h}$, i.e., $\|\mathtt{A}\mathbf{h}\|$.

However we must avoid the trivial solution $\mathbf{h} = \mathbf{0}$, so we impose a constraint $\|\mathbf{h}\| = 1$. This is OK since $\mathtt{H}$ is homogeneous.

Direct Linear Transform (DLT)

Solving the linear system

So now we have the minimization problem

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\operatorname{argmin}} \|\mathtt{A}\mathbf{h}\|, \text{subject to } \|\mathbf{h}\| = 1.$$

whose solution is to let $\mathbf{h}$ be the unit eigenvector of $\mathtt{A}^T\mathtt{A}$ with the smallest eigenvalue, or the unit singular vector of $\mathtt{A}$ corresponding to the smallest singular value of $\mathtt{A}$.

This is important to remember: to solve an overconstrained homogeneous linear system $\mathtt{A}\mathbf{x} = \mathbf{0}$ by minimizing $\|\mathtt{A}\mathbf{h}\|$ subject to $\|\mathbf{h}\| = 1$, we perform SVD on $\mathtt{A}$ (see next section) or compute the eigenvector corresponding to the smallest eigenvalue of $\mathtt{A}^T\mathtt{A}$.

Direct Linear Transform (DLT)

The basic DLT for $\mathtt{H}$

DLT: Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the 2D homography matrix $\mathtt{H}$ such that $\mathbf{x}'_i = \mathtt{H}\mathbf{x}_i$.

DLT: Algorithm

(i) For each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, compute the matrix $\mathtt{A}_i$ as in equation (1). When $w'_i = 0$, use different rows.

(ii) Assemble the n 2×9 matrices $\mathtt{A}_i$ into a single $2n \times 9$ matrix $\mathtt{A}$.

(iii) Obtain the SVD $\mathtt{A} = \mathtt{UDV}^T$. The unit singular vector (column of $\mathtt{V}$) corresponding to the smallest singular value (diagonal element of $\mathtt{D}$) is the desired $\mathbf{h}$.

(iv) Rearrange $\mathbf{h}$ to obtain $\mathtt{H}$.

Direct Linear Transform (DLT)

Similar approaches

Other methods: we can select one element of $\mathbf{h}$ to be equal to 1 (or any other arbitrary value) to obtain an inhomogeneous linear system which can be solved by the usual least squares methods (see text).

We can also compute $\mathtt{H}$ in the same way using line correspondences or conic correspondences. The derivations are quite similar.

Direct Linear Transform (DLT)

Example code

For example code for the DLT in Octave, see `dlt_demo.m` on the course Web site.

Outline

Singular value decomposition (SVD)
Definition

The SVD is an incredibly useful factorization, particularly for the kinds of estimation problems that come up in computer vision.

Singular Value Decomposition

Given an $m \times n$ matrix $\mathtt{A}$, the singular value decomposition of $\mathtt{A}$ is

$$\mathtt{A} = \mathtt{UDV}^T$$

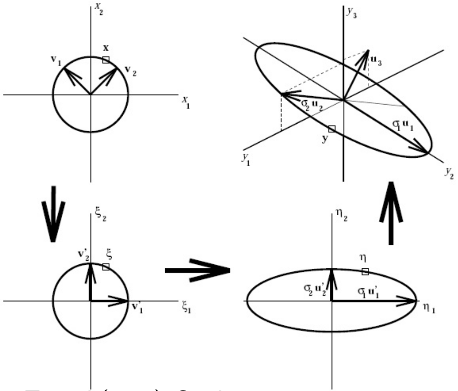
where the columns of $\mathtt{U} \in \mathbb{R}^{m \times m}$ and $\mathtt{V} \in \mathbb{R}^{n \times n}$ are orthogonal unit vectors and $\mathtt{D} \in \mathbb{R}^{m \times n}$ is a diagonal matrix whose elements σ_i, with $\sigma_1 \geq \cdots \geq \sigma_p \geq 0$, $(p = \min(m,n))$, are called the singular values of $\mathtt{A}$.

Singular value decomposition (SVD)

Geometric interpretation

Writing $\mathtt{A} = \mathtt{UDV}^T$ models the transformation $\mathbf{y} = \mathtt{A}\mathbf{x}$ as a rotation, a "stretch" of the unit hypershpere into a hyperellipse, and a rotation of the hyperellipse. Example:

$$\mathtt{A} = \frac{1}{\sqrt{2}} \begin{bmatrix} \sqrt{3} & \sqrt{3} \\ -3 & 3 \\ 1 & 1 \end{bmatrix}$$



Example from Tomasi (2004), Section 3.

Singular value decomposition (SVD)

Properties of the SVD

The SVD has many useful properties:

- If $m = n$ and $\sigma_i \neq 0, \forall i$, then $\mathtt{A}$ is invertible. The ratio $C = \sigma_1 / \sigma_n$ is called the condition number of $\mathtt{A}$ and tells us how close $\mathtt{A}$ is to singularity. When $1/C$ is close to numerical precision, we say $\mathtt{A}$ is ill-conditioned and should be considered singular.
- The number of nonzero σ_i is the rank of $\mathtt{A}$. Numerically, we must specify a tolerance, e.g. $\epsilon = 10^{-6}$, and say the number of singular values greater than ϵ is the rank of $\mathtt{A}$.
- We can get the inverse or pseudoinverse of $\mathtt{A}$ using the SVD: $\mathtt{A}^{-1} = \mathtt{V}\mathtt{D}^{-1}\mathtt{U}^T$ or $\mathtt{A}^{+} = \mathtt{V}\mathtt{D}_0^{-1}\mathtt{U}^T$, where $\mathtt{D}_0^{-1}$ is $\mathtt{D}^{-1}$ for the nonzero singular values and 0 otherwise.

Singular value decomposition (SVD)

Properties of the SVD

More properties:

- The columns of $\mathtt{U}$ corresponding to the nonzero σ_i span the range of $\mathtt{A}$. The columns of $\mathtt{V}$ corresponding to the zero singular values span the null space of $\mathtt{A}$.
- The squares of the nonzero singular values of $\mathtt{A}$ are the nonzero eigenvalues of $\mathtt{A}^T\mathtt{A}$ and $\mathtt{A}\mathtt{A}^T$. The columns of $\mathtt{U}$ are the eigenvectors of $\mathtt{A}\mathtt{A}^T$ and the columns of $\mathtt{V}$ are the eigenvectors of $\mathtt{A}^T\mathtt{A}$. Additionally, we can write $\mathtt{A}\mathbf{u}_k = \sigma_k \mathbf{v}_k$ and $\mathtt{A}^T\mathbf{v}_k = \sigma_k \mathbf{u}_k$ where $\mathbf{v}_k$ is the kth column of $\mathtt{V}$ and $\mathbf{u}_k$ is the kth column of $\mathtt{U}$.
- The singular values of $\mathtt{A}$ are related to the Frobenius norm of $\mathtt{A}$:

$$\|\mathtt{A}\|_F^2 = \sum_{i,j} a_{i,j}^2 = \sum_i \sigma_i^2 .$$

Singular value decomposition (SVD)

Applications

Here are some of the SVD's many uses:

- In inhomogeneous linear least squares problems ($\mathtt{A}\mathbf{x} = \mathbf{b}$), we use the SVD to obtain the pseudoinverse of A and let $\mathbf{x} = \mathtt{A}^{+}\mathbf{b}$.
- In homogeneous least squares problems, when we want to minimize $\|\mathtt{A}\mathbf{x}\|$ subject to $\|\mathbf{x}\| = 1$, we obtain the SVD and let $\mathbf{x}$ be the last column of $\mathtt{V}$ (since it is also the least eigenvector of $\mathtt{A}^{T}\mathtt{A}$).
- In some cases we can use the SVD to enforce constraints on an estimated matrix. For example, if we obtain an estimate $\mathtt{R}$ of a rotation matrix that is not quite orthogonal, we can compute the orthogonal matrix $\hat{\mathtt{R}} = \mathtt{U}\mathtt{I}\mathtt{V}^{T}$ that is closest to $\mathtt{R}$ measured by the Frobenius norm. We can use a similar approach for rank constraints.

Outline

Cost functions
Algebraic distance

DLT minimizes $\|\mathtt{A}\mathbf{h}\|$. The vector $\epsilon = \mathtt{A}\mathbf{h}$ is called the residual vector. Each of the components of ϵ comes from one of the individual correspondences generating a row of $\mathtt{A}$.

The part of the vector ϵ_i contributed by one correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ is called the algebraic error for correspondence i and homography $\mathtt{H}$. The norm of ϵ_i is called the algebraic distance between $\mathbf{x}'_i$ and $\mathtt{H}\mathbf{x}_i$.

The algebraic distance is a convenient cost function because it leads to a straightforward linear solution, but it is not geometrically or statistically meaningful!

We will find that normalization is crucial to obtaining good results from algorithms minimizing algebraic error.

We will also look at algorithms that use DLT and similar linear algebraic error-minimizing routines to obtain an initial solution, then minimize a statistical or geometrical cost function from there.

Cost functions
Geometric error

Here we'll use $\mathbf{x}$ to represent a measured image point, $\hat{\mathbf{x}}$ to denote an estimated point, and $\bar{\mathbf{x}}$ to represent the true value of a point.

As a starting point, imagine we have perfect measurements in the first image and error in the second image.

Let $d(\mathbf{x}, \mathbf{y})$ be the Euclidean distance between the inhomogeneous representations of points $\mathbf{x}$ and $\mathbf{y}$.

We call the transfer error for the set of correspondences

$$\sum_i d(\mathbf{x}'_i, \mathrm{H}\bar{\mathbf{x}}_i)^2.$$

We can estimate the homography $\hat{\mathrm{H}}$ minimizing the transfer error.

Cost functions

Geometric error

In most practical situations, we don't actually know the true position $\bar{\mathbf{x}}_i$ of the point $\mathbf{x}_i$. Then it makes sense to measure the symmetric transfer error, i.e., the transfer error in both directions:

$$\sum_i d(\mathbf{x}_i, \mathtt{H}^{-1}\mathbf{x}'_i)^2 + d(\mathbf{x}'_i, \mathtt{H}\mathbf{x}_i)^2.$$

We can estimate the homography $\hat{\mathtt{H}}$ minimizing the symmetric transfer error.

Cost functions

Reprojection error

Another approach is to come up with not only an estimate $\hat{\mathrm{H}}$, but also estimates $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}'_i$ of the true points $\bar{\mathbf{x}}_i$ and $\bar{\mathbf{x}}'_i$, ensuring that $\hat{\mathrm{H}}\hat{\mathbf{x}}_i = \hat{\mathbf{x}}'_i$.

In this case we want to minimize the reprojection error

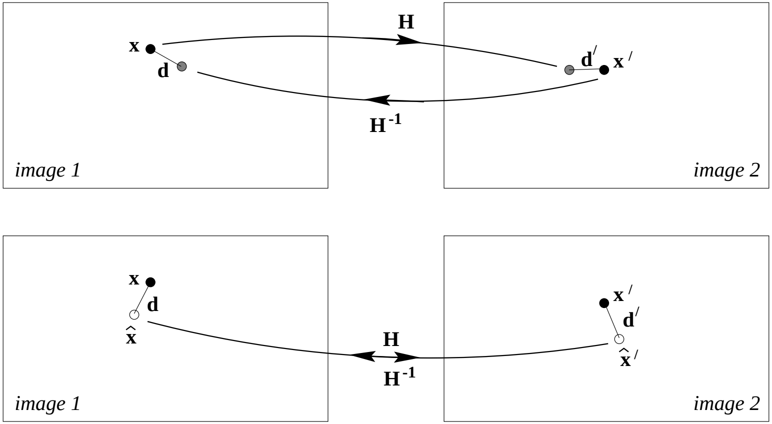
$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2, \text{ subject to } \hat{\mathbf{x}}'_i = \hat{\mathrm{H}}\hat{\mathbf{x}}_i, \forall i.$$

Reprojection error will be the natural cost function when we are estimating 3D world points $\mathbf{X}_i$ projecting to $\mathbf{x}_i$ and $\mathbf{x}'_i$.

Cost functions

Comparison of transfer error and reprojection error

Here is a comparison of symmetric transfer error and reprojection error:



Hartley and Zisserman (2004), Fig. 4.2

Cost functions

Comparison of algebraic and geometric distance

The algebraic and geometric methods turn out to be equivalent whenever $\hat{w}'_i = w'_i = 1, \forall i$.

This is always true in the case that $\mathtt{H}$ is an affinity, and the DLT specializes to affinities without any problem (just set $h_7 = h_8 = 0$ in Equation (1)).

Cost functions
Other cost functions

There are other cost functions that attempt to model the simplicity of algebraic error while approximating geometric error as closely as possible.

One is the Sampson error. See text for details.

Cost functions

Maximum likelihood estimation

If we assume spherical Gaussian measurement errors in one image, the maximum likelihood estimate of $\mathtt{H}$ is the one minimizing the transfer error

$$\sum_i d(\mathbf{x}'_i, \mathtt{H}\bar{\mathbf{x}}_i)^2.$$

If we assume spherical Gaussian measurement errors in both images, the maximum likelihood estimate of $\mathtt{H}$ turns out to be the one minimizing the reprojection error

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2.$$

See text for the derivations. Notice that maximum likelihood with a Gaussian noise model often leads to least-squares methods.

Cost functions
Maximum likelihood estimation

That geometric error cost functions arise from maximum likelihood means they have good theoretical justification: they are statistically optimal under certain assumptions.

Is the assumption of Gaussian measurement noise reasonable?

It is reasonable if care is taken to eliminate outliers prior to performing the estimation.

Outline

Normalization

Problem with the DLT

What happens with the DLT when we replace $\mathbf{x}_i$ by $\mathtt{T}\mathbf{x}_i$ and replace $\mathbf{x}'_i$ by $\mathtt{T}'\mathbf{x}'_i$ for arbitrary homographies $\mathtt{T}$ and $\mathtt{T}'$?

- We would prefer the DLT to give us a transformed homography $\tilde{\mathtt{H}} = \mathtt{T}'\mathtt{H}\mathtt{T}^{-1}$, where $\mathtt{H}$ is the homography we would obtain from the DLT using the untransformed points.
- If this were the case, we would get the same estimated homography regardless of the image coordinate system, origin, etc.
- However, it doesn't turn out that way! The DLT is not transformation invariant.

The DLT's lack of transformation invariance is a big problem but can be minimized through data normalization.

Geometric error minimization, on the other hand, is invariant to similarity transformations.

Normalization

Fixing the problem with the DLT

Hartley and Zisserman find it is possible to pre-normalize $\mathbf{x}_i$ and $\mathbf{x}'_i$ with isotropic scaling to obtain reasonable solutions:

Isotropic scaling

- Transform the coordinates in each image so their centroid is at the origin.
- Then scale the coordinates so that the average distance from the origin along each dimension is 1. In 2D, this means the average magnitude of $(x_i, y_i)^T$ becomes $\sqrt{2}$.

With isotropic scaling, the DLT becomes invariant to similarity transformations.

Data normalization is an *essential* step in the DLT algorithm. It must *not* be considered optional (Hartley and Zisserman, 2004, p. 108).

Normalization

The normalized DLT for $\mathtt{H}$

Normalized DLT: Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the 2D homography matrix $\mathtt{H}$ such that $\mathbf{x}'_i = \mathtt{H}\mathbf{x}_i$.

Normalized DLT: Algorithm

(i) Compute a similarity transform $\mathtt{T}$ consisting of a translation and a scale that takes $\mathbf{x}_i$ to $\tilde{\mathbf{x}}_i$ such that the centroid is $(0,0)^T$ and the average distance from the origin is $\sqrt{2}$.

(ii) Do the same for $\mathbf{x}'_i$, estimating a similarity $\mathtt{T}'$ taking $\mathbf{x}'_i$ to $\tilde{\mathbf{x}}'_i$.

(iii) Apply the basic DLT to obtain $\tilde{\mathtt{H}}$ from $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}'_i$.

(iv) Return $\mathtt{H} = \mathtt{T}'^{-1}\tilde{\mathtt{H}}\mathtt{T}$.

Outline

Iterative minimization
Motivation

We saw that the DLT is a simple algorithm that minimizes the algebraic error.

However, we would prefer to minimize the geometric error, not the algebraic error.

For some problems, geometric error can be minimized analytically, but more often minimizing geometric error requires iterative methods such as Gauss-Newton.

The advantage of iterative minimization methods are their power. But they have many disadvantages compared to linear minimization techniques like the DLT:

- They are slower.
- They generally need an initial estimate.
- The might not always converge.
- Finding the right stopping criterion can be difficult.

Iterative minimization

Formulating the problem

Here are Hartley and Zisserman's steps for implementing an iterative minimization algorithm:

- Decide on a cost function such as reprojection error.
- Find a suitable parameterization of the entity to be estimated. Over-parameterization is usually OK and even beneficial (see text).
- Write a function specification expressing the cost as a function of the parameters.
- Use a linear method such as the DLT for initialization of the parameters.
- Starting from the initial solution, perform iteration to minimize the cost function.

We will consider as an example the problem of minimizing reprojection error for the homography estimation problem.

Iterative minimization
Function specification

A large class of cost functions can be formulated in terms of Mahalanobis distance between a measurement vector $\mathbf{X} \in \mathbb{R}^{\mathbf{N}}$ and points on a model submanifold $S \subset \mathbb{R}^N$.

We formulate the problem in terms of a parameter vector $\mathbf{P} \in \mathbb{R}^M$ and write a function $\mathbf{f} : \mathbb{R}^M \mapsto \mathbb{R}^N$ mapping $\mathbf{P}$ to an element of the measurement space.

Then the cost function is the squared Mahalanobis distance

$$\|\mathbf{X} - \mathbf{f}(\mathbf{P})\|_\Sigma^2 = (\mathbf{X} - \mathbf{f}(\mathbf{P}))^T \Sigma^{-1} (\mathbf{X} - \mathbf{f}(\mathbf{P})).$$

- If $\mathbf{f}(\mathbf{P})$ is linear, we obtain a solution using a generalized pseudoinverse.
- If $\mathbf{f}(\mathbf{P})$ is nonlinear (it usually is in vision problems), we have a nonlinear least squares problem that must be solved iteratively. The most common algorithm is Levenberg-Marquardt.

Iterative minimization

Nonlinear least squares

Here is the basic iterative scheme:

- Pick some initial solution $\mathbf{P}_0$ (hopefully close to the actual solution).
- Let $k = 0$.
- While $\mathbf{P}_k$ is not a minimum of $\|\mathbf{X} - \mathbf{f}(\mathbf{P})\|_\Sigma^2$, do:
    - Compute a step $\delta\mathbf{P}$.
    - Let $\mathbf{P}_{k+1} = \mathbf{P}_k + \delta\mathbf{P}$.
    - Let $k = k + 1$.

To find the best step $\delta\mathbf{P}$, we try to jump to the new point $\mathbf{P}_k + \delta\mathbf{P}$ that would be optimal under some simplified assumptions about $\mathbf{f}$. One technique is to linearize $\mathbf{f}$ around $\mathbf{P}_k$.

Iterative minimization
Linearizing f

For vector-valued function $\mathbf{f}(\mathbf{P}) = (f_1(\mathbf{P}), \ldots, f_N(\mathbf{P}))^T$, in the region of $\mathbf{P}$, we can approximate $\mathbf{f}$ by the first-order Taylor expansion

$$\mathbf{f}(\mathbf{P}) = \mathbf{f}\left(\mathbf{P}_0 + (\mathbf{P} - \mathbf{P}_0)\right) = \mathbf{f}(\mathbf{P}_0 + \delta\mathbf{P}) \approx \mathbf{f}(\mathbf{P}_0) + \mathrm{J}_\mathbf{f}(\mathbf{P}_0)\delta\mathbf{P},$$

where $\mathrm{J}_\mathbf{f}(\mathbf{P}_0)$ is the Jacobian of $\mathbf{f}$ evaluated at $\mathbf{P}_0$:

$$\mathrm{J}_\mathbf{f}(\mathbf{P}_0) = \begin{pmatrix} \nabla f_1^T(\mathbf{P}_0) \\ \vdots \\ \nabla f_N^T(\mathbf{P}_0) \end{pmatrix}$$

and $\nabla f_i(\mathbf{P}_0)$ is the gradient of f_i evaluated at $\mathbf{P}_0$, i.e.,

$$\nabla f_i(\mathbf{P}_0) = \left(\frac{\partial f_i}{\partial P_1}(\mathbf{P}_0), \ldots, \frac{\partial f_i}{\partial P_M}(\mathbf{P}_0) \right)^T.$$

Iterative minimization
Minimizing the linearized $\mathbf{f}$

We've approximated $\mathbf{f}(\mathbf{P})$ by the linear function $\mathbf{f}(\mathbf{P}_0) + \mathrm{J}_{\mathbf{f}}(\mathbf{P}_0)(\mathbf{P} - \mathbf{P}_0)$.

Let's assume for the moment that we use the L_2 norm $(\Sigma = \mathtt{I})$ and that $\mathbf{X} = \mathbf{0}$. Then minimizing $\|\mathbf{X} - \mathbf{f}(\mathbf{P})\|^2_\Sigma$ just means minimizing

$$E(\mathbf{P}) = \sum_{i=1}^{N} f_i^2(\mathbf{P}).$$

If $\mathbf{P}$ is a minimum of $E(\mathbf{P})$, we know that

$$\mathbf{F}(\mathbf{P}) = \frac{1}{2}\nabla E(\mathbf{P}) = \mathbf{0}.$$

Substituting the definition for $E(\mathbf{P})$ and writing as a matrix equation, we can obtain

$$\mathbf{F}(\mathbf{P}) = \mathrm{J}_{\mathbf{f}}^T(\mathbf{P})\mathbf{f}(\mathbf{P}) = \mathbf{0}.$$

Now we'll use Newton's method to find the zero of $\mathbf{F}$.

Iterative minimization

Minimizing the linearized $\mathbf{f}$

Newton's method for a vector valued fuction $\mathbf{F}$ is to solve

$$\mathrm{J}_{\mathbf{F}}(\mathbf{P}_k)\delta\mathbf{P} = -\mathbf{F}(\mathbf{P}_k)$$

for $\delta\mathbf{P}$ then jump to $\mathbf{P}_{k+1} = \mathbf{P}_k + \delta\mathbf{P}$.

In our case it turns out that

$$\mathrm{J}_{\mathbf{F}}(\mathbf{P}) = \mathrm{J}_{\mathbf{f}}^T(\mathbf{P})\mathrm{J}_{\mathbf{f}}(\mathbf{P}) + \sum_{i=1}^{N} f_i(\mathbf{P})\mathtt{H}_{f_i}(\mathbf{P}).$$

where $\mathtt{H}_{f_i}(\mathbf{P})$ is the Hessian (matrix of second derivatives of f_i) evaluated at $\mathbf{P}$.

Iterative minimization
Minimizing the linearized $\mathbf{f}$

Finally, to apply Newton's method we solve the linear system

$$\left[\mathrm{J}_{\mathbf{f}}^{T}(\mathbf{P})\mathrm{J}_{\mathbf{f}}(\mathbf{P}) + \sum_{i=1}^{N} f_i(\mathbf{P})\mathrm{H}_{f_i}(\mathbf{P}) \right] \delta\mathbf{P} = -\mathrm{J}_{\mathbf{f}}^{T}(\mathbf{P})\mathbf{f}(\mathbf{P}).$$

Crazy! Let's do an example in one dimension to get the idea.

Iterative minimization
Minimizing the linearized **f** (example)

Example in one dimension.

Let $f(p) = p^2 - 4p + 5$. We know the minimum is at $p = 2$.

The "Jacobian" in one dimension is actually just the derivative:

$$\mathrm{J}_f(p) = \left[f'(p) \right] = \left[2p - 4 \right].$$

The error function is

$$E(p) = f^2(p),$$

so the function we want to find the zero of using Newton's method is

$$F(p) = \frac{1}{2} \nabla E(p) = \mathrm{J}_f^T(p) f(p) = (2p-4)(p^2 - 4p + 5).$$

Iterative minimization

Minimizing the linearized **f** (example)

Example in one dimension, continued.

To find the zero of $F(p)$, we need

$$\begin{aligned} \mathrm{J}_F(p) &= \mathrm{J}_f^T(p)\mathrm{J}_f(p) + \sum_{i=0}^{N} f_i(p)\mathrm{H}_{f_i}(p) \\ &= (2p-4)^2 + (p^2 - 4p + 5)\cdot 2 \\ &= 6p^2 - 24p + 26. \end{aligned}$$

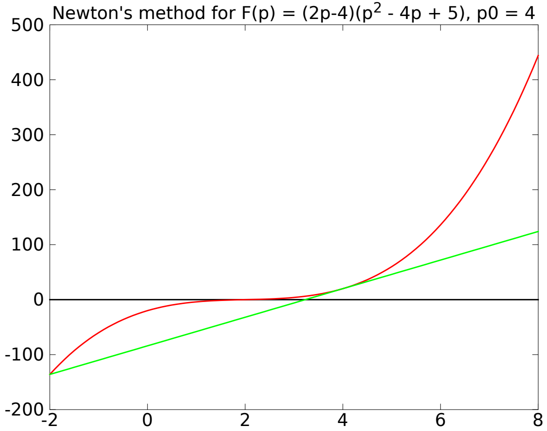
then we solve the linear system

$$\mathrm{J}_F(p)\delta p = -F(p).$$

Iterative minimization

Minimizing the linearized **f**

Suppose our initial guess for the parameter is $p = 4$.



Verify that $\delta p = -10/13 \approx -0.76923$.

Iterative minimization
Minimizing the linearized $\mathbf{f}$

Problem with Newton's method: the Hessians

$$\begin{bmatrix} \frac{\partial^2 f_i}{\partial p_1^2} & \frac{\partial^2 f_i}{\partial p_1 \partial p_2} & \cdots \\ \vdots & \vdots & \vdots \\ \frac{\partial^2 f_i}{\partial p_M \partial p_1} & \frac{\partial^2 f_i}{\partial p_M \partial p_2} & \cdots \end{bmatrix}$$

are tedious and expensive to calculate, especially if M, the dimensionality of the parameter vector, is large.

Iterative minimization

Minimizing the linearized **f**

Because of the difficulty of calculating Hessians we typically use quick-and-dirty approximations rather than explicitly calculate them.

The Gauss-Newton algorithm drops the second-order terms entirely.

The Levenberg-Marquardt algorithm approximates the second order terms by a scaled identity matrix:

$$\left[\mathrm{J}_{\mathbf{f}}^T(\mathbf{P})\mathrm{J}_{\mathbf{f}}(\mathbf{P}) + \mu \mathrm{I} \right] \delta\mathbf{P} = -\mathrm{J}_{\mathbf{f}}^T(\mathbf{P})\mathbf{f}(\mathbf{P}).$$

where the parameter μ is adapted during optimization.

Lucky for us, Levenberg-Marquardt is implemented in many packages: Matlab `lsqnonlin()`, Octave `leasqr()`, and *Numerical Recipes in C* `mrqmin()`.

See the text, Appendix 6, for information on adapting Levenberg-Marquardt to very large problems.

Iterative minimization

Iterative minimization applied to homography estimation

In the case of homography estimation, we have a set of 2D coordinates of corresponding points $\mathbf{x}_i$ and $\mathbf{x}'_i$, so the dimensionality of the measurement space is $N = 4n$ and the measurement is a vector $\mathbf{X} \in \mathbb{R}^N$.

Suppose our parameterization was to choose a set of points $\hat{\mathbf{x}}_i$ in the first image, then choose a homography $\mathtt{H}$.

- The corresponding points $\hat{\mathbf{x}}'_i = \hat{\mathtt{H}}\hat{\mathbf{x}}_i$ would then be fixed.
- The parameter vector $\mathbf{P} \in \mathbb{R}^M$, then, would contain the $2n$ parameters of $\hat{\mathbf{x}}_i$ and the 9 parameters of $\hat{\mathtt{H}}$, so $M = 2n + 9$.
- The resulting model (the set of measurements in $\mathbb{R}^N$ that can be be generated with our parameterization) is a $2n + 8$ dimensional submanifold $S \subset \mathbb{R}^N$.

[Analogy: think about a circle as a 1D submanifold of $\mathbb{R}^2$.]

Iterative minimization

Iterative minimization applied to homography estimation

Given the model in the previous page, it is straightforward to write the mapping

$$\mathbf{f} : (\mathbf{h}, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n) \mapsto (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}'_1, \ldots, \hat{\mathbf{x}}_n, \hat{\mathbf{x}}'_n)$$

where $\hat{\mathbf{x}}'_i = \mathtt{H}\hat{\mathbf{x}}_i$.

The reprojection error cost function becomes $\|\mathbf{X} - \mathbf{f}(\mathbf{P})\|^2$, which is just the Mahalanobis cost function with $\Sigma = \mathtt{I}$.

This means Levenberg-Marquardt applies. The resulting algorithm is Hartley and Zisserman's Gold Standard MLE algorithm for estimating $\mathtt{H}$.

Outline

Gold Standard algorithm
The idea

The Gold Standard algorithm for $\mathtt{H}$ tries to find

$$(\hat{\mathtt{H}}, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n) = \underset{\hat{\mathtt{H}}, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n}{\operatorname{argmax}} P(\mathbf{x}_1, \mathbf{x}'_1, \ldots, \mathbf{x}_n, \mathbf{x}'_n \mid \mathtt{H}, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$$

We know that maximizing the likelihood in the equation, assuming Gaussian measurement error, means minimizing the reprojection error

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathtt{H}}\hat{\mathbf{x}}_i)^2$$

This is a nonlinear least squares problem, which Levenberg-Marquardt can solve, but we need an initial solution, which we can obtain using the DLT.

Gold Standard algorithm

The algorithm

Gold Standard for H: Objective

Given $n > 4$ image point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the maximum likelihood estimate $\hat{\mathrm{H}}$.

Gold Standard

(i) Compute an initial estimate of $\hat{\mathrm{H}}$ using the normalized DLT.

(ii) Compute an initial estimate of the subsidiary variables $\{\hat{\mathbf{x}}_i\}$ using $\{\mathbf{x}_i\}$ (see text for a better way).

(iii) Minimize the cost

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathrm{H}}\hat{\mathbf{x}}_i)^2$$

over $\hat{\mathrm{H}}, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n$, using Levenberg-Marquardt over $2n + 9$ variables: $2n$ for the points $\{\hat{\mathbf{x}}_i\}$ and 9 for the homography matrix $\hat{\mathrm{H}}$.

Gold Standard algorithm

Example code

For example code for the Gold Standard algorithm in Octave, see `gs_demo.m` on the course Web site.

Note that if you're using Matlab you can use the Matlab port of the Octave `leasqr()` function or use the Matlab Optimization Toolbox function `lsqnonlin()`. But be careful as the two functions work a bit differently.

Outline

Robust estimation
Introduction

The Gold Standard algorithm is optimal if the measurement error for the corresponding points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ is actually Gaussian.

In practice, though, the points and their correspondences are obtained through an automatic procedure which makes mistakes.

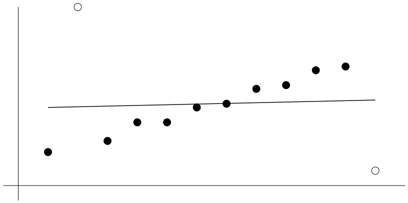
These mistakes, or outliers, will severely disrupt our estimates, so they should be removed.

We seek to obtain a set of inliers that will be used for estimation and a set of outliers that will be ignored.

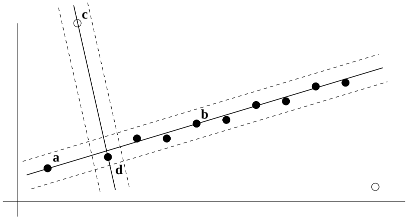
This task is called robust estimation because we want the estimation method to be robust to outliers following an unmodeled error distribution.

Robust estimation
RANSAC motivation

Example: fitting a line $x' = ax + b$ to a set of points.



Least squares fit is skewed by outliers.



RANSAC support for two candidate lines.

The RANSAC (Random Sample Consensus, Fischler and Bolles, 1981) algorithm is one of the many robust methods to solve this problem.

Robust estimation
RANSAC idea

The idea of RANSAC is that we only need two points to determine a line.

So to begin, we pick two points at random to define a line.

The support for this line is the number of points that lie within a distance threshold t of that line.

We repeat for a while, and the line with the most support is deemed best.

The points within the threshold are called inliers and they are said to make up the consensus set.

In the figure, we see that the line $\overline{\mathbf{ab}}$ has a support of 10 but the line $\overline{\mathbf{de}}$ has a support of only 2. We would select $\overline{\mathbf{ab}}$ as a better fit to the data than $\overline{\mathbf{de}}$.

Robust estimation
RANSAC algorithm

Hartley and Zisserman's (2004) adaptation of Fischler and Bolles' (1981) RANSAC:

RANSAC: Objective

Robust fit of a model to a data set S which contains outliers

RANSAC: Algorithm

(i) Randomly select a sample s from S and instantiate the model from s.

(ii) Find the consensus set (inliers) S_i within distance threshold t of the model.

(iii) If $|S_i| \geq T$ re-estimate the model using all of the points in S_i and terminate; otherwise repeat from (i).

(iv) After N trials, select the largest consensus set S_i and re-estimate the model using all of the points in S_i.

Robust estimation
RANSAC parameters

RANSAC has three free parameters:

- t: the distance threshold,
- T: the minimum number of inliers for early termination,
- N: the number of samples.

t can be determined empirically, or, if the error distribution is known to be Gaussian with standard deviation σ, a 95% or similar confidence interval can be calculated.

See Table 4.2 in the text for reasonable values of t for various vision problems.

Robust estimation

RANSAC parameters

N can be determined empirically, or if the proportion w of inliers is approximately known, we can choose N giving (e.g.) a 99% probability that on some iteration we will choose a sample containing inliers only.

Example: in homography estimation our sample size would be 4. If we assume a 50% outlier rate, we should select $N = 72$ samples to assure a 99% probability of sampling at least one set with 4 inliers.

See Table 4.3 in the text for some example values and how to calculate N in general.

T, the acceptable consensus set size, should be approximately the number of inliers thought to be in the data.

For example, in the case of homography estimation, if we have 100 correspondences and a 50% outlier rate, we should let $T = 50$.

Robust estimation

Adaptive version of RANSAC for unknown w

One variant, when the percentage of inliers w is unknown, is adpative RANSAC:

- Initialize $N = \infty$.
- While running RANSAC, decrease N whenever you obtain a sample with a bigger consensus set than previously seen.
- Terminate after N iterations.

A bigger consensus set means w is bigger than previously thought, so N need not be so large.

Robust estimation

Robust maximum likelihood estimation

Note that step (iv) of RANSAC was to re-estimate the model from all of the points in S_i. We should use maximum likelihood in this case.

Problem: the set of inliers could change after we compute the new maximum likelihood model.

We could just accept the estimate anyway.

Some approaches recompute the inliers after obtaining the maximum likelihood model then repeat maximum likelihood model estimation until the consensus set converges.

See text for detailed discussion and other alternative approaches.

Robust estimation
Using RANSAC to estimate H

Automatic H estimation: Objective

Given two images, compute the homography.

Automatic H estimation: Algorithm

(i) Compute a set of interest points in each image.

(ii) Compute putative correspondences between the point sets.

(iii) RANSAC robust estimation: Repeat for N samples, where N is determined adaptively as previously described:

  (a) Select a random sample of 4 correspondences and compute H.

  (b) Calculate the distance $d_\perp$ for each putative correspondence.

  (c) Find the number of inliers for which $d_\perp < t = \sqrt{5.99}\sigma$ pixels.

(iv) Re-estimate H using the inliers and the Gold Standard algorithm

(v) (Optional) use the new H to recompute the matching set of interest points and repeat from (iv) until convergence.

Robust estimation

Using RANSAC to estimate $\mathtt{H}$

There are two implementation details that need consideration: how to measure the distance and how to select the samples.

- For distance, the symmetric transfer error $d^2_{\text{transfer}} = d(\mathbf{x}, \mathtt{H}^{-1}\mathbf{x}')^2 + d(\mathbf{x}', \mathtt{H}\mathbf{x})^2$ is appropriate since it is easy to compute. Reprojection error is better but expensive.
- Widespread distribution of the samples is good, to ensure good interpolation in the rest of the image. The sampler can be biased to pick points in different regions of the image rather than uniformly.

Robust estimation
Example results

Example initial images (Hartley and Zisserman, 2004, Fig. 4.9):

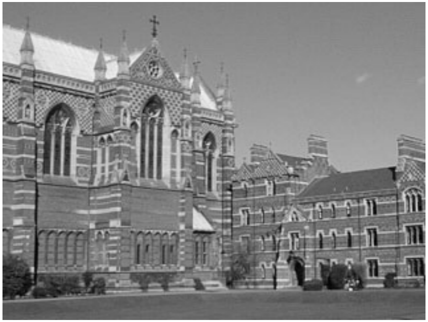


Image 1

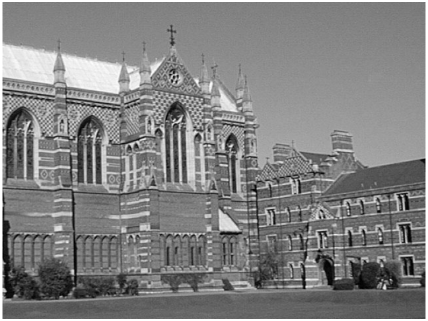
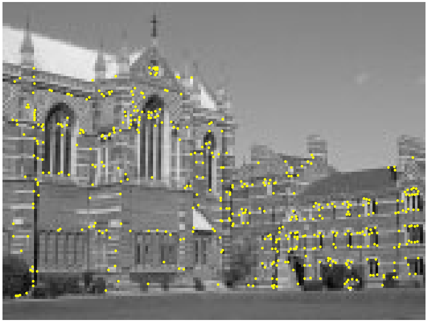


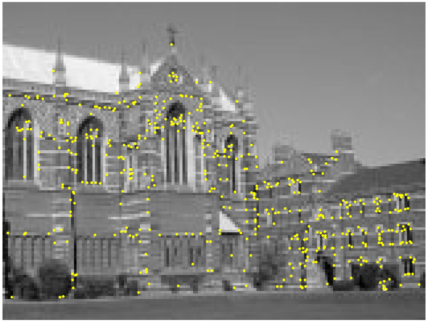
Image 2, related by rotation around camera center

Robust estimation

Example results

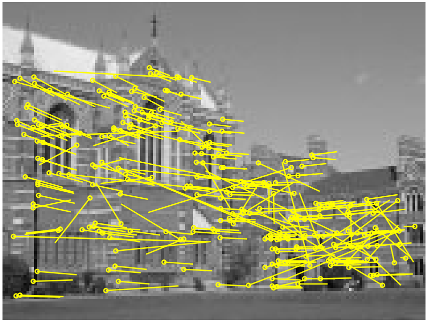
Detected corners, about 500 on each image.

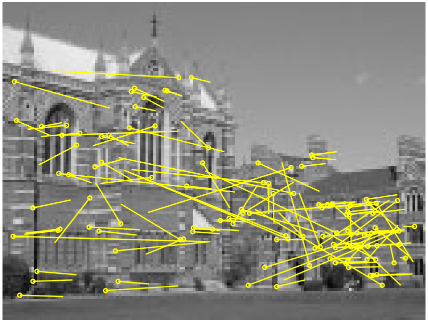




Robust estimation
Example results

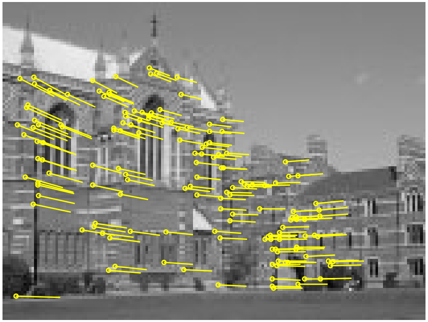
Initial set of 268 correspondences obtained by SSD of image patches around the corners:

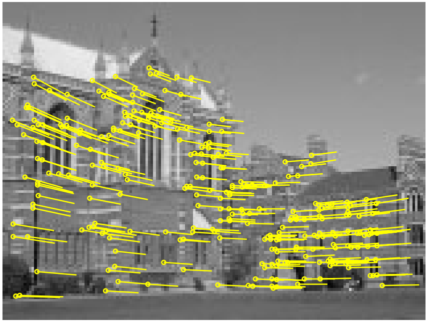

268 putative correspondences, Hartley and Zisserman (2004), Fig. 4.9(e)


117/268 outliers, Hartley and Zisserman (2004), Fig. 4.9(f)

Robust estimation
Example results

Final set of 262 correspondences after RANSAC, guided matching, and MLE.


151 inliers consistent with `H` found by RANSAC, Hartley and Zisserman (2004), Fig. 4.9(g).


Final set of 262 correspondences after guided matching and MLE beginning from the RANSAC solution, Hartley and Zisserman (2004), Fig. 4.9(h).

Robust estimation
Implementation

See the course Web site for a test using OpenCV, its built-in Shi-Tomasi feature detector and its built-in RANSAC `H` estimation function.

See the Torr toolbox at `http://cms.brookes.ac.uk/staff/PhilipTorr/Code/master_code.htm` for a Matlab implementation using the Harris corner detector.

When there is significant rotation and/or scale in the image plane, the SIFT (Scale Invariant Feature Transform) feature detector is much better than Harris or Shi-Tomasi.

There is a large literature on feature detectors enabling correspondence estimation over multiple views.

Transform invariance matching will be very important when we move to multiple views of a general scene and estimate the fundamental matrix `F` rather than a homography `H`.