Lecture 8: RANSAC and Minimal Solvers

1 The Outlier Problem

In previous lectures we have studied the algebraic equations that govern projective camera systems. Under the assumption that the data given to us in the form of point correspondences is correct, we have derived algorithms for approximately solving these in the presence of moderate noise. However, since correspondences are determined automatically this will not be true in practice. A typical situation is shown in Figure 1 where correspondences between two images have been determined using SIFT descriptors. In practice we have to expect that the data contains (at least) a small portion of incorrect matches. We refer to these as **outliers** and the rest as **inliers**.

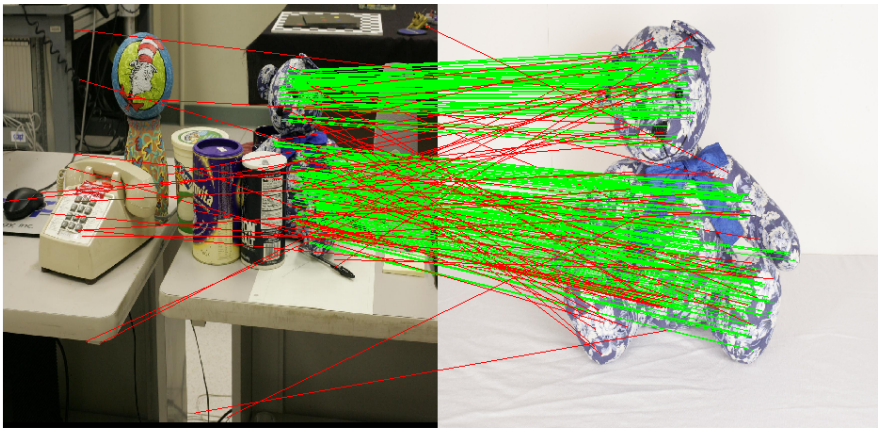

Figure 1: The Outlier Problem. When automatically detecting correspondences using descriptors such as SIFT there will always be a portion of incorrect matches. Green lines in the figure correspond to correct matches and red lines correspond to outliers.

As we saw in Lecture 7 the outliers typically do not fulfill the Gaussian noise assumption for the particular problem that we are trying to solve, and they can severely degrade the quality of the estimation. To address this issue we can use robust loss-functions. However, these can be sensitive to initialization and can often only handle a relatively small portion of outliers. Therefore we need a method for removing outliers as well as providing reliable starting solutions that can be locally refined.

2 RANSAC

Random sample consensus (RANSAC) is a method for removing outliers. The idea is simple; If the number of outliers is small, then if we pick a small subset of the measurements at random, we are likely to pick an outlier free set.

The outline of the algorithm is as follows:

1. Randomly select a small subset of measurements and solve the problem using only these.

2. Evaluate the error residuals for the rest of the measurements under the solution from 1. The **Consensus Set** for this solution is the set of measurements with error residuals less than some predefined threshold.

3. Repeat a number of times and select the solution that gives the largest consensus set.

The probability of randomly selecting a set of inliers depends on the size of the set and the proportion of inliers.

Exercise 1. Assume that we want to fit a line to a set of points. We randomly select 2 points and fit a line to these in each RANSAC iteration. Suppose that 10% of the points are outliers. How many iterations are required to find at least one set of only inliers with probability $p = 95\%$? You may assume that the set of points is large such that the portion of outliers do not change when removing a point. (Hint: First compute the probability of failure.)

Exercise 2. Same question as before but now we select 8 point correspondences to estimate a Fundamental matrix.

In practice it is a good idea to run more iterations than what is needed since, because of noise, not all inlier sets work equally well for estimating the solution. For example in the case of line estimation; if the two inlier points used to estimate the line are very close to each other then the line estimate can be very poor due to noise. Therefore the estimation may still generate a small consensus set.

3 Minimal Solvers and Solution of Polynomial Equation Systems

The more measurements we use in each RANSAC iteration the more iterations we need to run for finding good inlier sets. Therefore it is essential to use as few measurements as possible. Minimal solvers are a class of algebraic solvers that compute solutions from a minimal amount of data. In Lecture 6 we computed the Essential matrices from 8 or more point correspondences. However the essential matrix has only 5 degrees of freedom and a minimal solver for this problem therefore only uses 5 points. The 8-point algorithm is more general in that it works for any number of correspondences above 8 whereas the minimal solver only works for precisely 5. Still, in the context of a RANSAC algorithm the minimal solver is preferable.

Minimal solvers often need to find solutions to systems of non-linear equations. Next we will present a method for solving polynomial systems of equations. The idea is to transform the problem into an eigenvalue problem.

For simplicity we will first consider a system of two equations in two variables.

$$\begin{cases} x^2 - y - 3 &= 0 \\ xy - x &= 0. \end{cases} \tag{1}$$

By factoring out x from the second equation it can be seen that the system has three roots, namely $(0, -3)$, $(2, 1)$ and $(-2, 1)$.

In the following sections we will present a method for automatically finding these roots, that work under fairly general conditions. A polynomial p of degree n can represented using a monomial vector $m(x, y)$ containing all monomials of degree at most n and a coefficient vector c_p. For example, the polynomial $p(x, y) = 1 + 2x + 3y + 4x^2 + 5xy + 6y^2$ can be represented by $c_p^T m(x, y)$, where

$$c_p = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \quad \text{and} \quad m(x, y) = \begin{pmatrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{pmatrix}. \tag{2}$$

Using the monomials in $m(x, y)$ and a coefficient vector we can represent any second degree polynomial. The collection of monomials in $m(x, y)$ is called a **monomial basis**. The approach we will present is based on the observation that if we insert a root (x_0, y_0) in the monomial vector $m(x, y)$ then the resulting vector can be found by computing eigenvectors of a particular matrix.

3.1 The Action Matrix

We define T_x to be an operator that takes a polynomial $p(x, y)$ and multiplies it with x. If we apply T_x to the three monomials $1, x, y$ we get

$$\begin{aligned} 1 &\mapsto x && (3)\\ x &\mapsto x^2 && (4)\\ y &\mapsto xy. && (5) \end{aligned}$$

Now let us assume that (x_0, y_0) is a solution to (1). The result of applying T_x to the above monomials can then be simplified if we insert (x_0, y_0),

$$\begin{aligned} 1 &\mapsto x_0 && (6)\\ x_0 &\mapsto x_0^2 = y_0 + 3 && (7)\\ y_0 &\mapsto x_0 y_0 = x_0. && (8) \end{aligned}$$

Now suppose that a first order polynomial p is given by the coefficient vector $c_p = (c_p^1, c_p^2, c_p^3)$ and monomial vector $m(x, y) = (1, x, y)$. By $q(x, y)$ we denote the result of applying T_x to $p(x, y)$. Because of the reductions (6)-(8) we get

$$q(x_0, y_0) = c_p^1 x_0 + c_p^2(y_0 + 3) + c_p^3 x_0 = 3c_p^2 1 + (c_p^1 + c_p^3)x_0 + c_p^2 y_0. \quad (9)$$

We see that if (x_0, y_0) solves (1) then because of the reductions we can represent $q(x_0, y_0)$ using the vector $m(x_0, y_0)$ and a coefficient vector c_q. The coefficient vector can be found by identifying the monomials in (9),

$$\begin{pmatrix} c_q^1 \\ c_q^2 \\ c_q^3 \end{pmatrix} = \begin{pmatrix} 3c_p^2 \\ c_p^1 + c_p^3 \\ c_q^3 \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 3 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}}_{=M_x} \begin{pmatrix} c_p^1 \\ c_p^2 \\ c_p^3 \end{pmatrix}. \quad (10)$$

The matrix M_x is called the **action matrix** for the mapping T_x. Given the coefficients of $p(x, y)$ it computes the coefficients of $x_0 p(x_0, y_0)$ provided that (x_0, y_0) solves the system (1). Under certain conditions it is possible to compute the roots of the system from this matrix.

3.2 Finding the Roots.

Next we will show that the roots of the system (1) are eigenvalues to the action matrix M_x. For any polynomial p we have

$$x_0 p(x_0, y_0) = x_0 c_p^T m(x_0, y_0). \quad (11)$$

Furthermore,

$$x_0 p(x_0, y_0) = q(x_0, y_0) = c_q^T m(x_0, y_0) = (M_x c_p)^T m(x_0, y_0) = c_p^T M_x^T m(x_0, y_0). \quad (12)$$

Since this is true for any degree one polynomial (and therefore any coefficient matrix c_q of size 3×1) we must have that

$$x_0 m(x_0, y_0) = M_x^T m(x_0, y_0). \quad (13)$$

Therefore we can conclude that if (x_0, y_0) is a root of (1) then $m(x_0, y_0)$ is an eigenvector of M_x^T with eigenvalue x_0.

Exercise 3. Verify that $m(x_0, y_0)$ is an eigenvector of

$$M_x^T = \begin{pmatrix} 0 & 1 & 0 \\ 3 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad (14)$$

for all the three roots $(0, -3)$, $(2, 1)$ and $(-2, 1)$ of the system (1).

3.3 Algorithm

Based on the above derivations we now give an algorithm for finding the roots of a system of polynomials.

1. Select a basis of monomials.
2. Apply the mapping T_x to the monomial basis and reduce the result until the resulting expressions consists only of monomials from the basis.
3. Construct the action matrix M_x.
4. Compute eigenvalues and eigenvectors of M_x^T.
5. Extract solutions from the eigenvectors.

The theory from Section 3.2 says that the solutions will be among the eigenvectors. It does however not guarantee that there are no other eigenvectors. Therefore we might have to check the extracted solutions by inserting into the system of equations.

Furthermore, if the eigenvalues are not distinct there might be infinitely many eigenvectors to search. For example, if the x-coordinate of two of the roots are the same then M_x^T will have a double eigenvalue. If we cannot find the correct eigenvector then the eigenvalue will still give us some information, since it is not possible to have a root with x-coordinate that is not an eigenvalue M_x^T.

3.3.1 What degree of polynomials do we need?

In the above example we only considered monomials of degree 1 in (6)-(8). This worked since all the equations resulting from multiplication with x could be reduced to monomials of degree 1. Therefore we could represent both $p(x_0, y_0)$ and $x_0 p(x_0, y_0)$ with the monomial basis $1, x_0, y_0$. If this is possible or not depends on the system of equations. In general the basis has to be selected large enough so that all the reductions result in terms that are present in the basis.

The theory still holds even if we should not select the smallest possible monomial basis. For the system (1) we can consider all second degree polynomials. For example we can use the reductions:

$$
\begin{aligned}
1 &\mapsto x_0 && (15)\\
x_0 &\mapsto x_0^2 && (16)\\
y_0 &\mapsto x_0 y_0 && (17)\\
x_0^2 &\mapsto x_0^3 = x_0(y_0 + 3) = x_0 y_0 + 3x_0 && (18)\\
x_0 y_0 &\mapsto x_0^2 y_0 = x_0^2 && (19)\\
y_0^2 &\mapsto x_0 y_0^2 = x_0 y_0. && (20)
\end{aligned}
$$

Here we made reductions so that all the terms on the right hand side have degree 2 or less. Since all the monomials on the right hand side are also present on the left hand side we can construct an action matrix from these reductions. Note that some of these terms can be reduced further. However, the theory from Section 3.2 holds regardless if we do this or not. Further reduction would result in a different action matrix.

The resulting action matrix is in this case the 6×6 matrix

$$
M_x = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}. \tag{21}
$$

The transpose of this matrix has eigenvalues $\lambda = -2, 0, 2$ which agrees with our roots. The eigenvalue 0 does however have multiplicity four and therefore there is no unique eigenvector to this value. Hence we might not be able to find the solution $(0, -3)$ using the eigenvectors of M_x.

3.3.2 Using other mappings than T_x.

In section 3.1 we chose to construct the action matrix for multiplication with x. However, in principle any mapping $T_{q(x,y)}$ could be used. The choice of q does however affect the reductions (6)-(8). For example suppose that we use T_y instead. We get

$$\begin{aligned} 1 &\mapsto y_0 && (22)\\ x_0 &\mapsto x_0 y_0 && (23)\\ y_0 &\mapsto y_0^2 && (24)\\ x_0^2 &\mapsto x_0^2 y_0 = x_0^2 && (25)\\ x_0 y_0 &\mapsto x_0 y_0^2 = x_0 y_0 && (26)\\ y_0^2 &\mapsto y_0^3 = y_0^2(x_0^2 - 3) = x_0^2 - 3y_0^2. && (27) \end{aligned}$$

Here it does not seem possible to use only 1st order monomials since the degree of y_0^2 can not be reduced further using the equations in (1). (It is however possible to generate new equations from (1) that can be used for further reduction. However this is more complicated and we do not pursue this further.)

The resulting action matrix is in this case the 6×6 matrix

$$M_y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -3 \end{pmatrix}. \quad (28)$$

The transpose of this matrix has eigenvalues $-3, 1, 0$. Since there are two roots with y coordinate 1 the eigenvalue 1 will have at least multiplicity two. Therefore we can only extract the solution to $(0, -3)$ from this eigenspace.

A simple heuristic for generating action matrices with more distinct eigenvalues is to use a mapping $T_{q(x,y)}$ where $q(x, y)$ is a random combination of x and y. For example $0.5M_x^T + 0.5M_y^T$ (with M_x and M_y from (21) and (28) respectively) has five distinct eigenvalues.

Another trick that modifies the eigenspace is to drop some of the monomials. In (28) rows 1 and 2 are all zeros. This means that 1 and x_0 do not occur in any of the expressions after the reductions. Therefore we can remove these from the system. The new action matrix is

$$M_y = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -3 \end{pmatrix}. \quad (29)$$

Note however, that the monomial vector is now $m(x, y) = (y, x^2, xy, y^2)$ and therefore the eigenvector will not contain the value of x_0.

4 The 5-point solver

In this section we will construct a minimal solver for the problem of finding an Essential matrix. Given 5 point correspondences we will use the following equations:

$$\begin{aligned} \bar{\mathbf{x}}_i^T E \mathbf{x}_i = 0, \qquad & i = 1, ..., 5. && (30)\\ \det(E) = 0, && (31)\\ 2EE^T E - \text{trace}(EE^T)E = 0. && (32) \end{aligned}$$

The third constraint (32) actually consists of 9 polynomial equations since it is a matrix expression. Any matrix E that has a singular value decomposition of the form

$$E = U \underbrace{\begin{pmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{=S} V^T \tag{33}$$

will fulfill this constraint. This can be seen by inserting (33) into (32). Furthermore, it can be shown that any matrix that fulfills (32) must have a singular value decomposition as in (33).

To find the solutions of (30)-(32) we will first use (30) to reduce the number of variables. We construct an M matrix of size 5×9 from the 5 epipolar constraints, similar to what we did in Lecture 6. In contrast to the eight point algorithm, the M matrix is in itself not enough to determine all the 8 parameters of the essential matrix since it only represents 5 equations. Since the dimension of the nullspace of M is 4 we can find 4 linearly independent vectors v_i, $i = 1, ..., 4$ such that

$$M(\alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 + \alpha_4 v_4) = 0, \tag{34}$$

for any choice of coefficients $\alpha_1, ..., \alpha_4$. Reshaping these vectors into matrices we get

$$\bar{\mathbf{x}}_i^T(\alpha_1 E_1 + \alpha_2 E_2 + \alpha_3 E_3 + \alpha_4 E_4)\mathbf{x}_i = 0, \quad i = 1, ..., 5. \tag{35}$$

What remains is to find the coefficients $\alpha_1, ..., \alpha_4$ such that (31) and (32) are fulfilled. Note that since the scale of the essential matrix is arbitrary, we can assume that (for example) $\alpha_1 = 1$.

To determine the coefficients $\alpha_2, ..., \alpha_4$ we will use the method presented in the previous section. Equation (32) consists of 9 third order polynomials in $\alpha_2, \alpha_3, \alpha_4$. In addition we have (31) which consists of 1 third degree polynomial. To construct the action matrix we first need to compute the coefficients of these polynomials. These can easily be determined by rewriting (32)

$$2EE^TE - \text{trace}(EE^T)E = \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} \alpha_i\alpha_j\alpha_k \left(2E_iE_j^TE_k - \text{trace}(E_iE_j^T)E_k\right). \tag{36}$$

For each of the 9 constraints in (32) we can extract coefficients for the monomial $\alpha_i\alpha_j\alpha_k$ using this expression. Note that the monomials occur several times in this sum. For example, $(i, j, k) = (1, 1, 2)$ and $(i, j, k) = (1, 2, 1)$ both yield $\alpha_i\alpha_j\alpha_k = \alpha_1^2\alpha_2 = \alpha_2$. There are 64 terms in the sum but only 20 distinct monomials. The determinant constraint can be handled in the same way using the expression

$$\begin{aligned}\det(E) = \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4} \alpha_i\alpha_j\alpha_k\big(&e_{11}^ie_{22}^je_{33}^k + e_{12}^ie_{23}^je_{31}^k + e_{13}^ie_{21}^je_{32}^k \\ &- e_{11}^ie_{23}^je_{32}^k - e_{12}^ie_{21}^je_{33}^k - e_{13}^ie_{22}^je_{31}^k\big),\end{aligned} \tag{37}$$

where e_{ab}^i is element (a, b) in matrix E_i. In summary we construct a 10×20 matrix that contains the coefficients of all the 10 polynomials. The columns of this matrix correspond to the coefficients of the monomials:

$$\{\alpha_4^3, \alpha_3\alpha_4^2, \alpha_3^2\alpha_4, \alpha_3^3, \alpha_2\alpha_4^2, \alpha_2\alpha_3\alpha_4, \alpha_2\alpha_3^2, \alpha_2^2\alpha_4, \alpha_2^2\alpha_3, \alpha_2^3, \alpha_4^2, \alpha_3\alpha_4, \alpha_3^2, \alpha_2\alpha_4, \alpha_2\alpha_3, \alpha_2^2, \alpha_4, \alpha_3, \alpha_2, 1\}. \tag{38}$$

The 10 first monomials are of degree 3 and the rest are of lower degree. If we modify the coefficient matrix by performing Gaussian elimination we can determine reductions for each of these terms, see Figure 2. For example, after elimination, the first row of the matrix only contains α_4^3 from the third order monomials and can therefore be used to replace this term with lower order terms. To create an action matrix we use the 10 equations represented by the new coefficient matrix to compute reductions of all the third order monomials. In this case it does not matter if we use T_{α_2}, T_{α_3} or T_{α_4}, since reductions to second order or less for all third order monomials are availible in the modified coefficient matrix.

In summary the 5-point solver consists of the following steps:

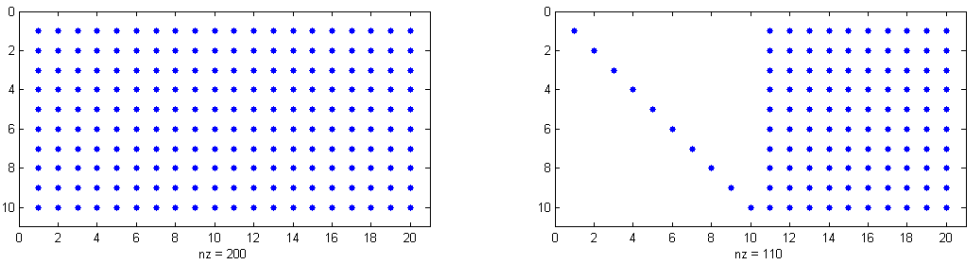

Figure 2: Shape of the 10×20 coefficient matrix before (left) and after elimination (right). A '*' means the element is non-zero.

1. Construct the 5×9 matrix M from the 5 point correspondences. (Note that the image points should be normalized as in Lecture 6.)
2. Compute the 4 vectors that span the nullspace of M and reshape them to the matrices E_1, E_2, E_3, E_4.
3. Using the expressions (36) and (37) compute coefficients for all monomials and construct the 10×20 coefficient matrix. (The monomial order does not have to be the same as (38), however the first 10 terms needs to be the 3rd order monomials.)
4. Perform Gaussian elimination on the coefficient matrix.
5. Construct the action matrix for either T_{α_2}, T_{α_3} or T_{α_4} using the reductions available in the modified coefficient matrix.

Figure 3 shows a comparison between the 5-point solver and the 8-point solver. For the pair of images depicted in the first row of the figure we apply a 1000 iterations of RANSAC. In the histograms on the second row we plot the size of the consensus set in each iteration. It can be seen that the 5-point solver generally finds consensus sets with a larger number of inliers.

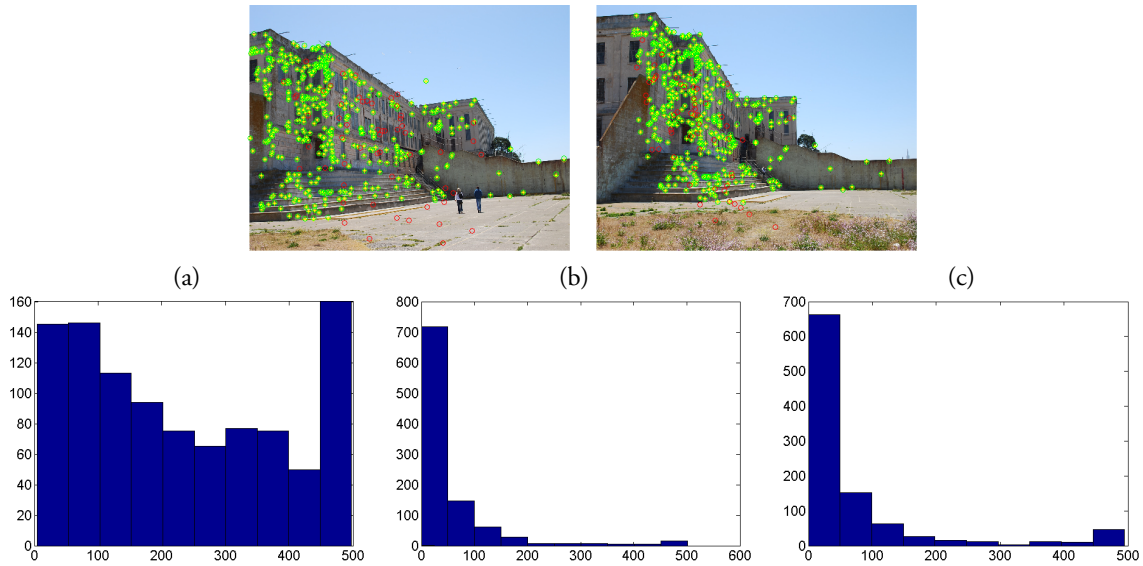

Figure 3: First row: The best solution obtained from the 5-point solver. Yellow '*' is an image point, green 'o' is the reprojection of an inlier and red 'o' is a reprojection of an outlier. Second row: Histogram over the size of the consensus set in each iteration of a 1000-iteration-RANSAC, using (a) - 5 points, (b) - 8 points and (c) - 10 points.