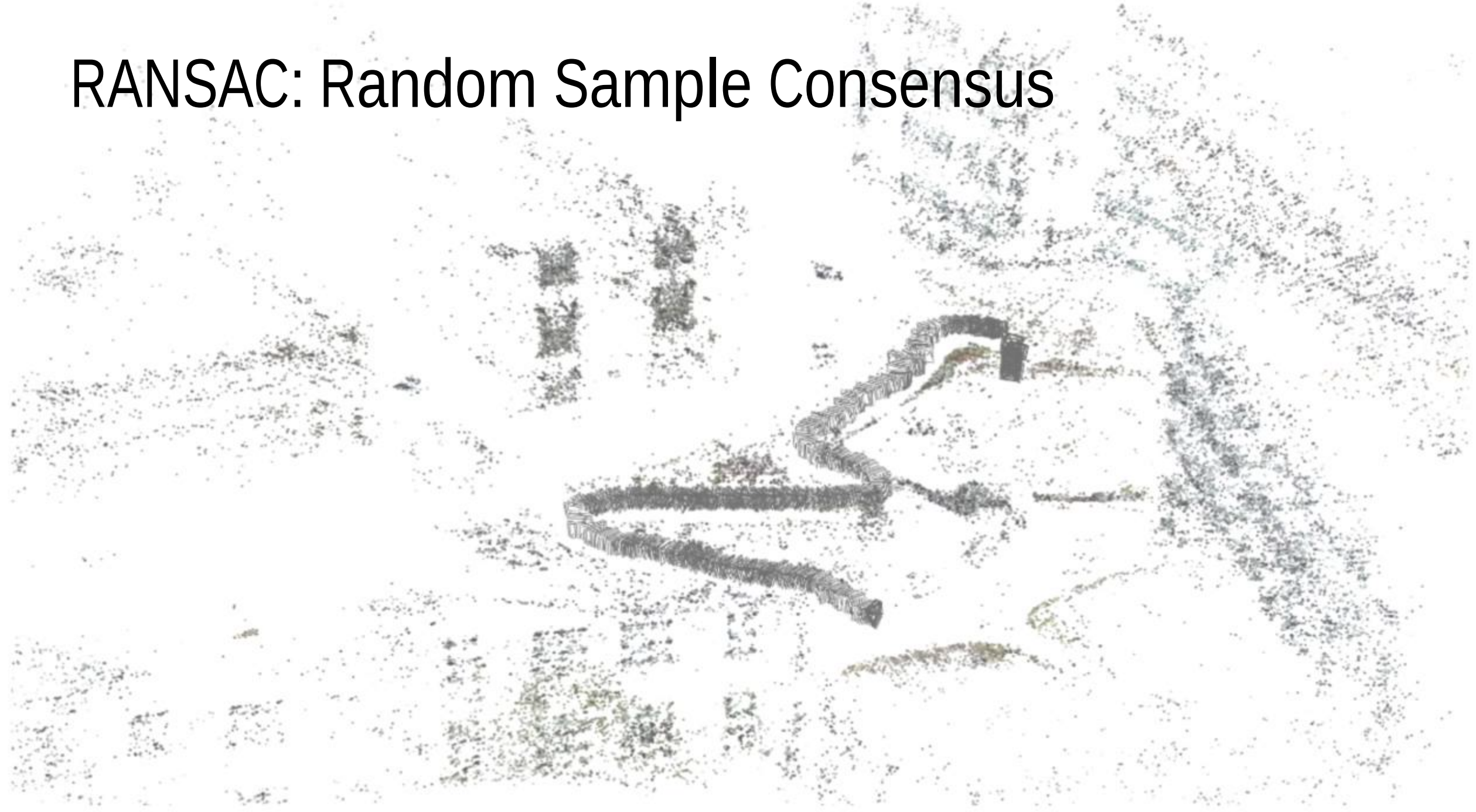
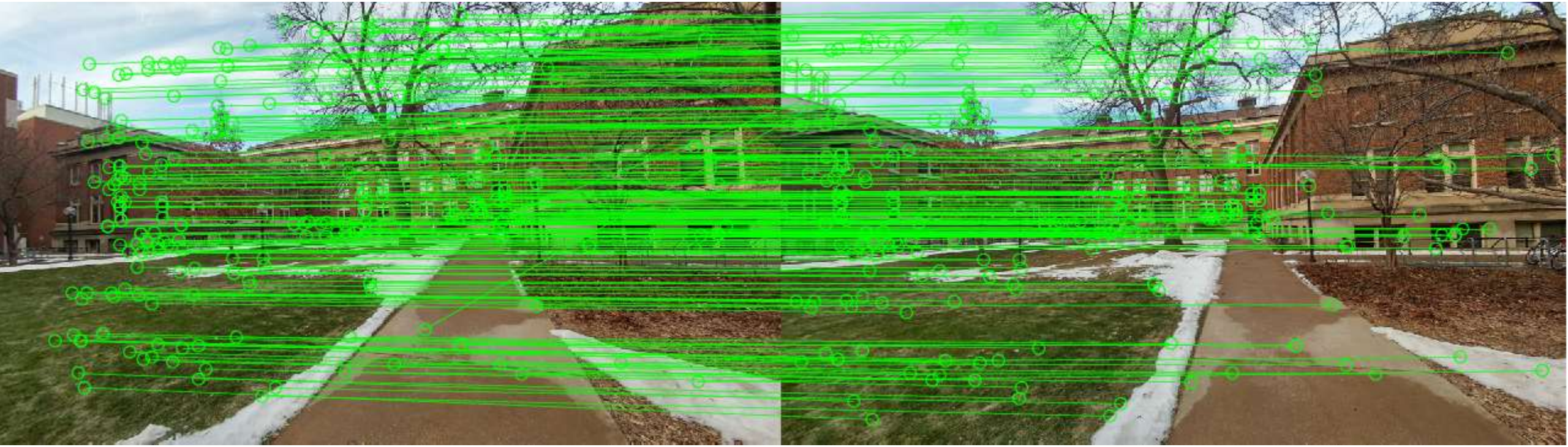
RANSAC: Random Sample Consensus



Fundamental Matrix Computation: Linear Least Squares

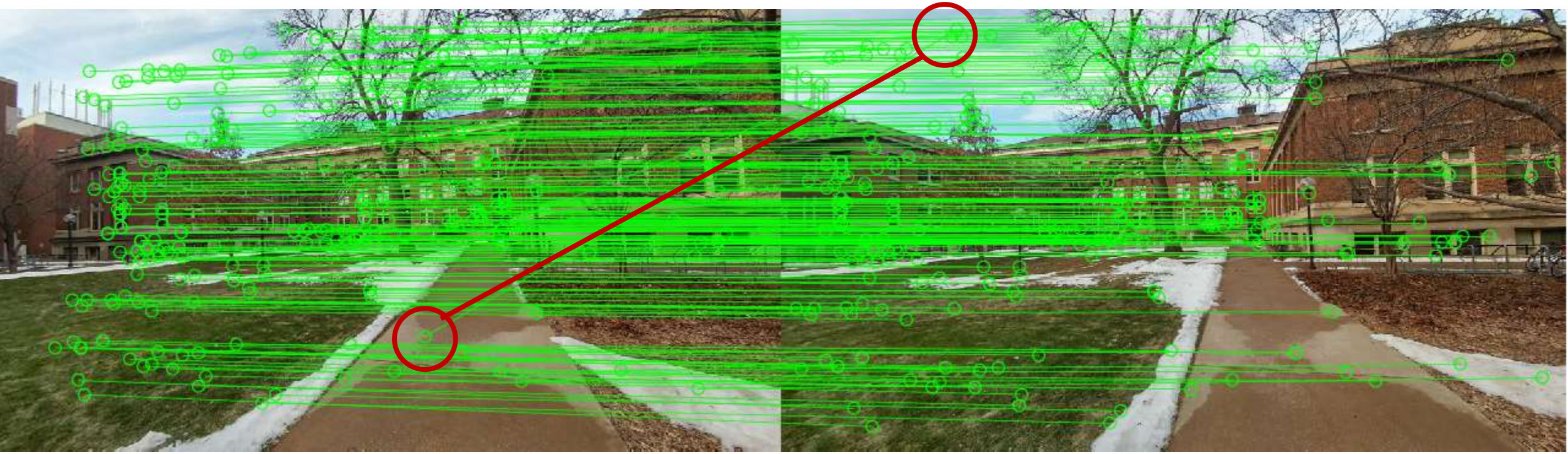
$$\begin{bmatrix} u_1^x v_1^x & u_1^y v_1^x & v_1^x & u_1^x v_1^y & u_1^y v_1^y & v_1^y & u_1^x & u_1^y & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_m^x v_m^x & u_m^y v_m^x & v_m^x & u_m^x v_m^y & u_m^y v_m^y & v_m^y & u_m^x & u_m^y & 1 \end{bmatrix} \mathbf{A} \quad \mathbf{X} \quad \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$



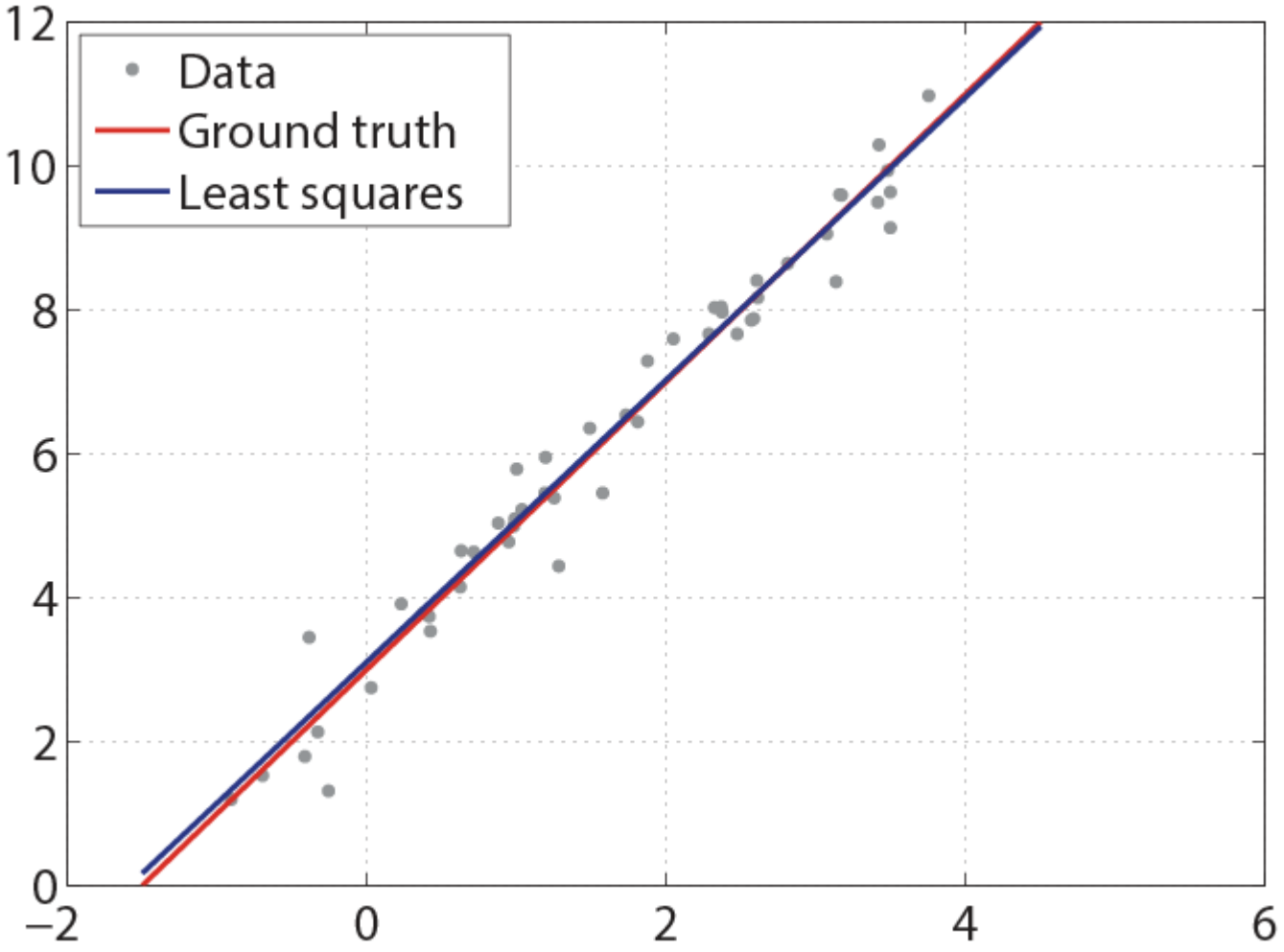
Fundamental Matrix Computation: Linear Least Squares

$$\begin{bmatrix} u_1^x v_1^x & u_1^y v_1^x & v_1^x & u_1^x v_1^y & u_1^y v_1^y & v_1^y & u_1^x & u_1^y & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_m^x v_m^x & u_m^y v_m^x & v_m^x & u_m^x v_m^y & u_m^y v_m^y & v_m^y & u_m^x & u_m^y & 1 \end{bmatrix} A \quad X \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Outlier?



Recall: Line Fitting (Ax=b)

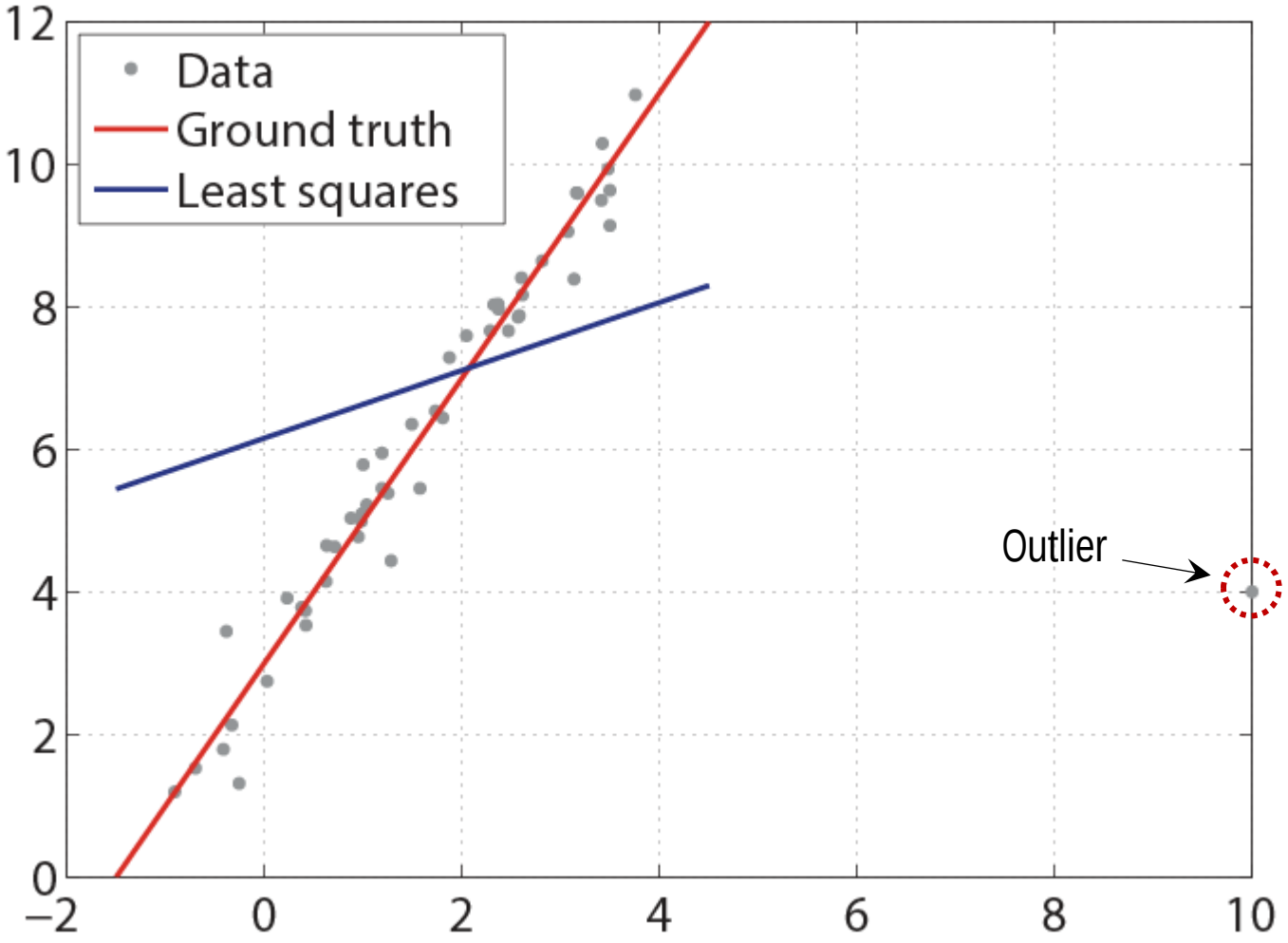


$$\begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \vdots & \vdots \\ u_n & 1 \end{bmatrix} \begin{bmatrix} m \\ d \end{bmatrix} \approx \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

$$A^T A x = A^T b$$

$$x = \left[A^T A \right]^{-1} A^T b$$

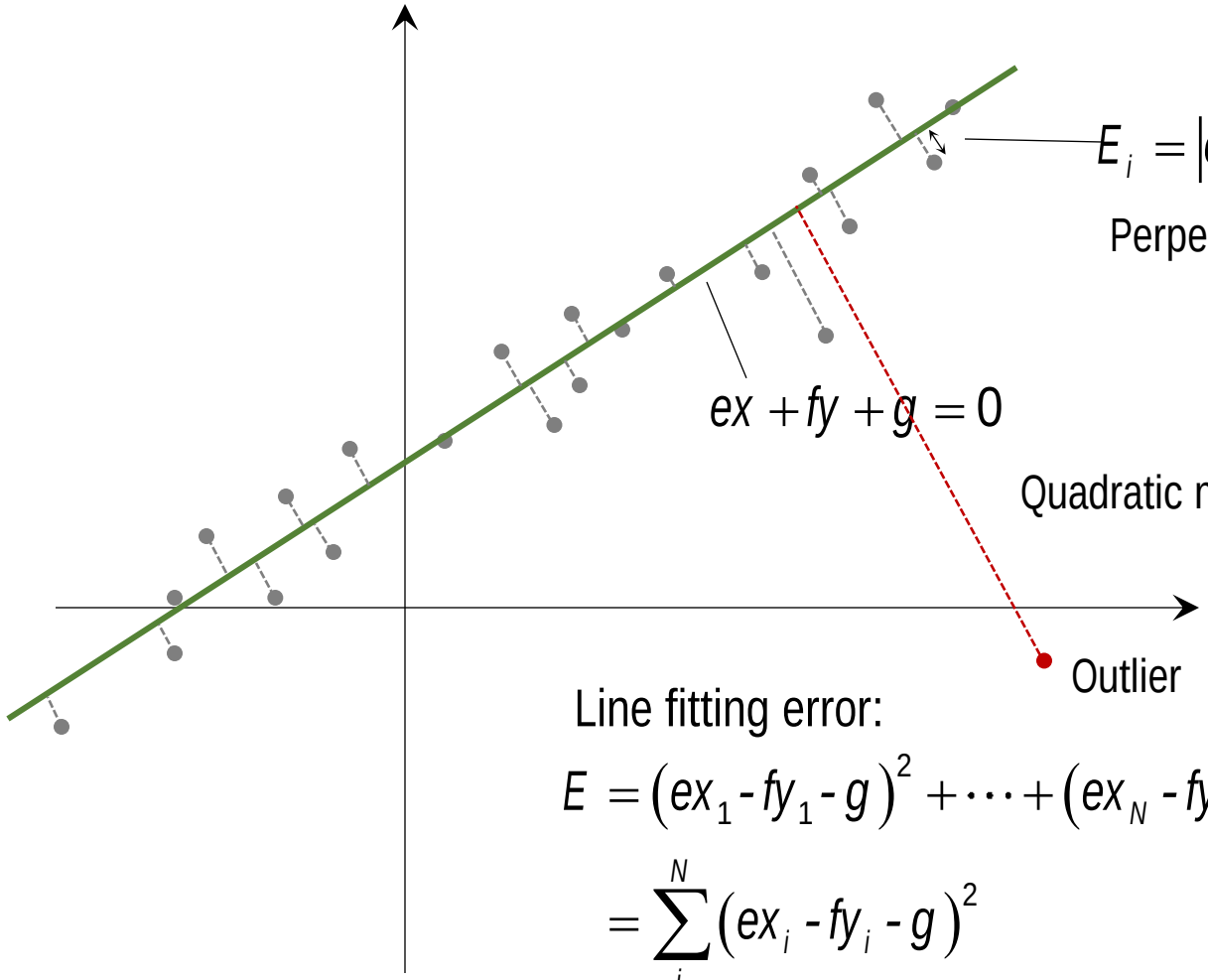
Outlier



$$\begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \vdots & \vdots \\ u_n & 1 \end{bmatrix} \begin{bmatrix} m \\ d \end{bmatrix} \approx \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

$$A^T A x = A^T b$$

$$x = \left[A^T A \right]^{-1} A^T b$$



$E_i = |ex_i + fy_i + g|$

Perpendicular distance

$ex + fy + g = 0$

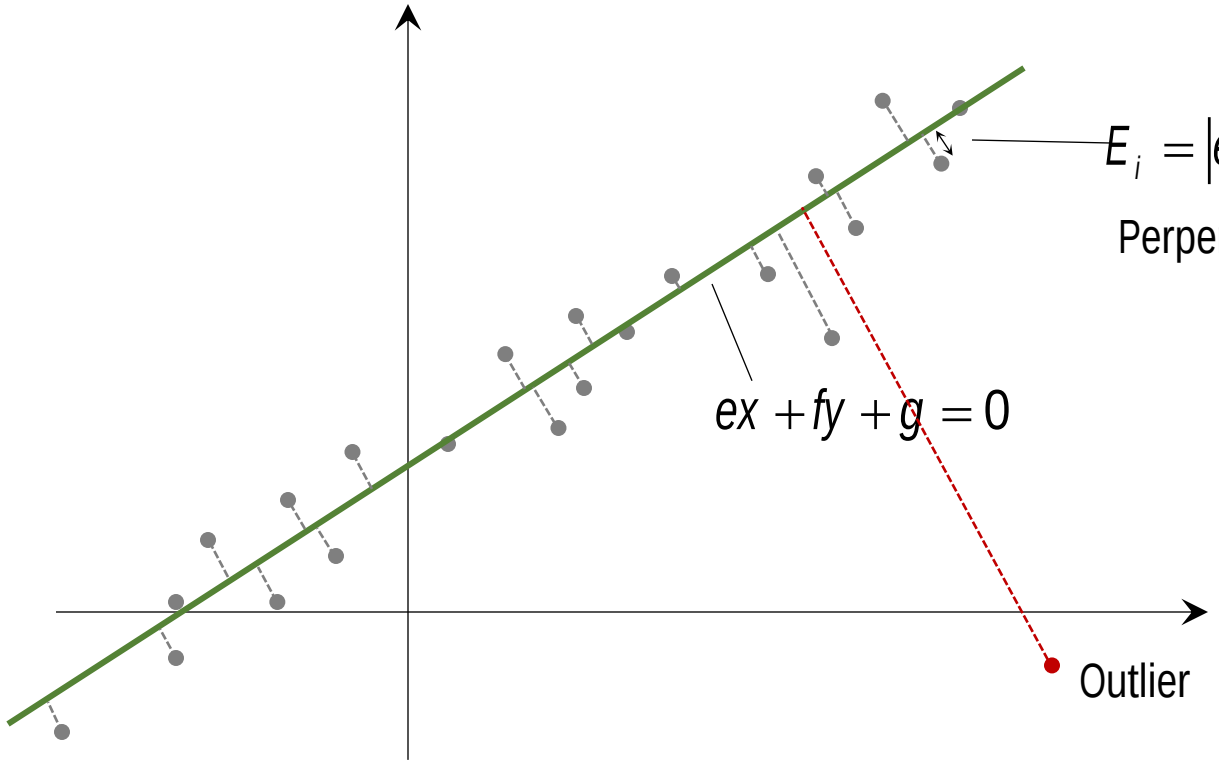
Quadratic magnification of error of outliers

Outlier

Line fitting error:

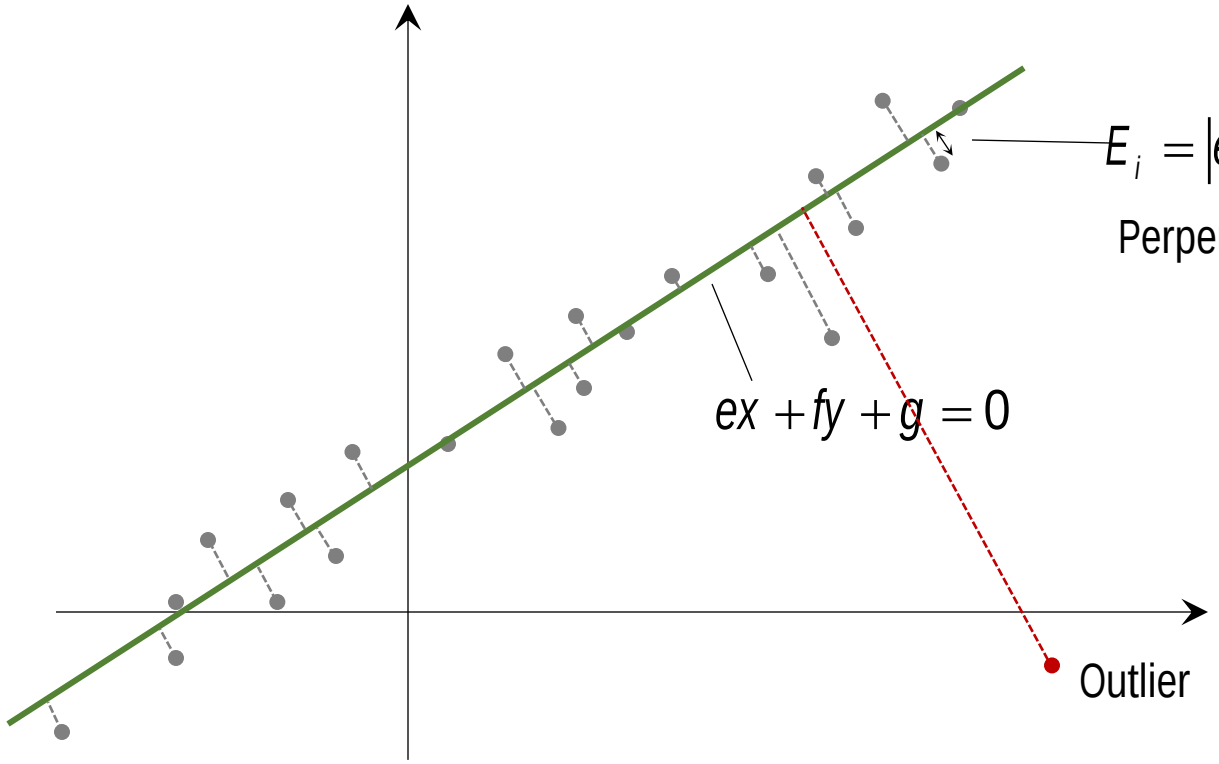
$$E = (ex_1 - fy_1 - g)^2 + \cdots + (ex_N - fy_N - g)^2$$

$$= \sum_i^N (ex_i - fy_i - g)^2$$



Outlier rejection strategy:

To find the best line that explanes the maximum number of points.



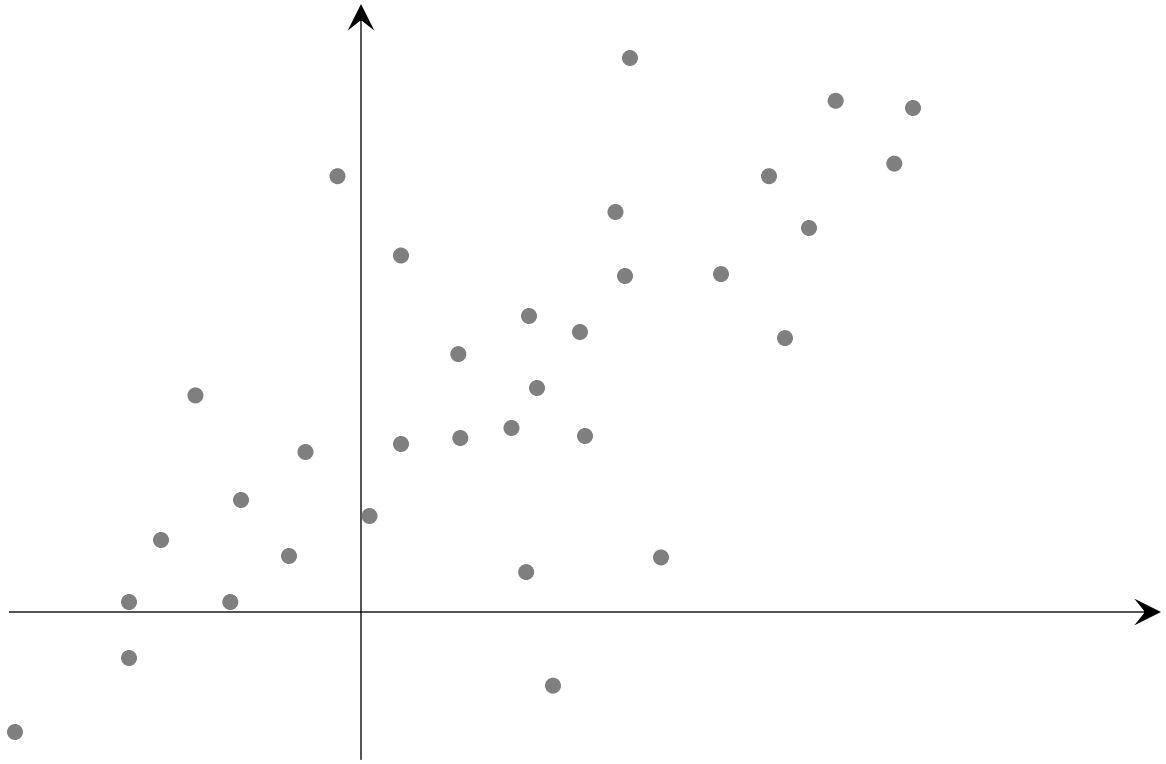
Outlier rejection strategy:

To find the best line that explanes the maximum number of points.

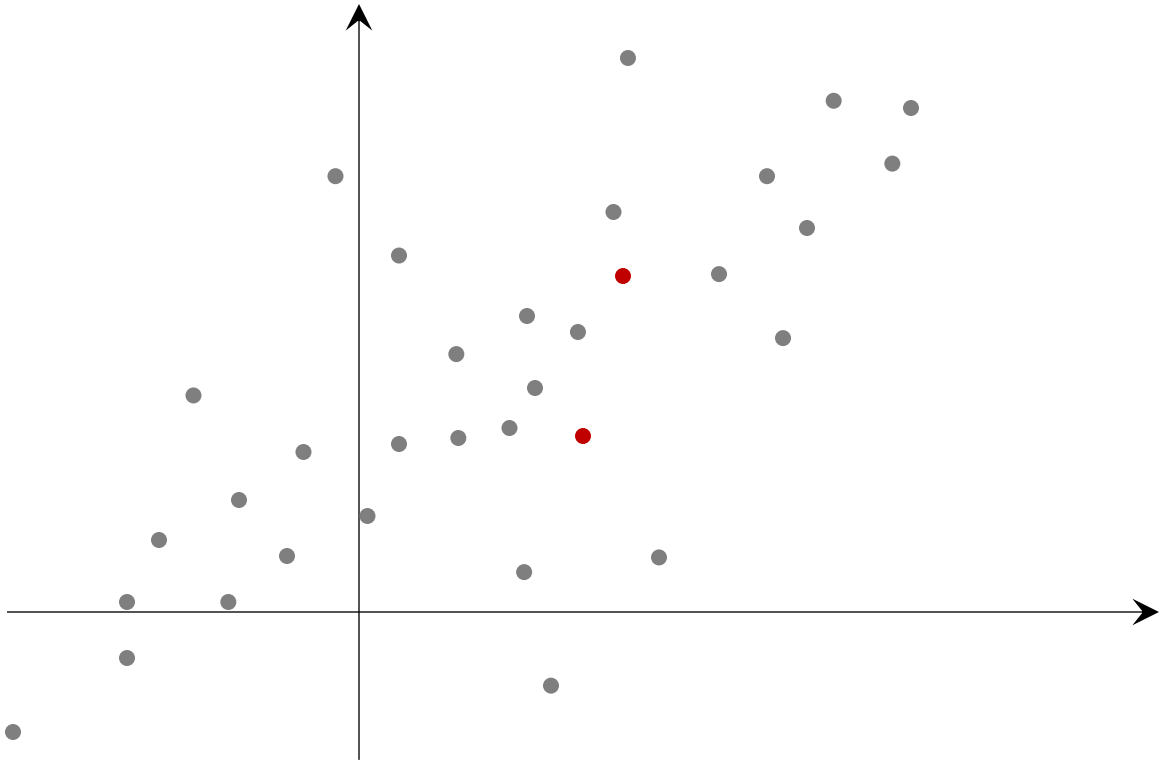
Assumptions:

1. Majority of good samples agree with the underlying model (good apples are same and simple.).

2. Bad samples does not consistently agree with a single model (all bad apples are different and complicated.).

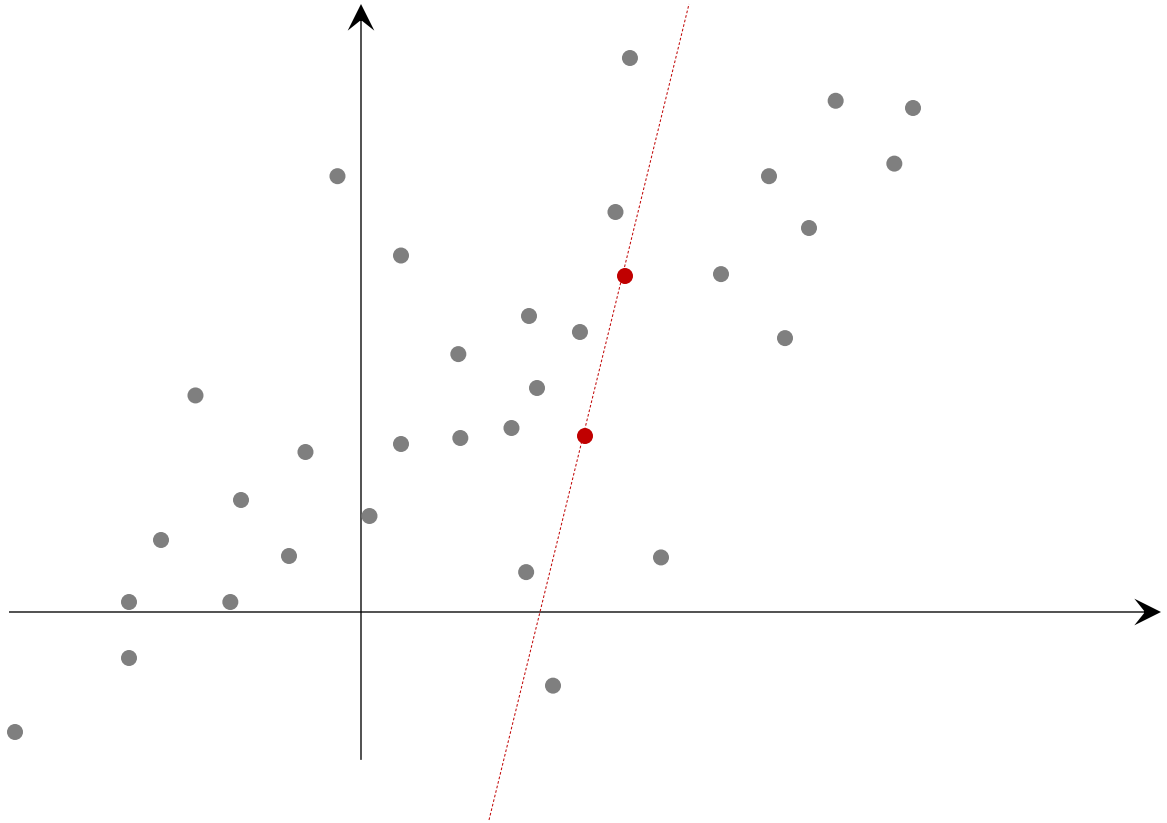


RANSAC: Random Sample Consensus



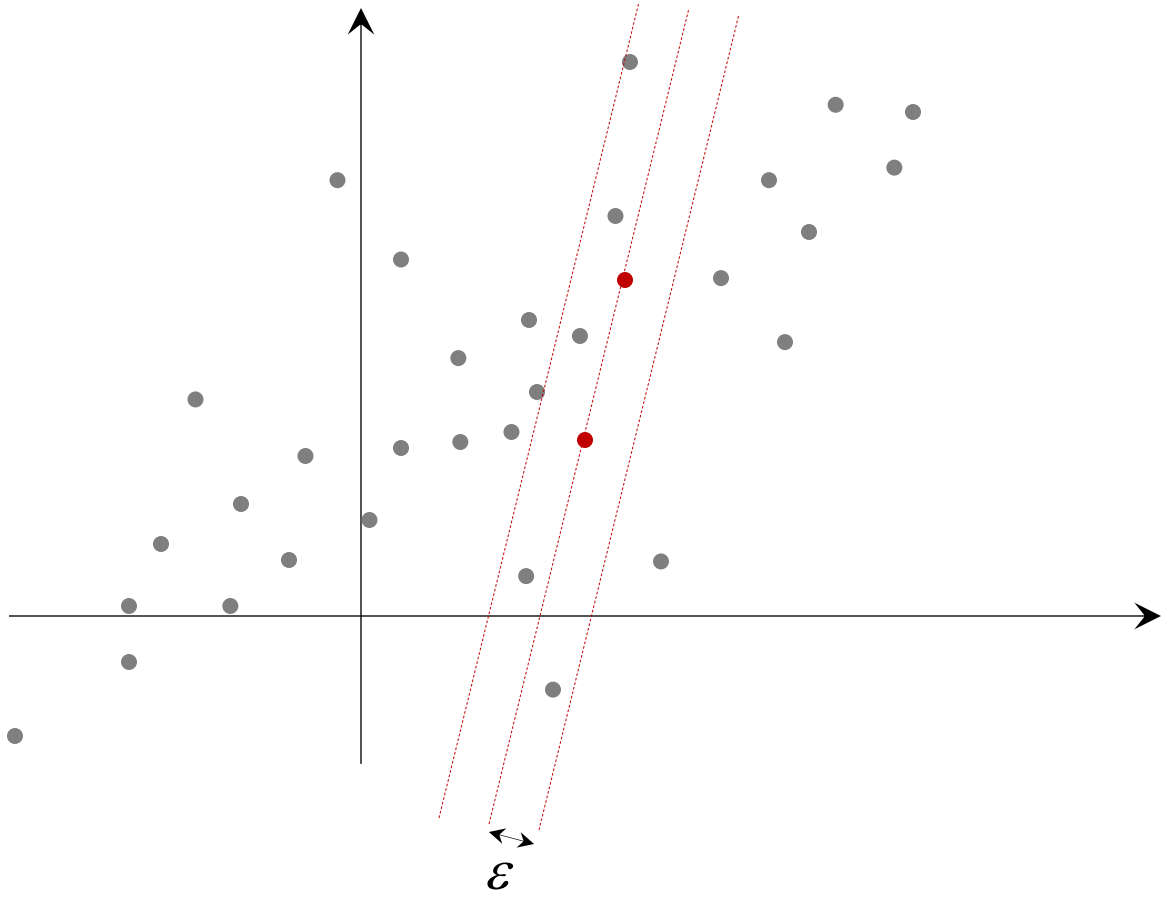
1. Random sampling

RANSAC: Random Sample Consensus



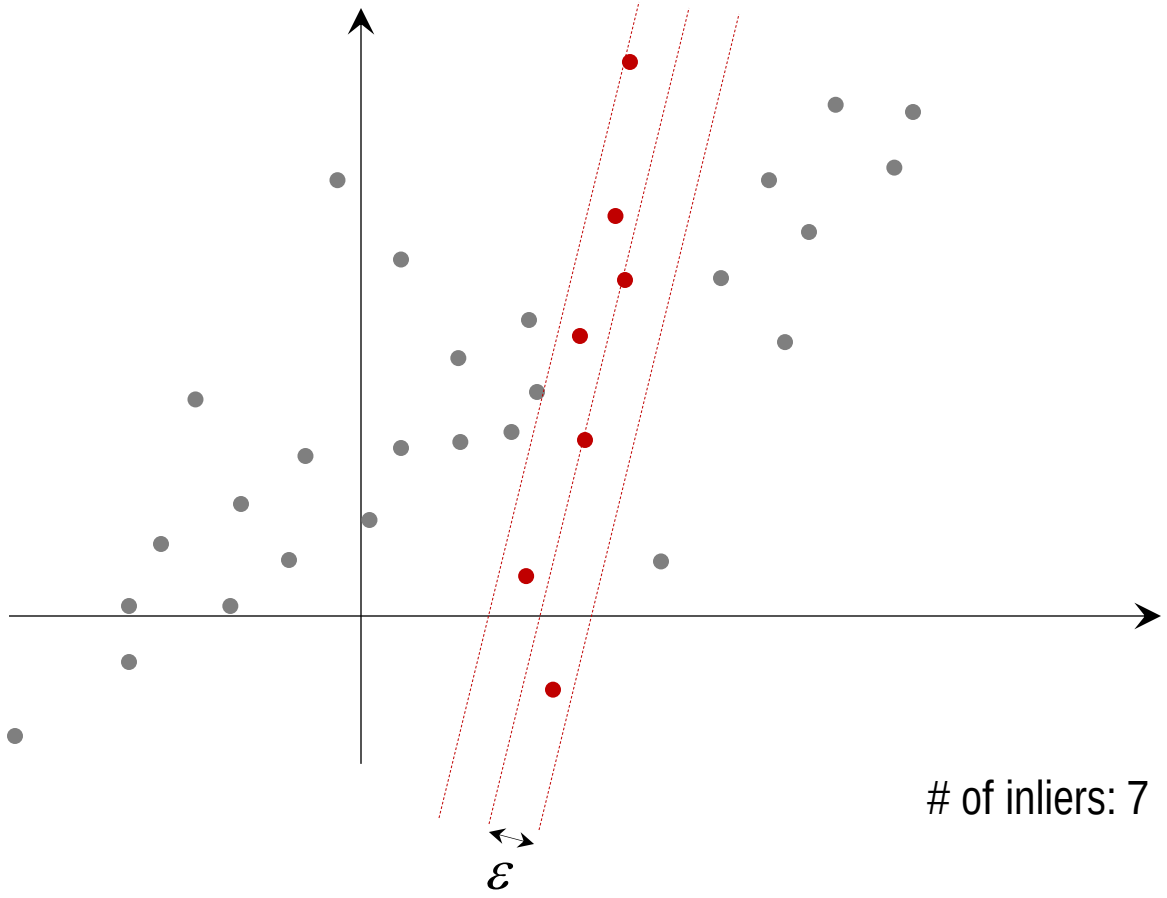
1. Random sampling
2. Model building

RANSAC: Random Sample Consensus



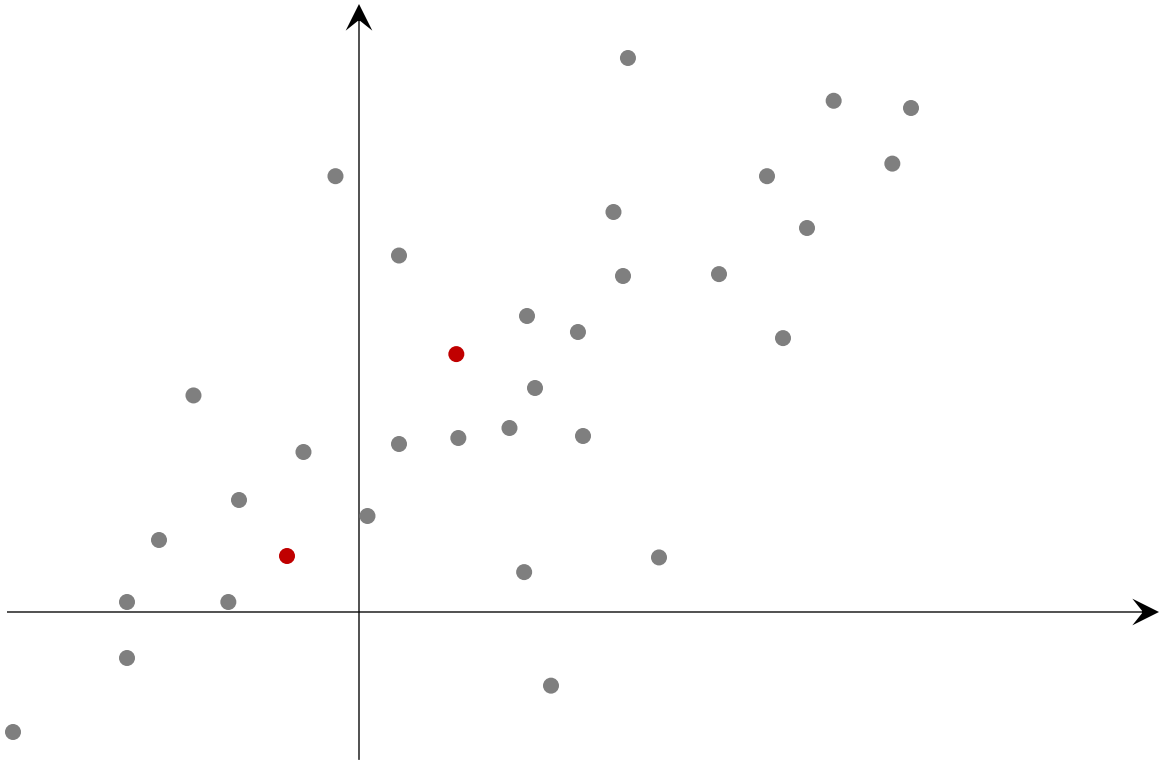
1. Random sampling
2. Model building
3. Thresholding

RANSAC: Random Sample Consensus



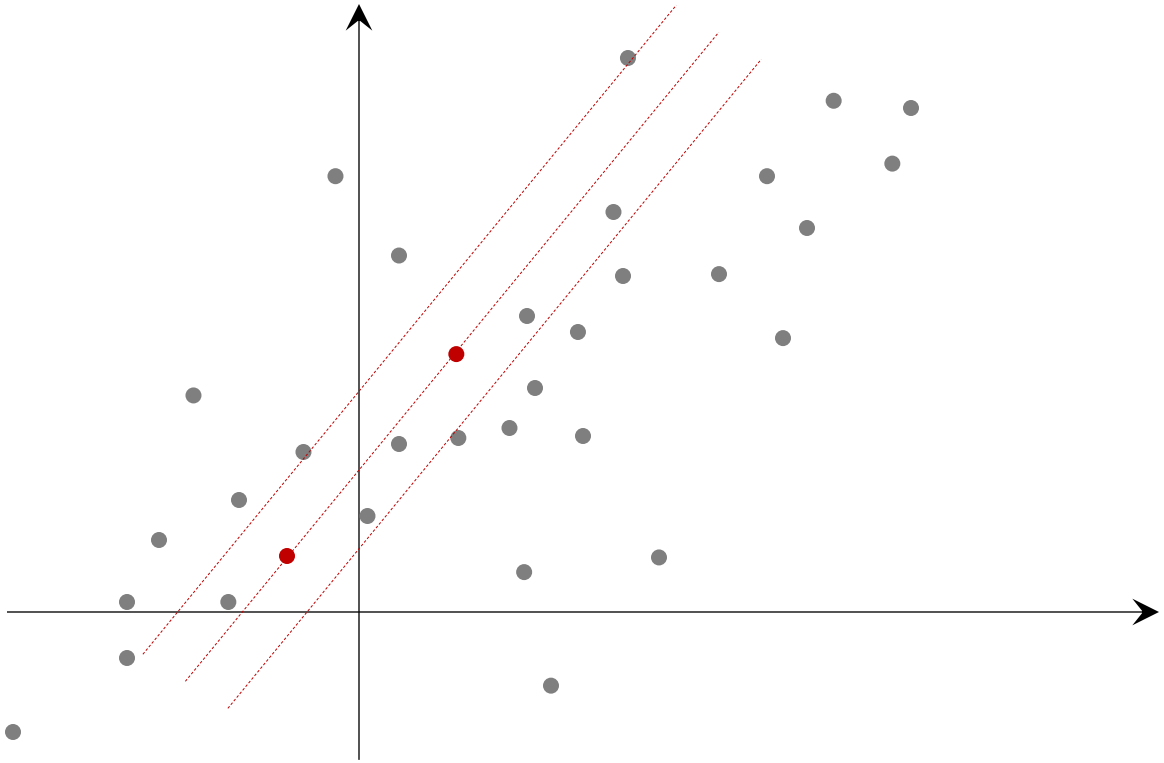
1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

RANSAC: Random Sample Consensus



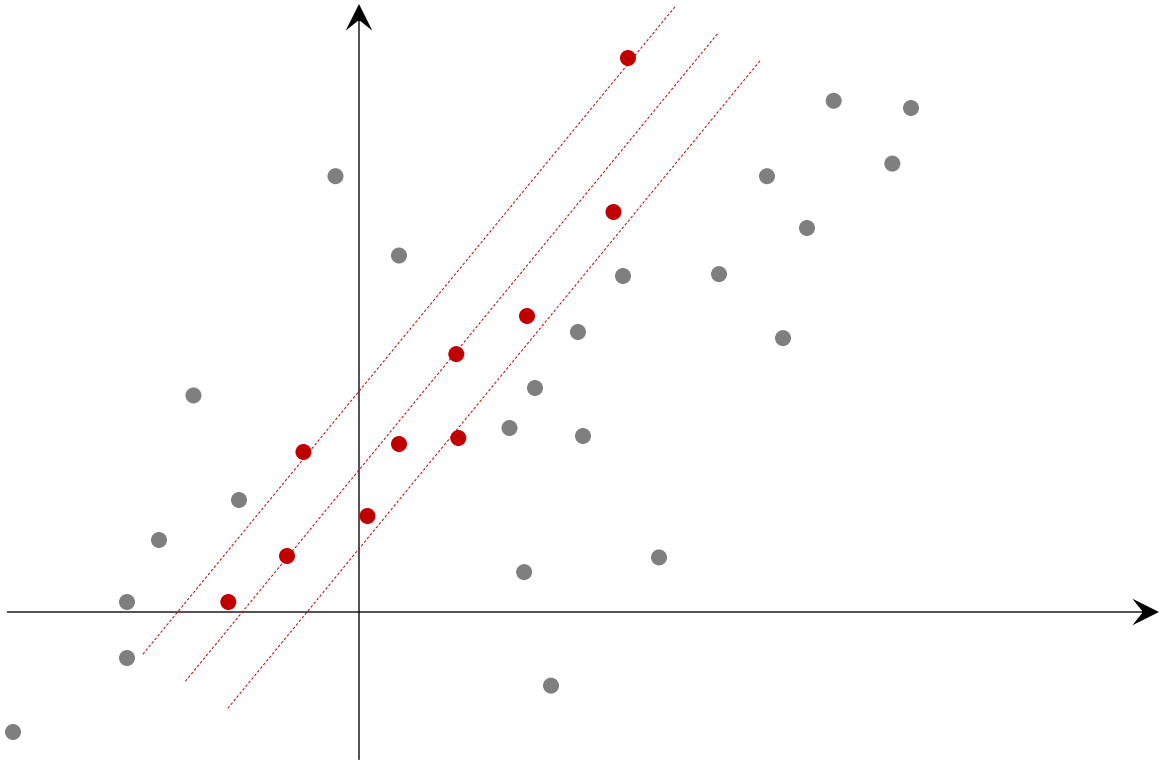
1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

RANSAC: Random Sample Consensus



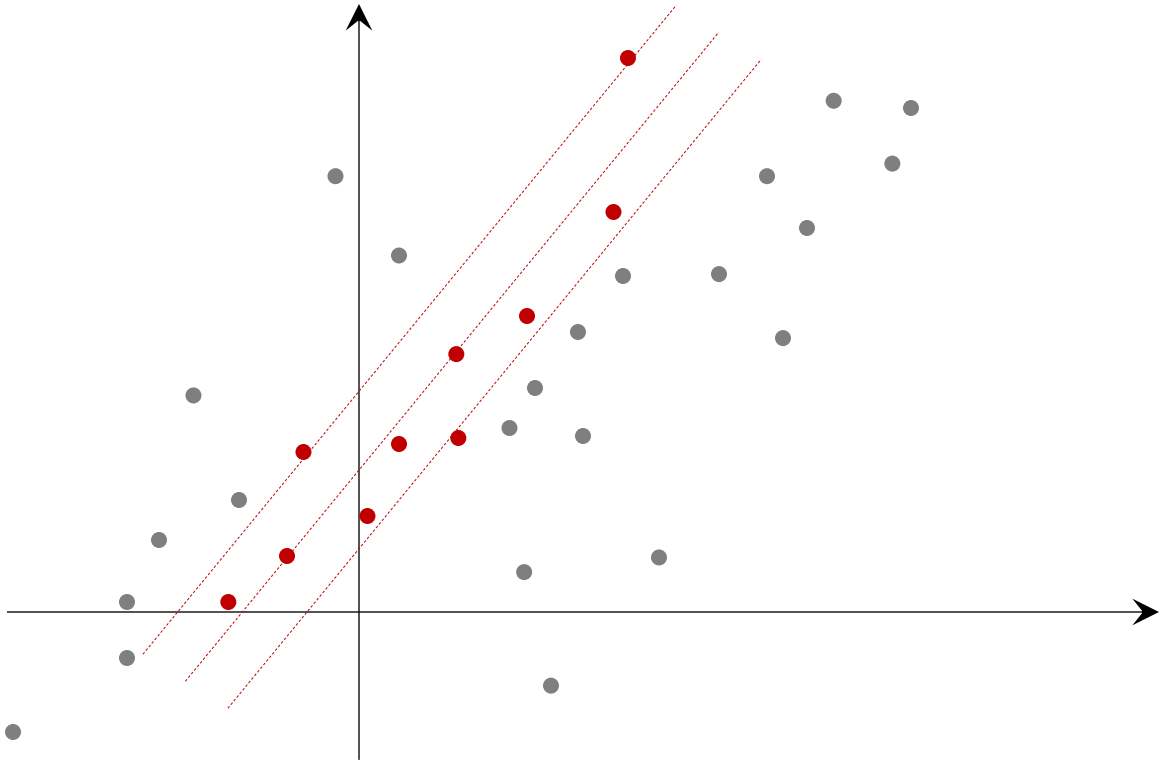
1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

RANSAC: Random Sample Consensus



1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

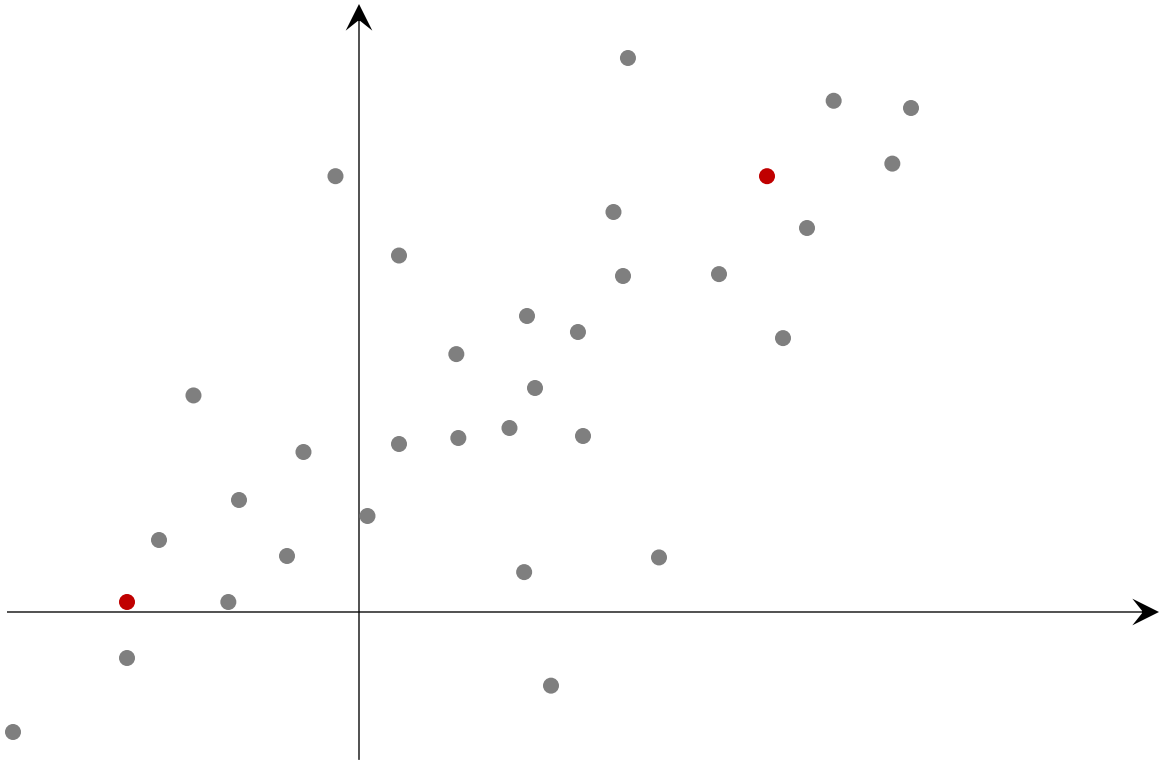
RANSAC: Random Sample Consensus



# of inliers: 10

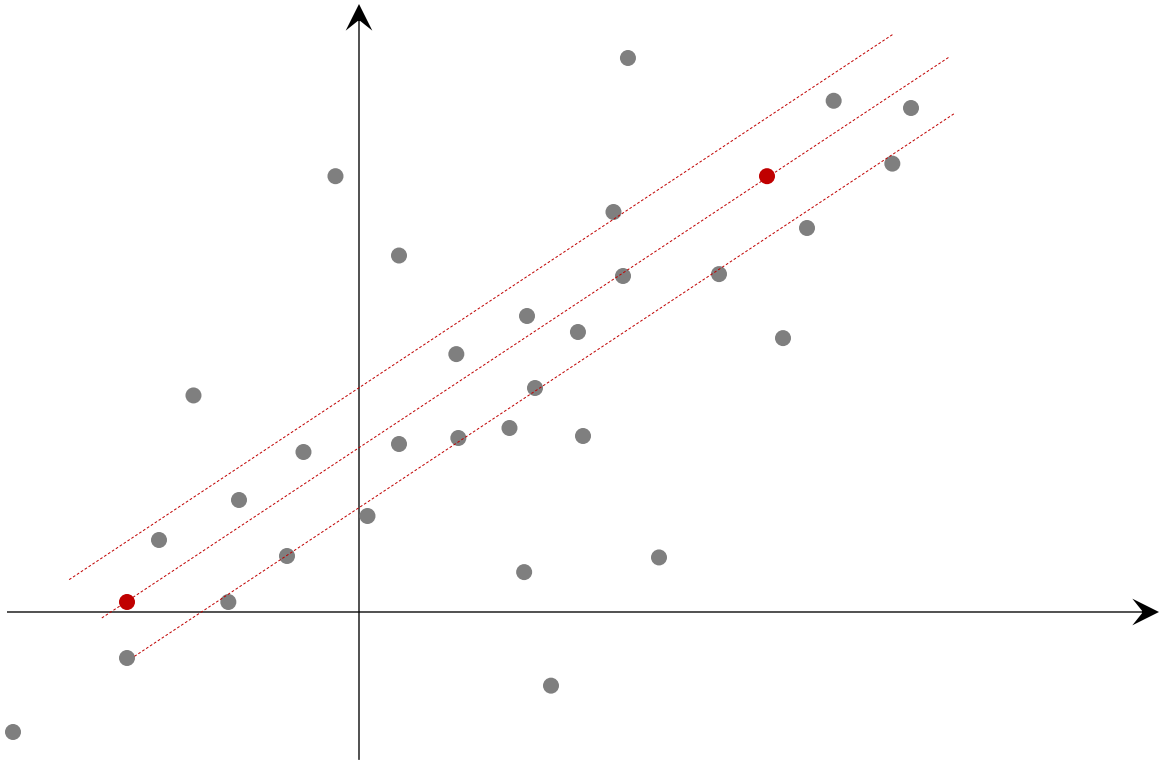
1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

RANSAC: Random Sample Consensus



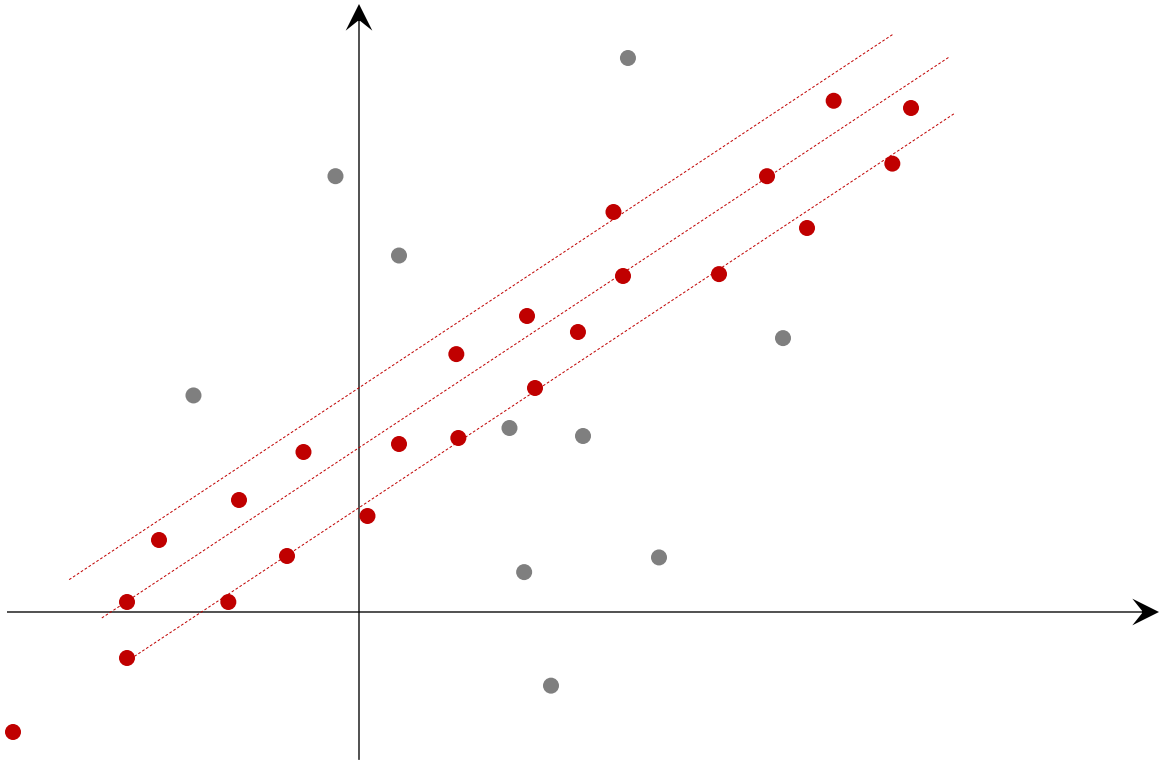
1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

RANSAC: Random Sample Consensus



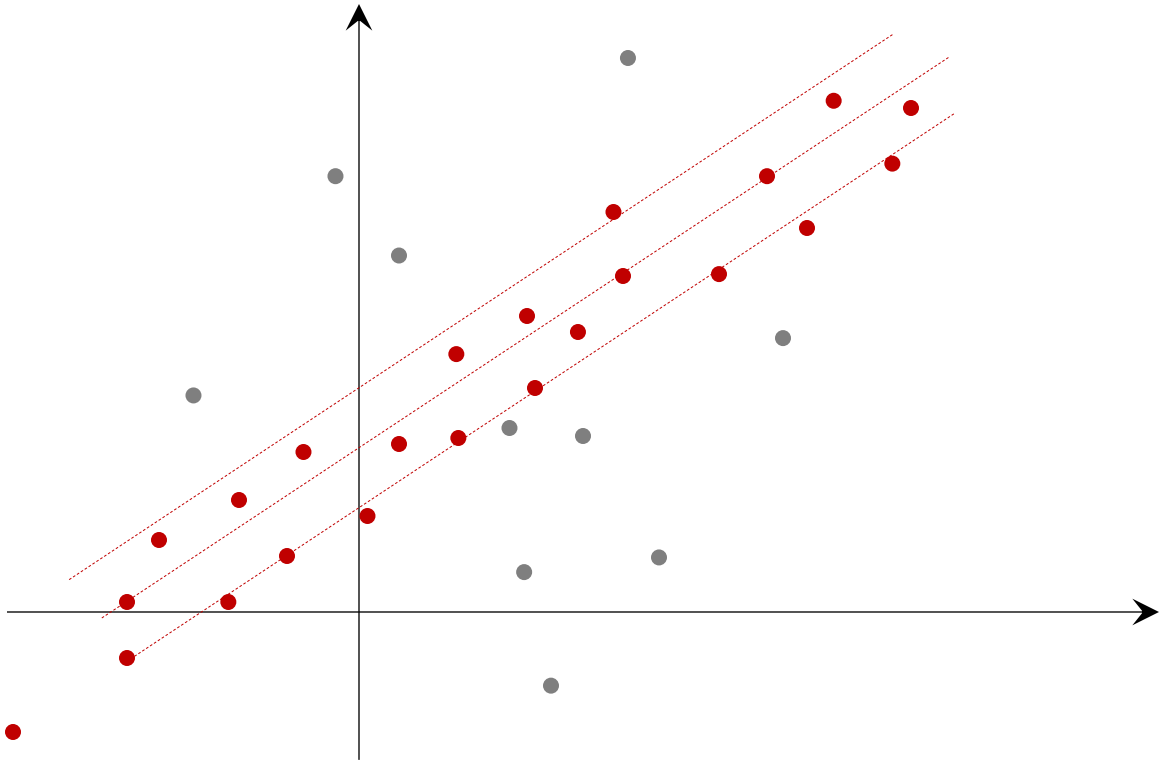
1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

RANSAC: Random Sample Consensus



1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

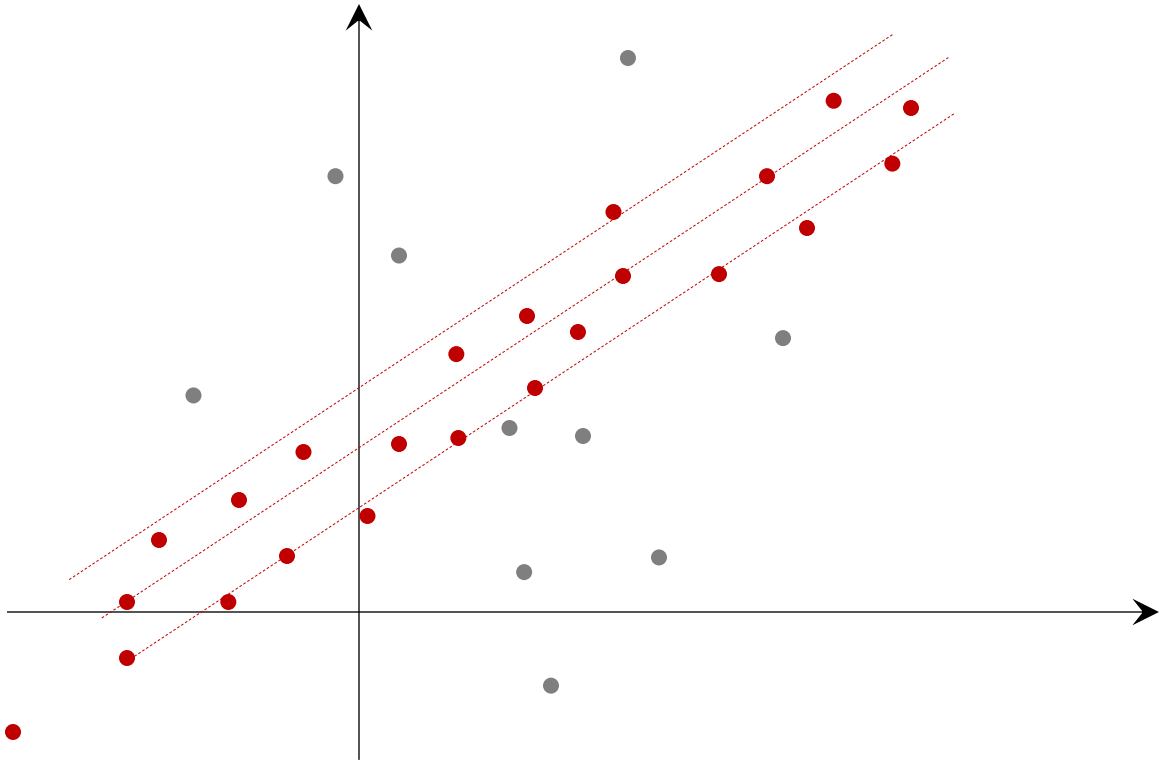
RANSAC: Random Sample Consensus



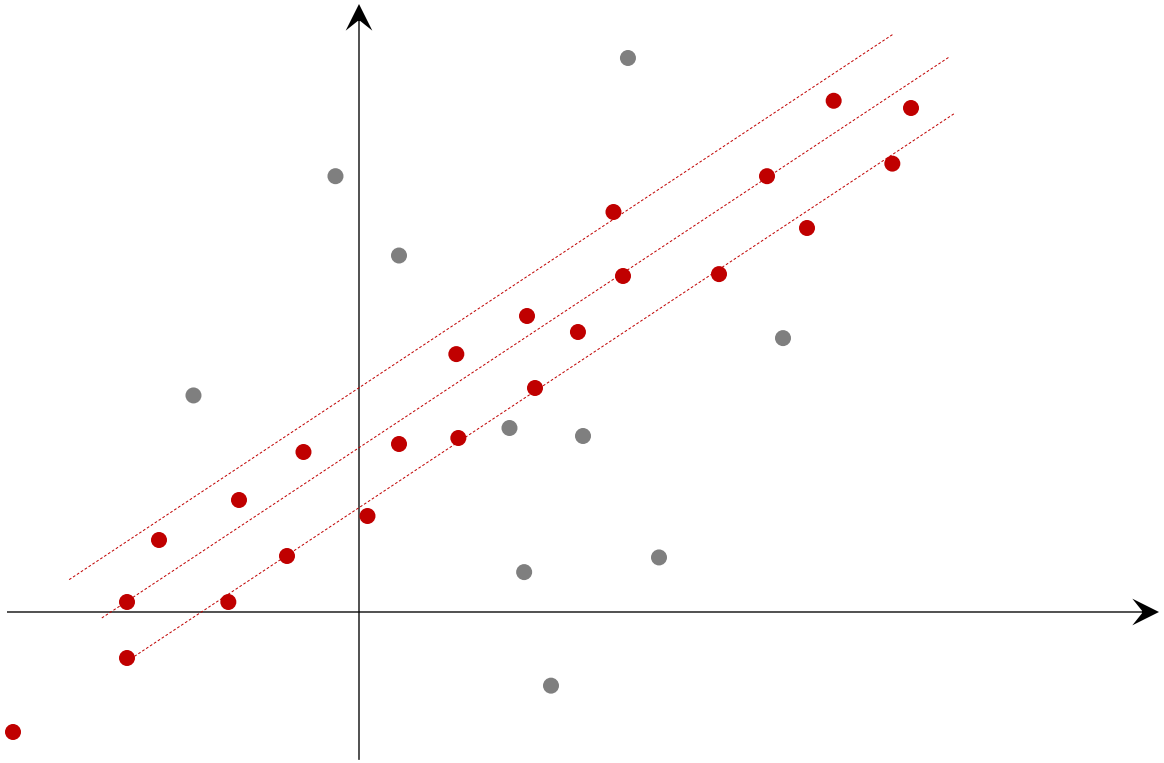
1. Random sampling
2. Model building
3. Thresholding
4. Inlier counting

$$\frac{\text{\# of inliers: 23}}{\text{Maximum number of inliers}}$$

RANSAC: Random Sample Consensus



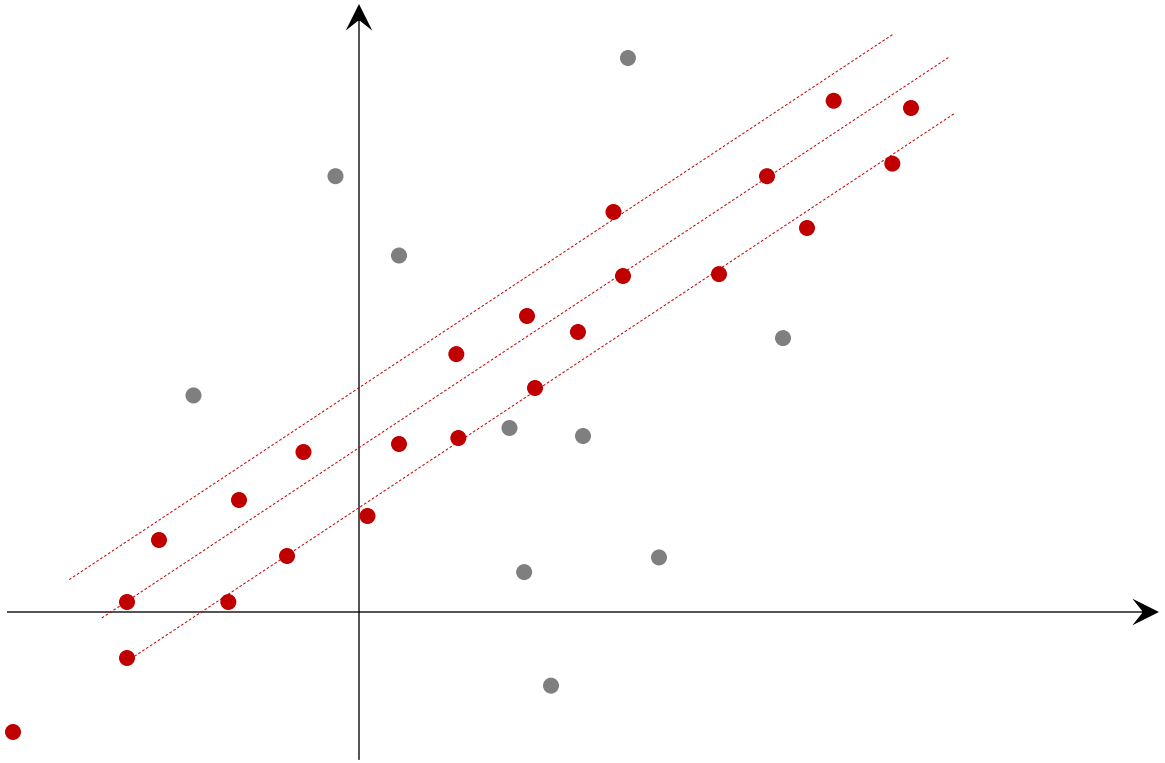
Required number of iterations with p success rate:



Required number of iterations with p success rate:

Probability of choosing an inlier:

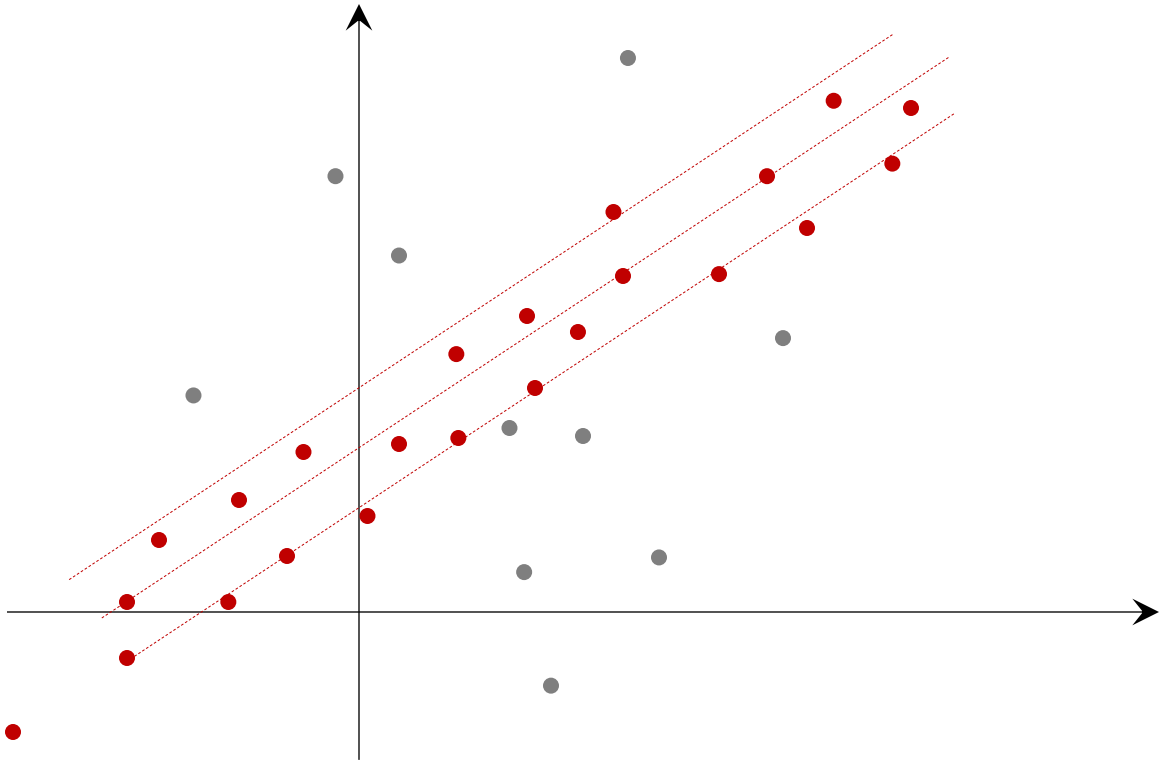
$$w = \frac{\#\,\text{of inliers}}{\#\,\text{of samples}}$$



Required number of iterations with p success rate:

Probability of choosing an inlier: $$w = \frac{\#\,\text{of inliers}}{\#\,\text{of samples}}$$

Probability of building a correct model: w^n where n is the number of samples to build a model.

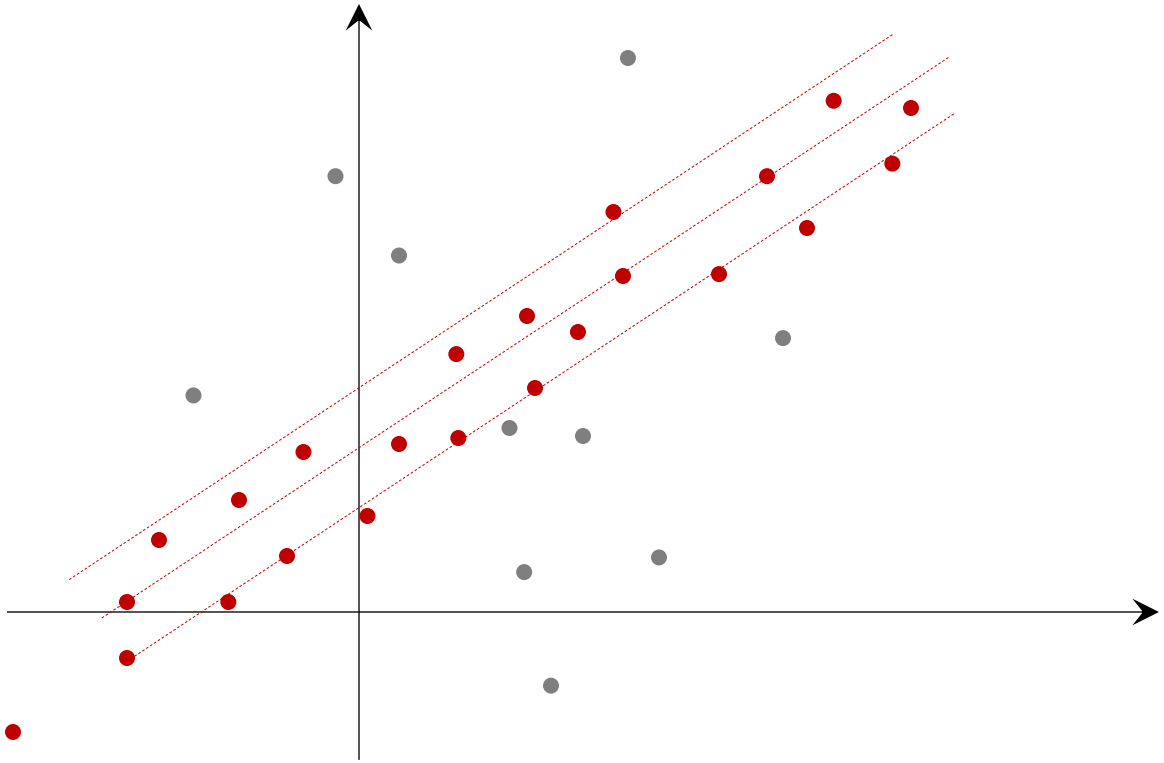


Required number of iterations with p success rate:

Probability of choosing an inlier: $$w = \frac{\#\text{ of inliers}}{\#\text{ of samples}}$$

Probability of building a correct model: w^n where n is the number of samples to build a model.

Probability of not building a correct model during k iterations: $\left(1\text{-}w^n\right)^k$



Required number of iterations with p success rate:

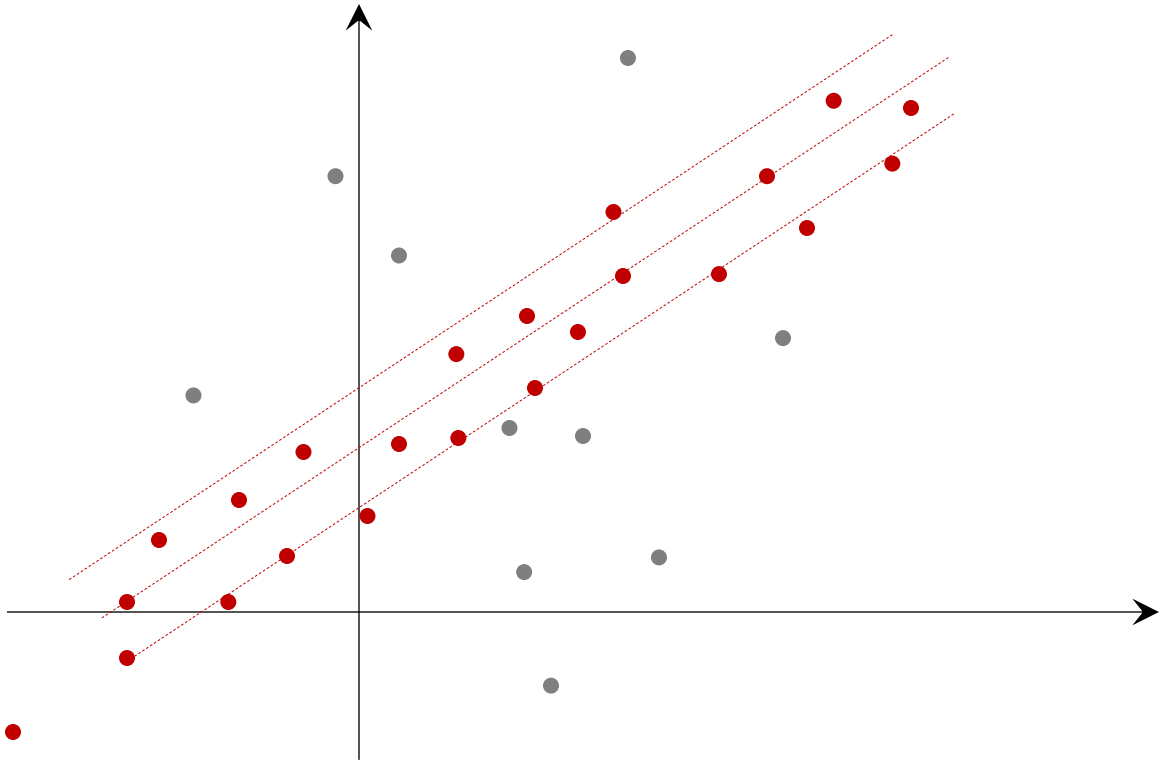
Probability of choosing an inlier: $$w = \frac{\#\text{ of inliers}}{\#\text{ of samples}}$$

Probability of building a correct model: w^n where n is the number of samples to build a model.

Probability of not building a correct model during k iterations: $\left(1-w^n\right)^k$

$\left(1-w^n\right)^k = 1-p$ where p is desired RANSAC success rate.

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



Required number of iterations with p success rate:

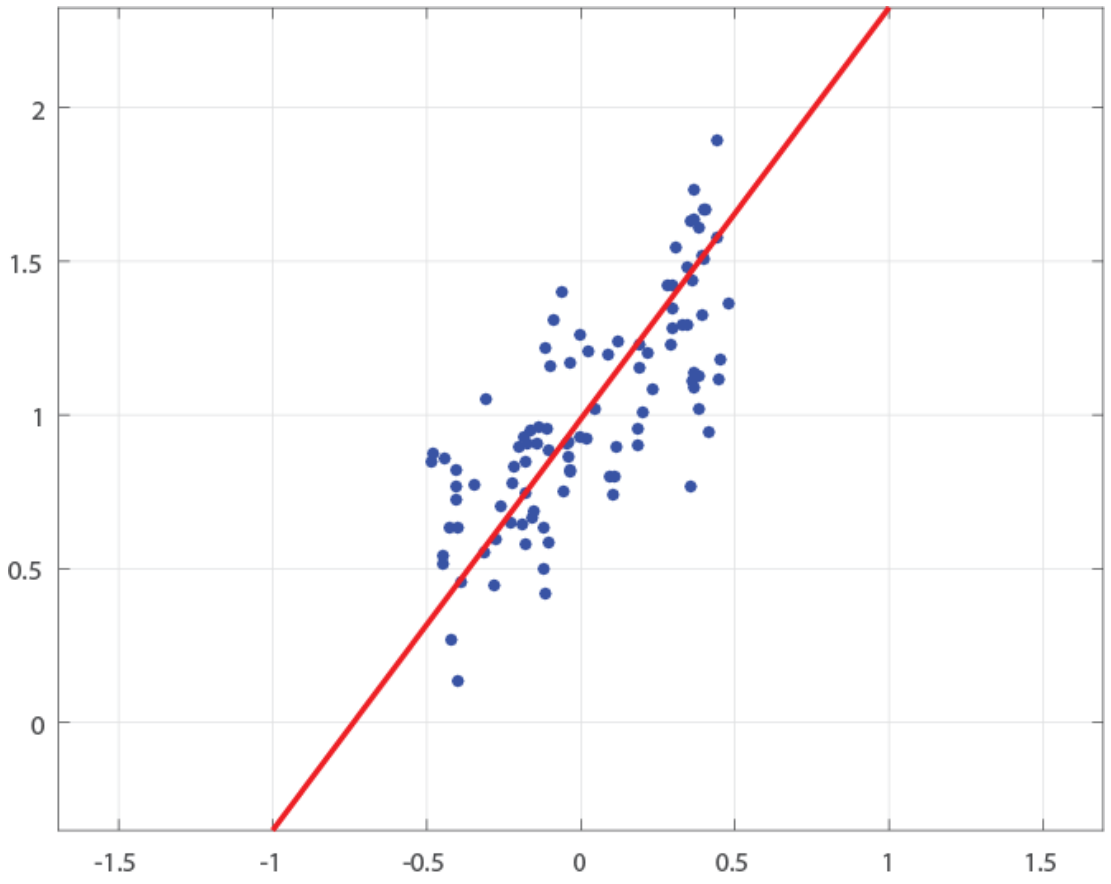
$$k = \frac{\log(1-p)}{\log(1-w^n)} \qquad \text{where } w = \frac{\#\text{ of inliers}}{\#\text{ of samples}}$$

Probability of choosing an inlier: $w = \frac{\#\text{ of inliers}}{\#\text{ of samples}}$

Probability of building a correct model: w^n where n is the number of samples to build a model.

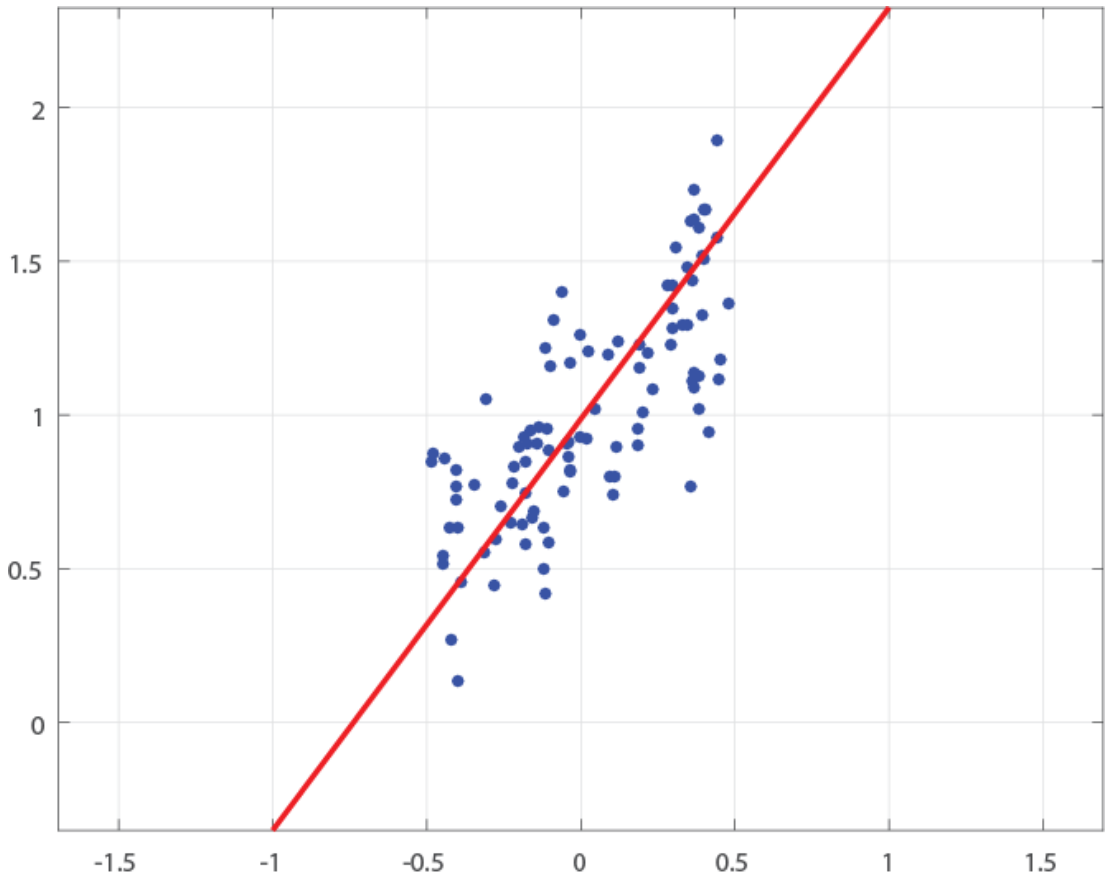
Probability of not building a correct model during k iterations: $(1-w^n)^k$

$(1-w^n)^k = 1-p$ where p is desired RANSAC success rate. $k = \frac{\log(1-p)}{\log(1-w^n)}$




```
function RANSAC_line

a = 1;
b = 1;

nPoints = 100;
x = rand(nPoints,1)-0.5;
y = a*x + b + 0.2*randn(nPoints,1);
```

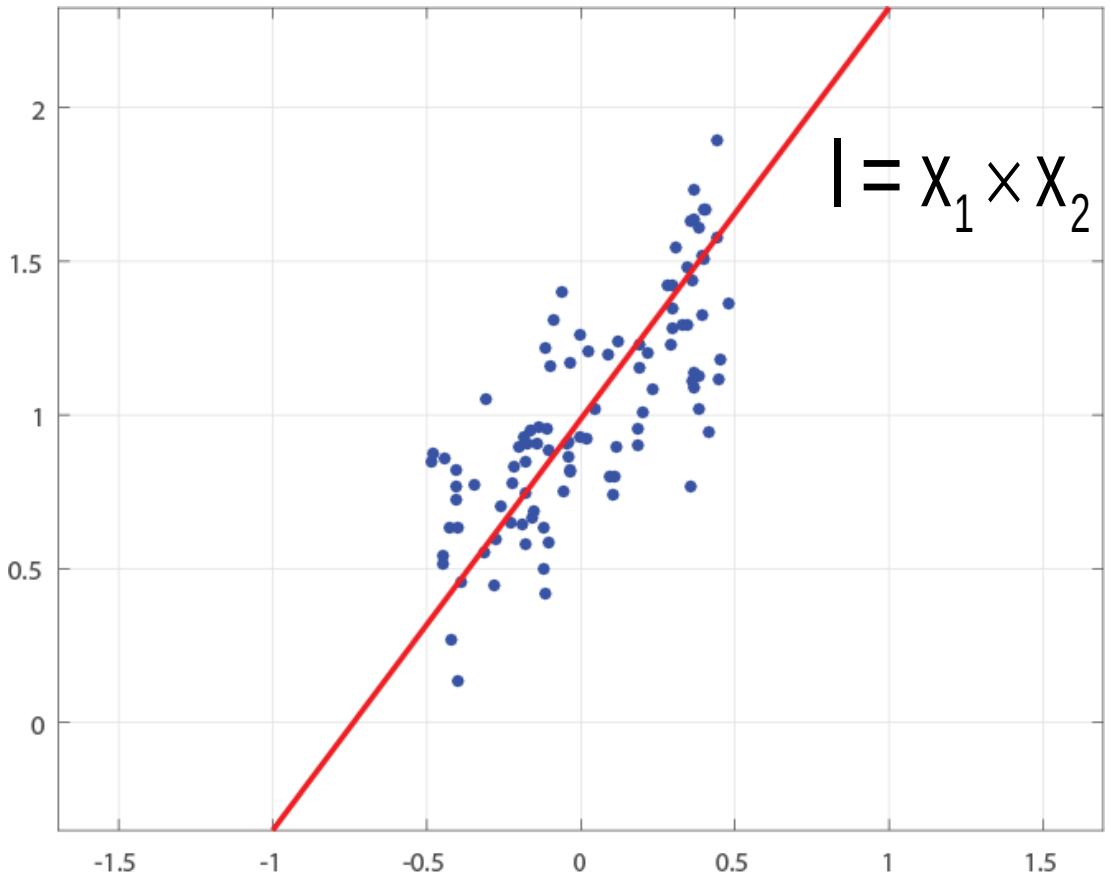
Generating samples




```
function RANSAC_line

a = 1;
b = 1;

nPoints = 100;
x = rand(nPoints,1)-0.5;
y = a*x + b + 0.2*randn(nPoints,1);

nRansacIter = 500;
threshold = 0.1;
max_nInliers = 0;
for i = 1 : nRansacIter
    r = randperm(nPoints);
    s1 = [x(r(1)); y(r(1)); 1];
    s2 = [x(r(2)); y(r(2)); 1];
```

Generating samples

Random sampling

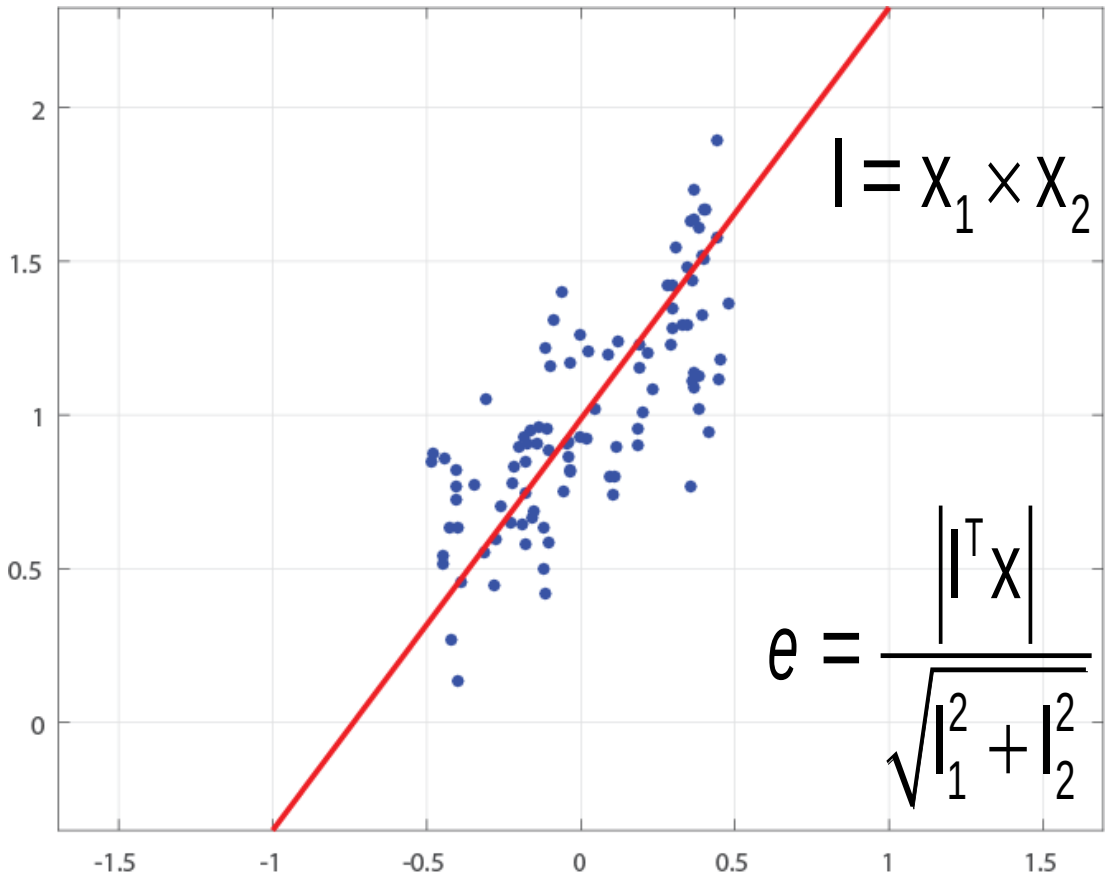



```
function RANSAC_line

a = 1;
b = 1;

nPoints = 100;
x = rand(nPoints,1)-0.5;
y = a*x + b + 0.2*randn(nPoints,1);

nRansacIter = 500;
threshold = 0.1;
max_nInliers = 0;
for i = 1 : nRansacIter
   r = randperm(nPoints);
   s1 = [x(r(1)); y(r(1)); 1];
   s2 = [x(r(2)); y(r(2)); 1];

   l = GetLineFromTwoPoints(s1, s2);
```

Generating samples

Random sampling

Model building



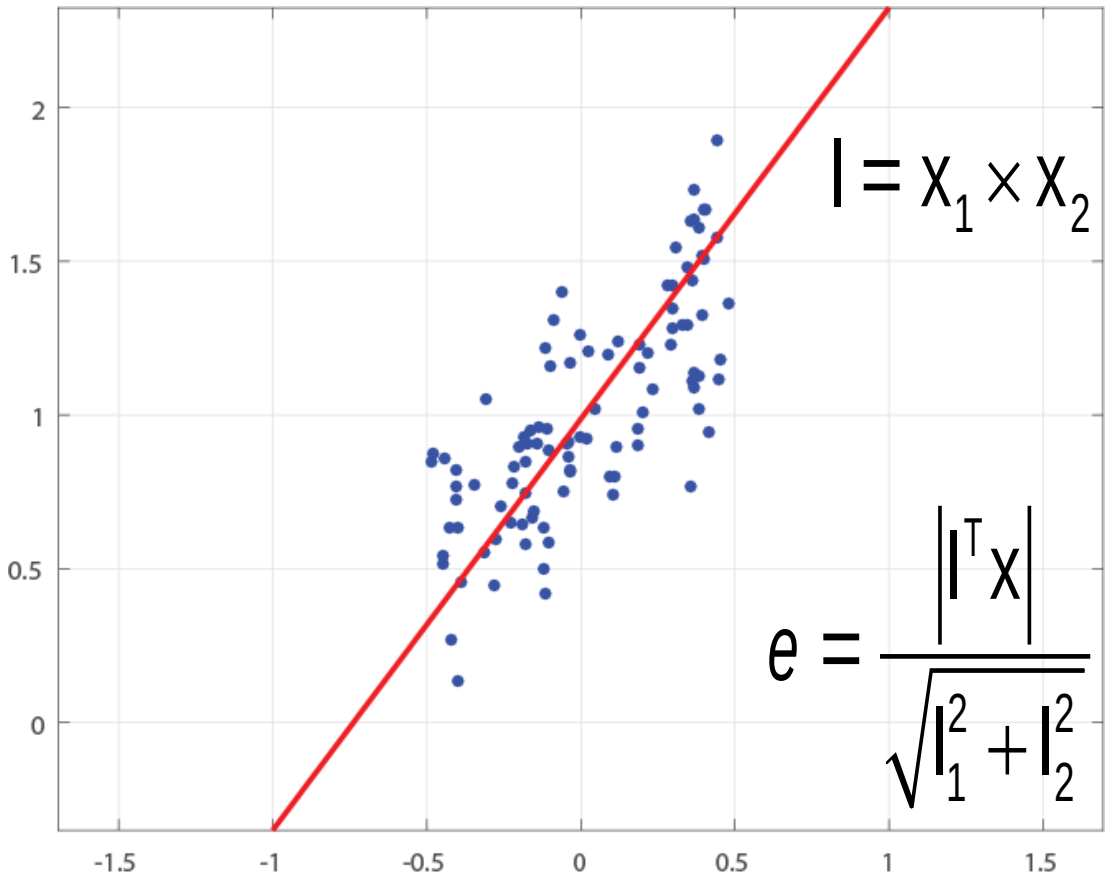

```
function RANSAC_line

a = 1;
b = 1;

nPoints = 100;
x = rand(nPoints,1)-0.5;
y = a*x + b + 0.2*randn(nPoints,1);

nRansacIter = 500;
threshold = 0.1;
max_nInliers = 0;
for i = 1 : nRansacIter
   r = randperm(nPoints);
   s1 = [x(r(1)); y(r(1)); 1];
   s2 = [x(r(2)); y(r(2)); 1];

   l = GetLineFromTwoPoints(s1, s2);

   nInliers = 0;
   for j = 1 : nPoints
      e = abs(l' * [x(j); y(j); 1])/norm(l(1:2));
      if (e < threshold)
         nInliers = nInliers + 1;
      end
   end
```

Generating samples

Random sampling

Model building

Thresholding
Inlier counting




```
function RANSAC_line

a = 1;
b = 1;

nPoints = 100;
x = rand(nPoints,1)-0.5;
y = a*x + b + 0.2*randn(nPoints,1);

nRansacIter = 500;
threshold = 0.1;
max_nInliers = 0;
for i = 1 : nRansacIter
   r = randperm(nPoints);
   s1 = [x(r(1)); y(r(1)); 1];
   s2 = [x(r(2)); y(r(2)); 1];

   l = GetLineFromTwoPoints(s1, s2);

   nInliers = 0;
   for j = 1 : nPoints
      e = abs(l' * [x(j); y(j); 1])/norm(l(1:2));
      if (e < threshold)
         nInliers = nInliers + 1;
      end
   end
    if (nInliers > max_nInliers)
      max_nInliers = nInliers;
      L = l;
   end
end
```

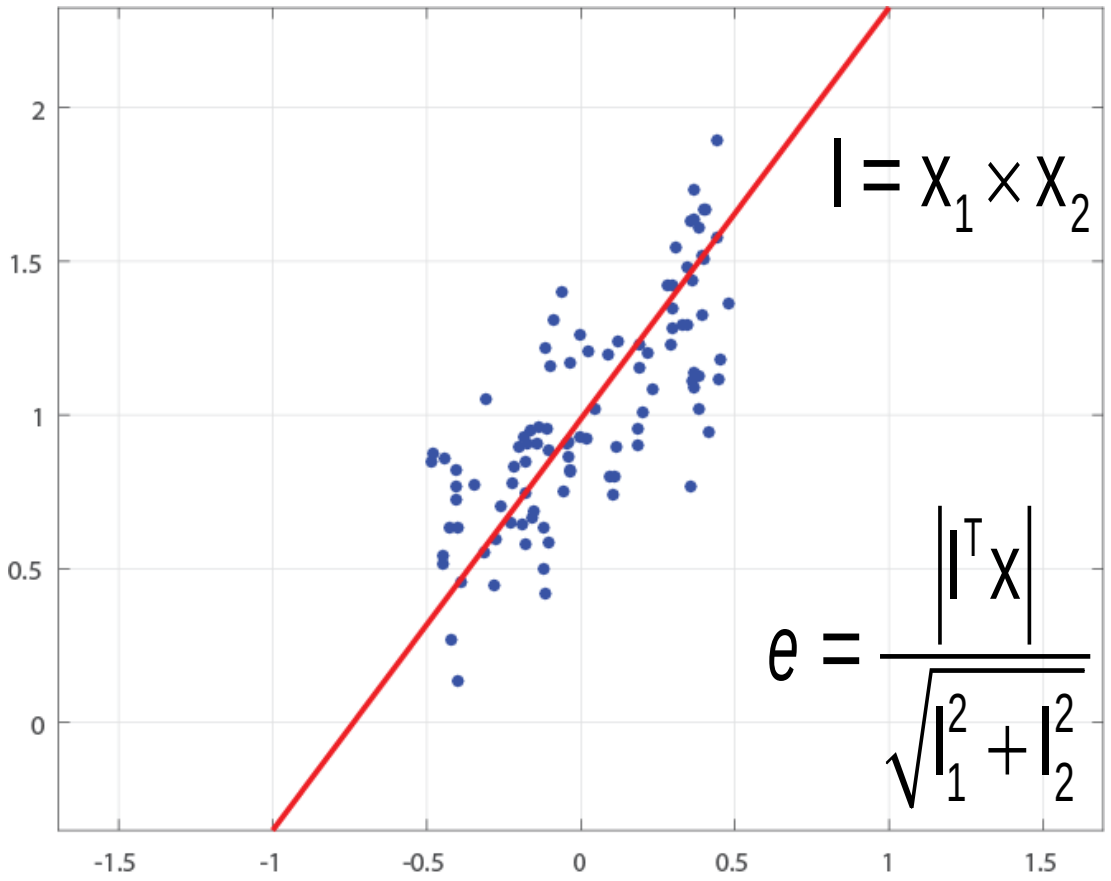
Generating samples

Random sampling

Model building

Thresholding
Inlier counting

Model updating




```
function RANSAC_line

a = 1;
b = 1;

nPoints = 100;
x = rand(nPoints,1)-0.5;
y = a*x + b + 0.2*randn(nPoints,1);

nRansacIter = 500;
threshold = 0.1;
max_nInliers = 0;
for i = 1 : nRansacIter
   r = randperm(nPoints);
   s1 = [x(r(1)); y(r(1)); 1];
   s2 = [x(r(2)); y(r(2)); 1];

   l = GetLineFromTwoPoints(s1, s2);

   nInliers = 0;
   for j = 1 : nPoints
      e = abs(l' * [x(j); y(j); 1])/norm(l(1:2));
      if (e < threshold)
         nInliers = nInliers + 1;
      end
   end
    if (nInliers > max_nInliers)
      max_nInliers = nInliers;
      L = l;
   end
end
```

Generating samples

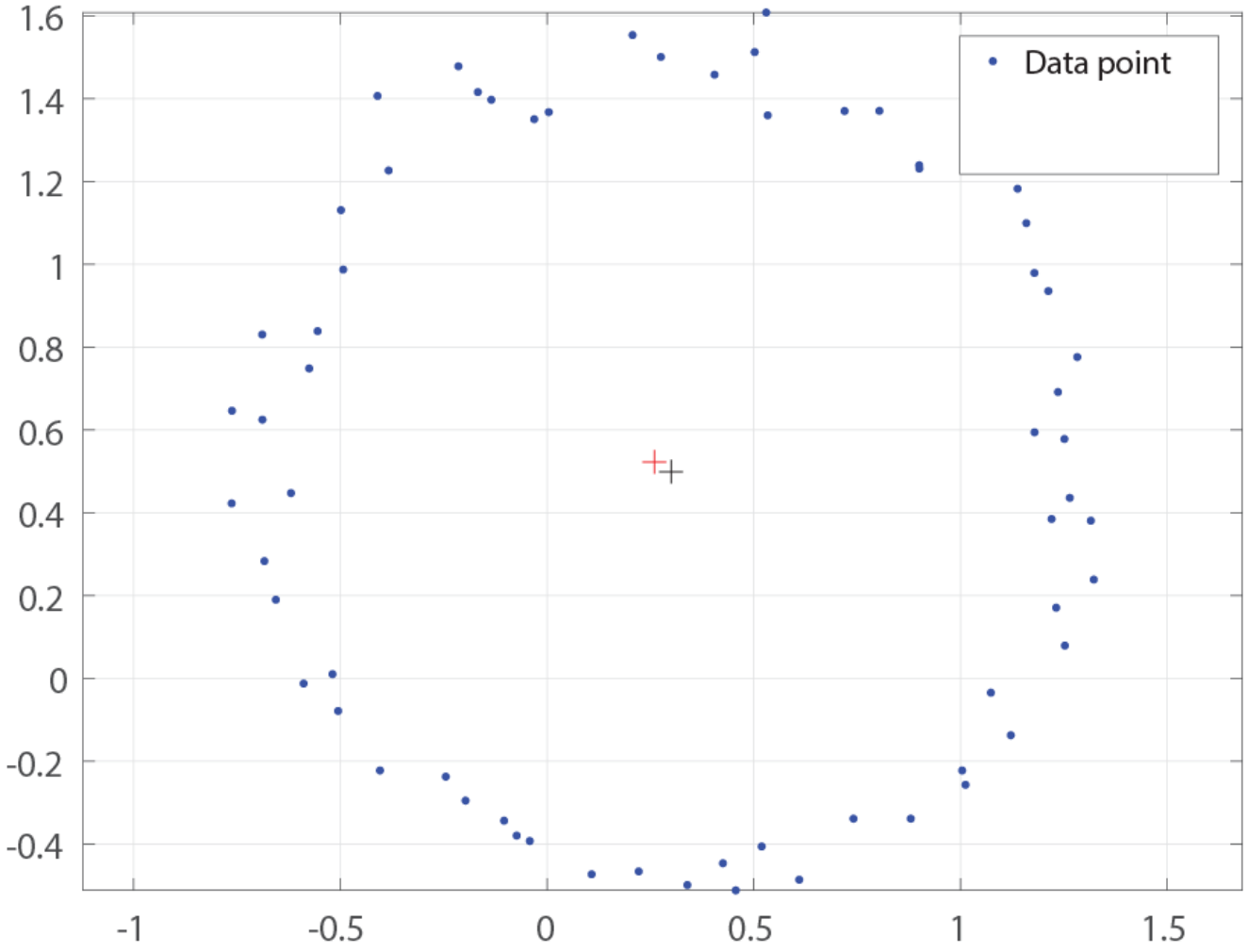
Random sampling

Model building

Thresholding
Inlier counting

Model updating

Recall: Circle Fitting (Ax=b)



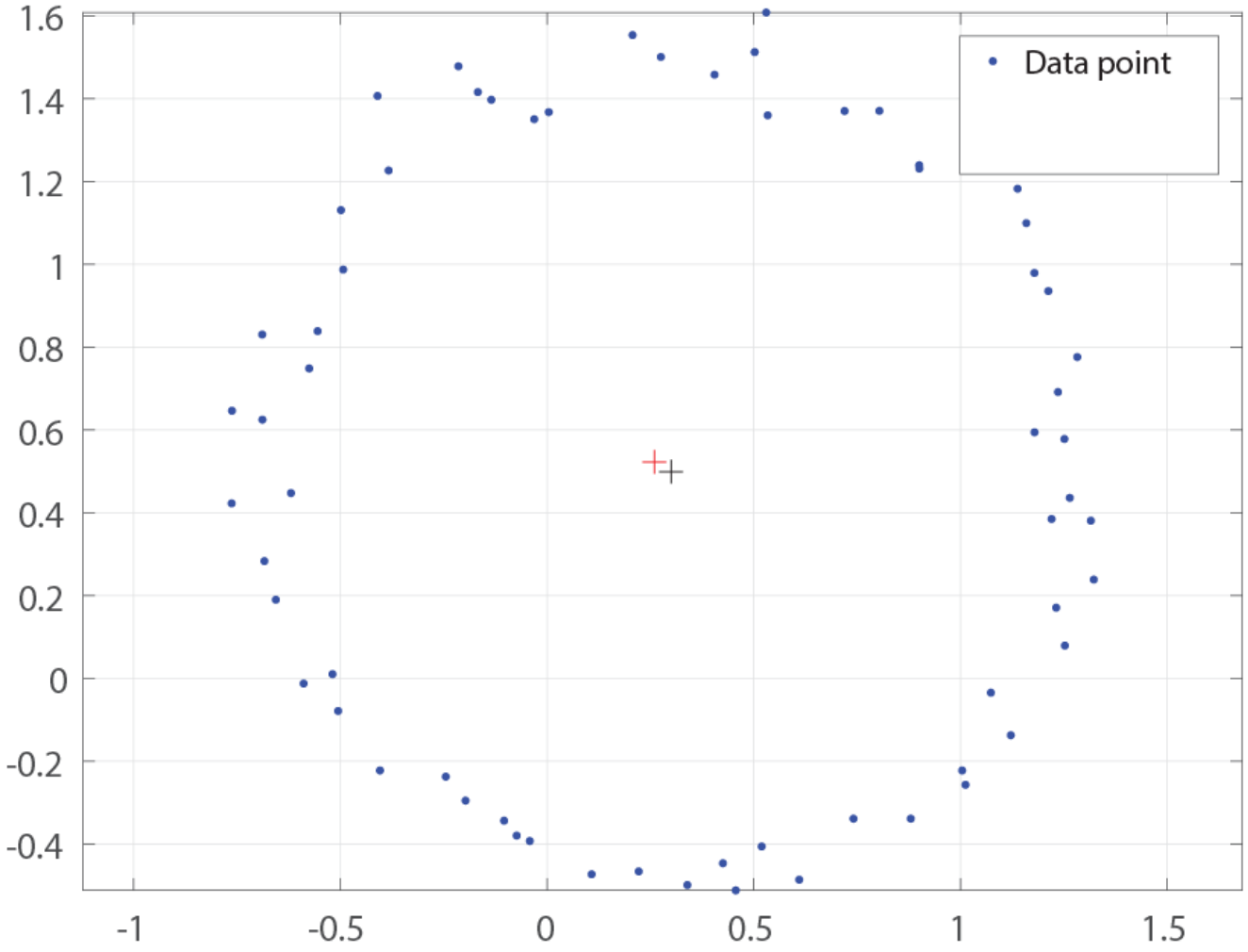
$$(x_1 - c_x)^2 + (y_1 - c_y)^2 = r^2$$

$$\vdots$$

$$(x_n - c_x)^2 + (y_n - c_y)^2 = r^2$$

Unkowns: c_x, c_y, r

Recall: Circle Fitting (Ax=b)

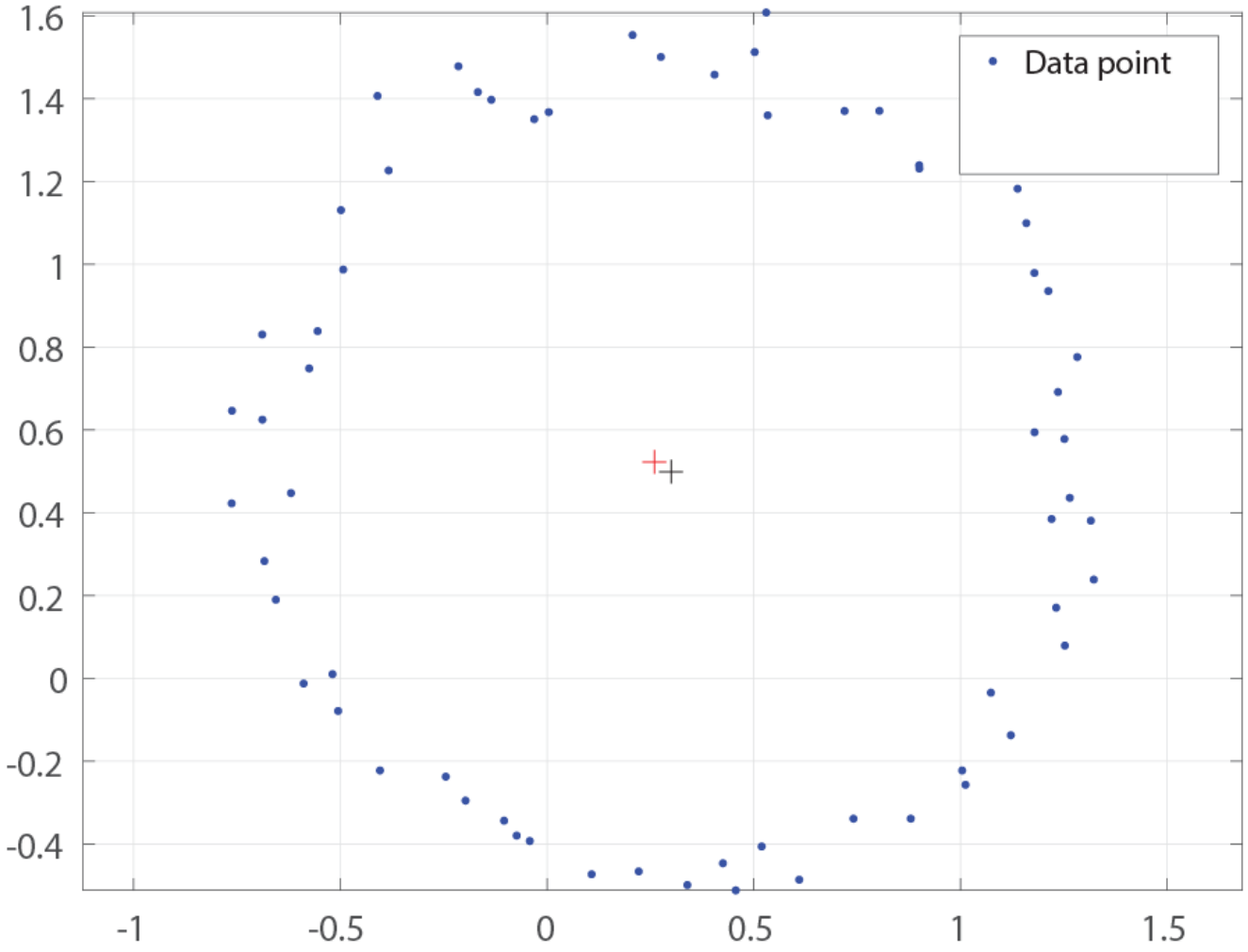


$$x_1^2 - 2x_1 c_x + c_x^2 + y_1^2 - 2y_1 c_y + c_y^2 = r^2$$

$$\vdots$$

$$x_n^2 - 2x_n c_x + c_x^2 + y_n^2 - 2y_n c_y + c_y^2 = r^2$$

Recall: Circle Fitting (Ax=b)



$$x_1^2 - 2x_1 c_x + c_x^2 + y_1^2 - 2y_1 c_y + c_y^2 = r^2$$

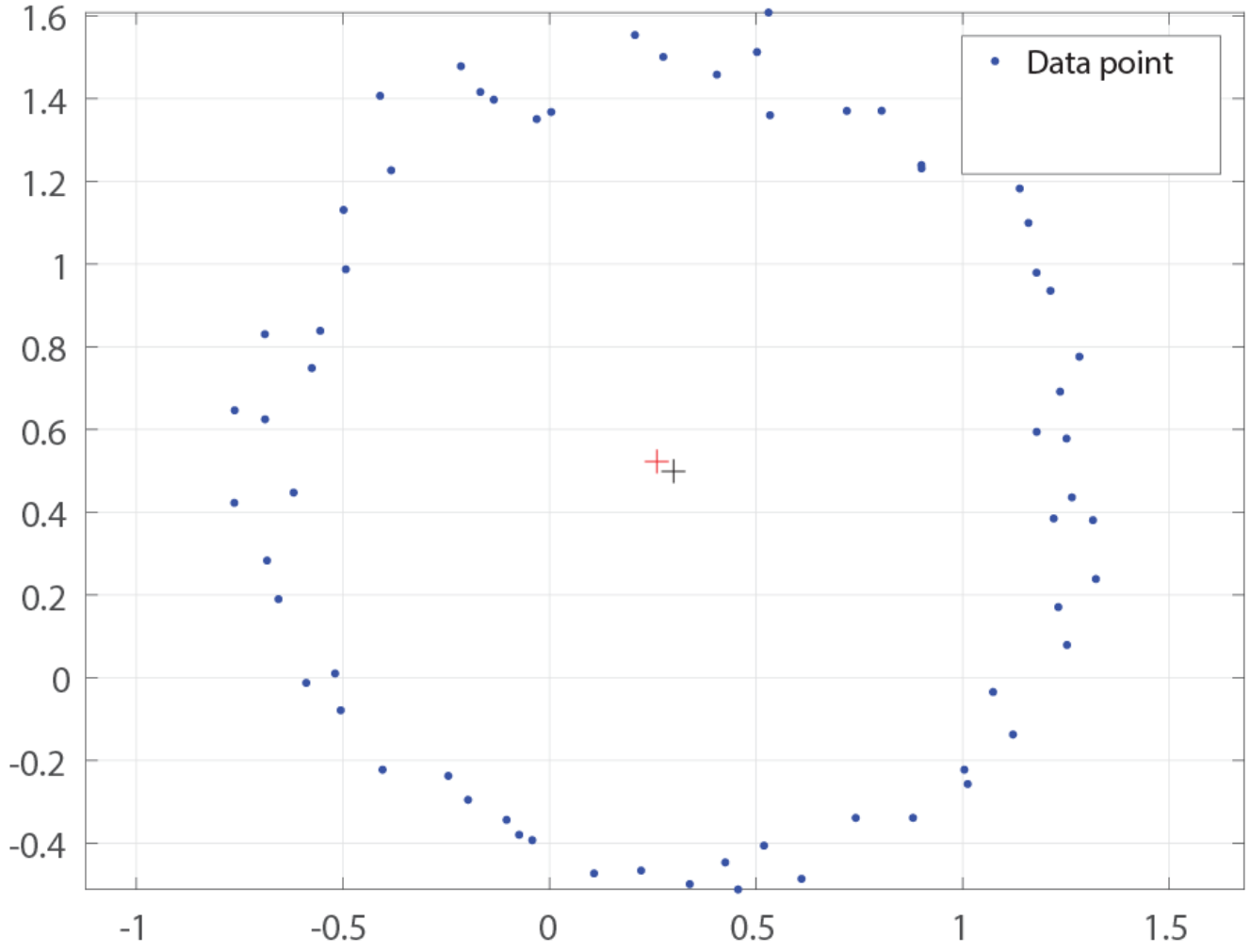
$$\vdots$$

$$x_n^2 - 2x_n c_x + c_x^2 + y_n^2 - 2y_n c_y + c_y^2 = r^2$$

$$\downarrow$$

$$x_i^2 - x_1^2 - 2c_x\left(x_i - x_1\right) + y_i^2 - y_1^2 - 2\left(y_i - y_1\right)c_y = 0$$

Recall: Circle Fitting (Ax=b)



$$x_1^2 - 2x_1 c_x + c_x^2 + y_1^2 - 2y_1 c_y + c_y^2 = r^2$$

$$\vdots$$

$$x_n^2 - 2x_n c_x + c_x^2 + y_n^2 - 2y_n c_y + c_y^2 = r^2$$

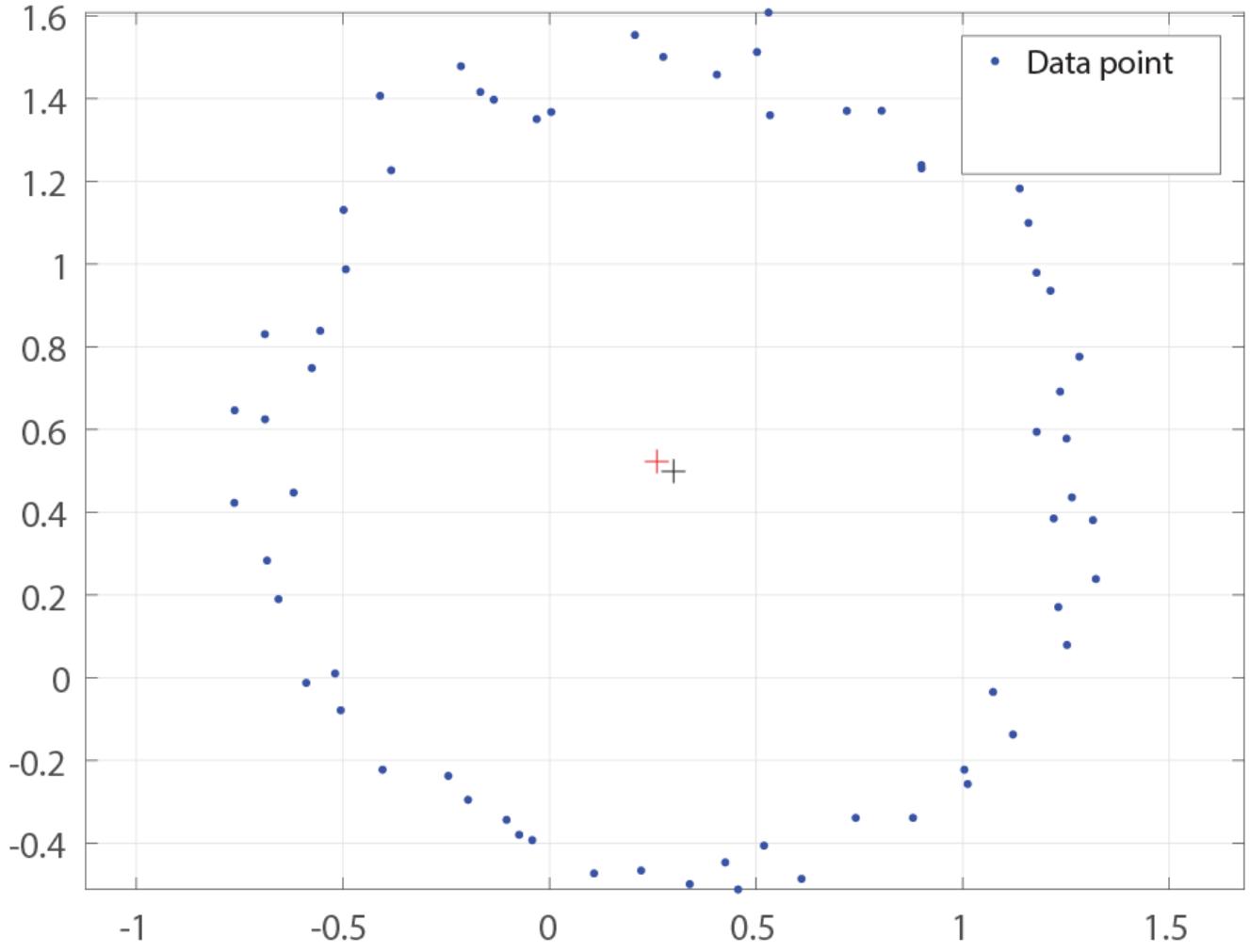
$$\downarrow$$

$$x_i^2 - x_1^2 - 2c_x\left(x_i - x_1\right) + y_i^2 - y_1^2 - 2\left(y_i - y_1\right)c_y = 0$$

$$\downarrow$$

$$\begin{bmatrix} 2\left(x_2 - x_1\right) & 2\left(y_2 - y_1\right) \\ \vdots & \vdots \\ 2\left(x_n - x_1\right) & 2\left(y_n - y_1\right) \end{bmatrix} \begin{bmatrix} c_x \\ c_y \end{bmatrix} = \begin{bmatrix} x_2^2 - x_1^2 + y_2^2 - y_1^2 \\ \\ x_n^2 - x_1^2 + y_n^2 - y_1^2 \end{bmatrix}$$

Circle Fitting (Ax=b)



$$x_1^2 - 2x_1 c_x + c_x^2 + y_1^2 - 2y_1 c_y + c_y^2 = r^2$$

$$\vdots$$

$$x_n^2 - 2x_n c_x + c_x^2 + y_n^2 - 2y_n c_y + c_y^2 = r^2$$

$$\downarrow$$

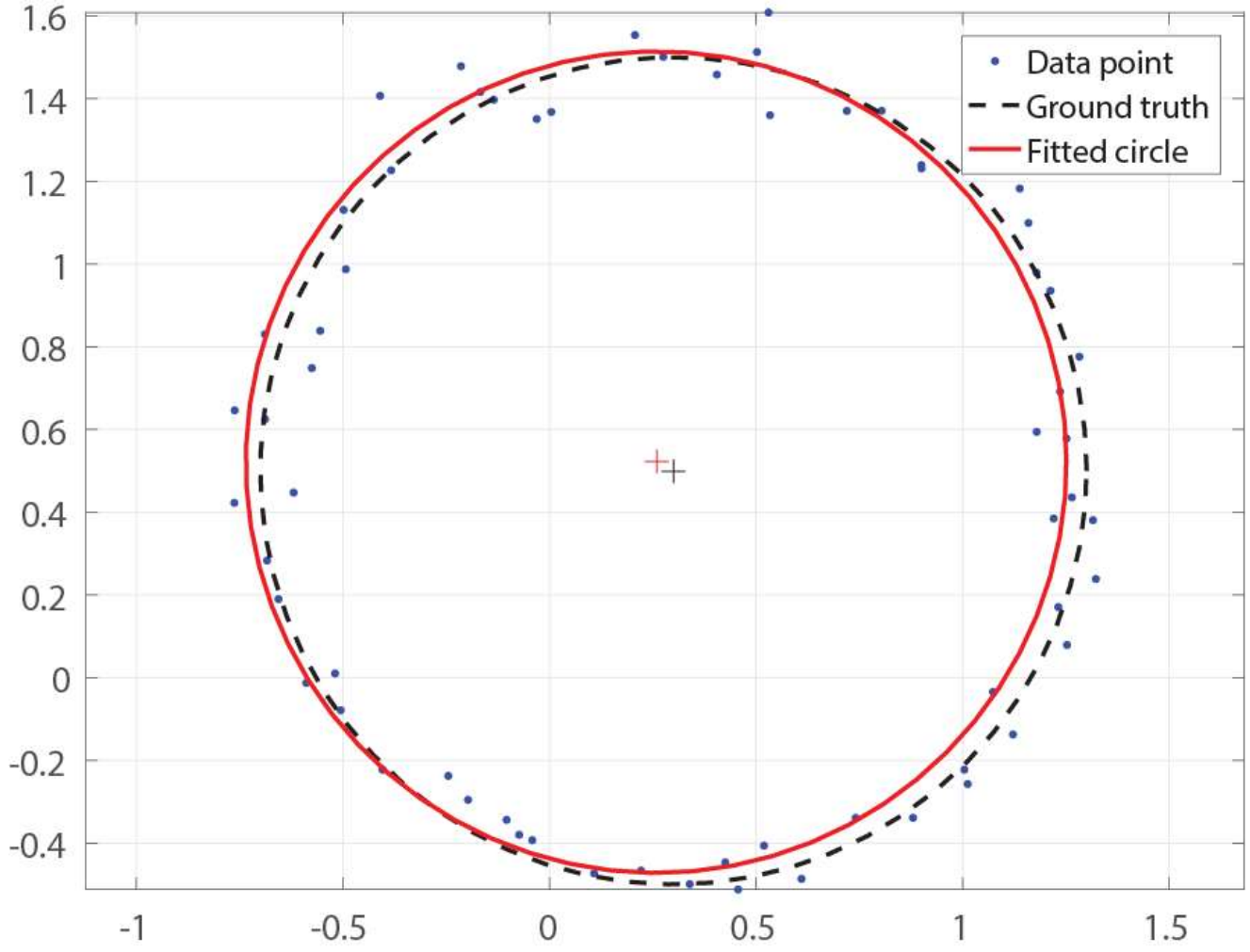
$$x_i^2 - x_1^2 - 2c_x(x_i - x_1) + y_i^2 - y_1^2 - 2(y_i - y_1)c_y = 0$$

$$\downarrow$$

$$\begin{bmatrix} 2(x_2 - x_1) & 2(y_2 - y_1) \\ \vdots & \vdots \\ 2(x_n - x_1) & 2(y_n - y_1) \end{bmatrix} \begin{bmatrix} c_x \\ c_y \end{bmatrix} = \begin{bmatrix} x_2^2 - x_1^2 + y_2^2 - y_1^2 \\ \vdots \\ x_n^2 - x_1^2 + y_n^2 - y_1^2 \end{bmatrix}$$

$$A \qquad x \qquad b$$

Recall: Circle Fitting (Ax=b)



$$x_1^2 - 2x_1 c_x + c_x^2 + y_1^2 - 2y_1 c_y + c_y^2 = r^2$$

$$\vdots$$

$$x_n^2 - 2x_n c_x + c_x^2 + y_n^2 - 2y_n c_y + c_y^2 = r^2$$

$$\downarrow$$

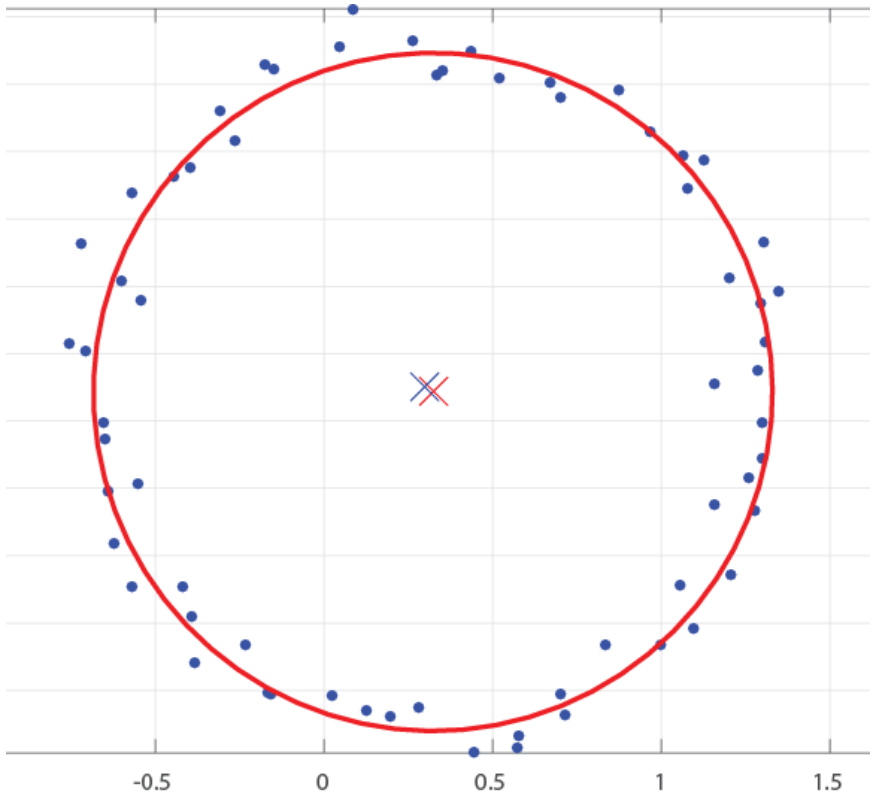
$$x_i^2 - x_1^2 - 2c_x(x_i - x_1) + y_i^2 - y_1^2 - 2(y_i - y_1)c_y = 0$$

$$\downarrow$$

$$\begin{bmatrix} 2(x_2 - x_1) & 2(y_2 - y_1) \\ \vdots & \vdots \\ 2(x_n - x_1) & 2(y_n - y_1) \end{bmatrix} \begin{bmatrix} c_x \\ c_y \end{bmatrix} = \begin{bmatrix} x_2^2 - x_1^2 + y_2^2 - y_1^2 \\ \vdots \\ x_n^2 - x_1^2 + y_n^2 - y_1^2 \end{bmatrix}$$

$$A \qquad x \qquad b$$

Circle Fitting RANSAC



$$\begin{bmatrix} 2(x_2 - x_1) & 2(y_2 - y_1) \\ \vdots & \vdots \\ 2(x_n - x_1) & 2(y_n - y_1) \end{bmatrix} \begin{bmatrix} c_x \\ c_y \end{bmatrix} = \begin{bmatrix} x_2^2 - x_1^2 + y_2^2 - y_1^2 \\ \\ x_n^2 - x_1^2 + y_n^2 - y_1^2 \end{bmatrix}$$

Download RANSAC_circle.m

```
function RANSAC_circle

cx = 0.3;
cy = 0.5;
r = 1;

theta = 0:0.1:2*pi+0.1;
theta = theta';
x = cos(theta) + cx + 0.05*randn(size(theta));
y = sin(theta) + cy + 0.05*randn(size(theta));

figure(1)
clf;
plot(x,y, 'b.');
axis equal
grid on

nRansacIter = 500;
threshold = 0.1;
max_nInliers = 0;
%% RANSAC circle
% U: Estimated center of circle
% R: Radius

%%
hold on
plot(R*cos(theta)+U(1), R*sin(theta)+U(2), 'r-');
hold on
plot(U(1), U(2), 'rx');
hold on
plot(cx, cy, 'bx');
```

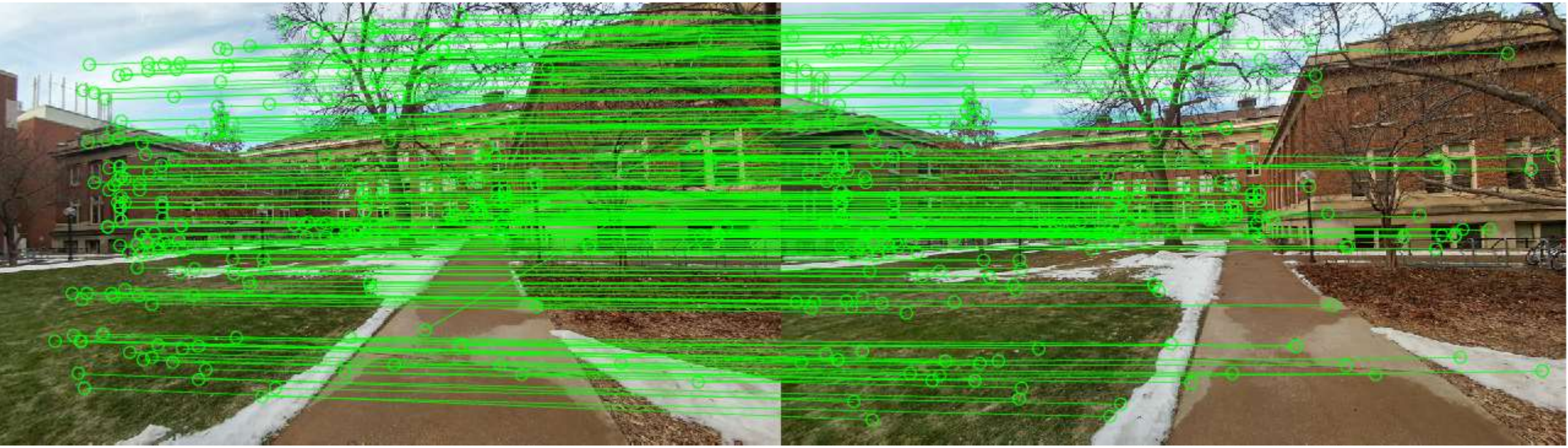
Fill out this part

HW #4 RANSAC Fundamental matrix

Algorithm 1 GetInliersRANSAC

$n \leftarrow 0$
for $i = 1 : M$ **do**
    Choose 8 correspondences, $\mathbf{u}_r$ and $\mathbf{v}_r$, randomly from $\mathbf{u}$ and $\mathbf{v}$.
    $\mathbf{F}_r$ = `ComputeFundamentalMatrix(`$\mathbf{u}_r$`,` $\mathbf{v}_r$`)`
    Compute the number of inliers, n_r, with respect to `F`.
    if $n_r > n$ **then**
        $n \leftarrow n_r$
        $\mathbf{F} = \mathbf{F}_r$
    end if
end for

Fundamental Matrix Computation via RANSAC



Fundamental Matrix Computation via RANSAC

$$\begin{bmatrix} u_1^x v_1^x & u_1^y v_1^x & v_1^x & u_1^x v_1^y & u_1^y v_1^y & v_1^y & u_1^x & u_1^y & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_m^x v_m^x & u_m^y v_m^x & v_m^x & u_m^x v_m^y & u_m^y v_m^y & v_m^y & u_m^x & u_m^y & 1 \end{bmatrix} A \quad \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} X = 0$$

Fundamental Matrix Computation via RANSAC

Epipolar line: $l_u = Fu$

Distance: $$d = \frac{\left|au_x + bu_y + c\right|}{\sqrt{a^2 + b^2}} = \frac{\left|Fu\right|}{\sqrt{(F_{1,:}u)^2 + (F_{2,:}u)^2}}$$

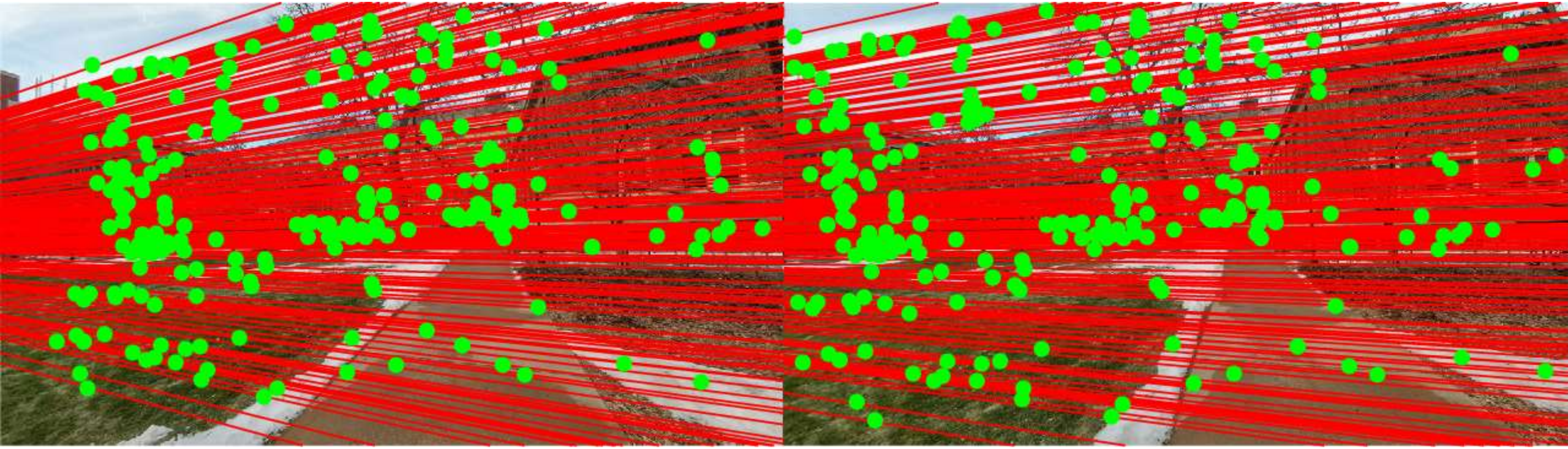
Fundamental Matrix Computation via RANSAC

# of inliers: 65 out of 260

Fundamental Matrix Computation via RANSAC

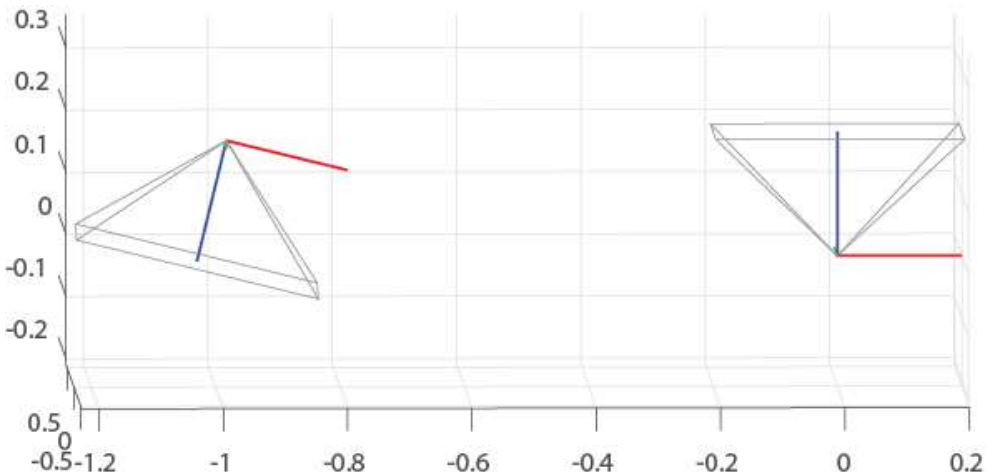
# of inliers: 65 out of 260

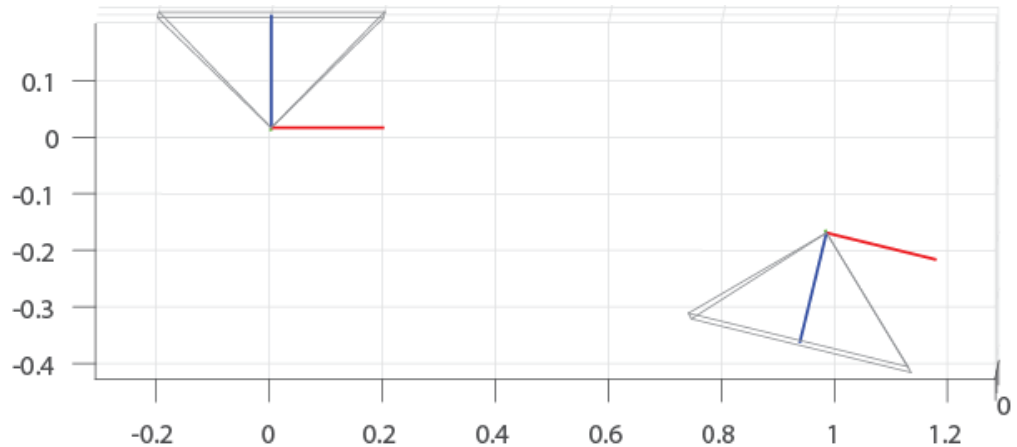
Fundamental Matrix Computation via RANSAC

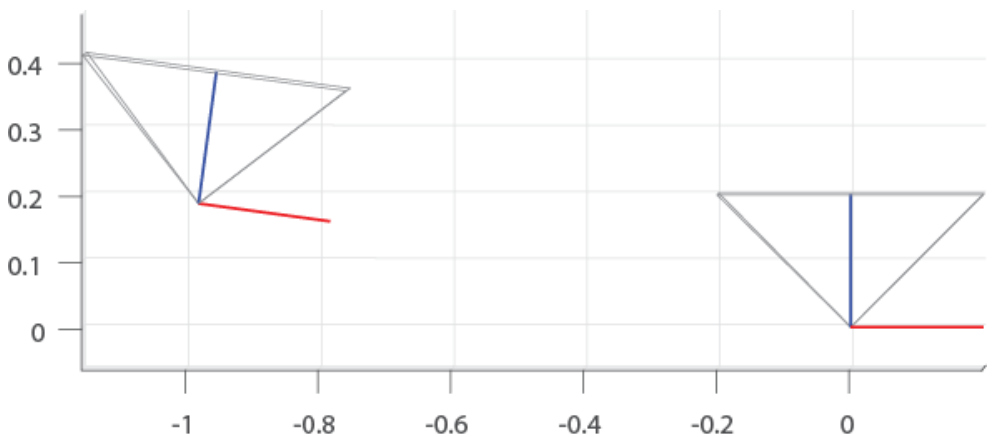


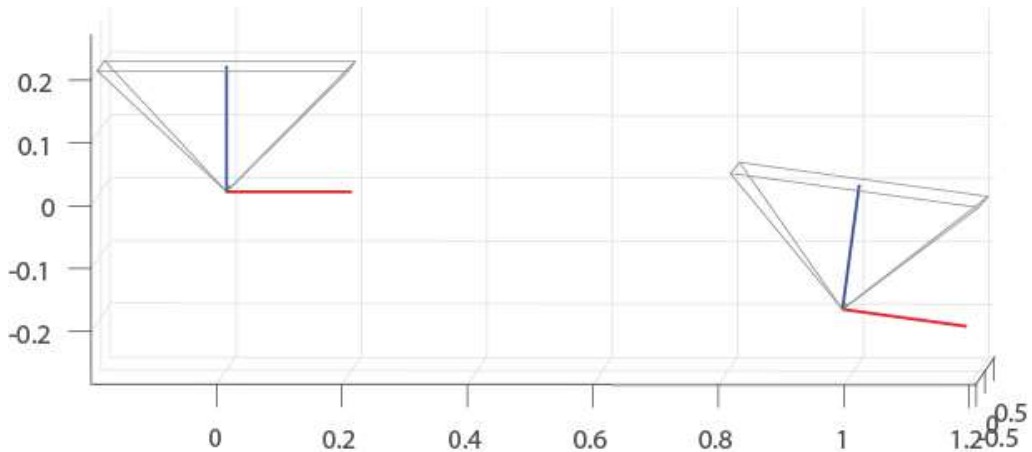
# of inliers: 186 out of 260

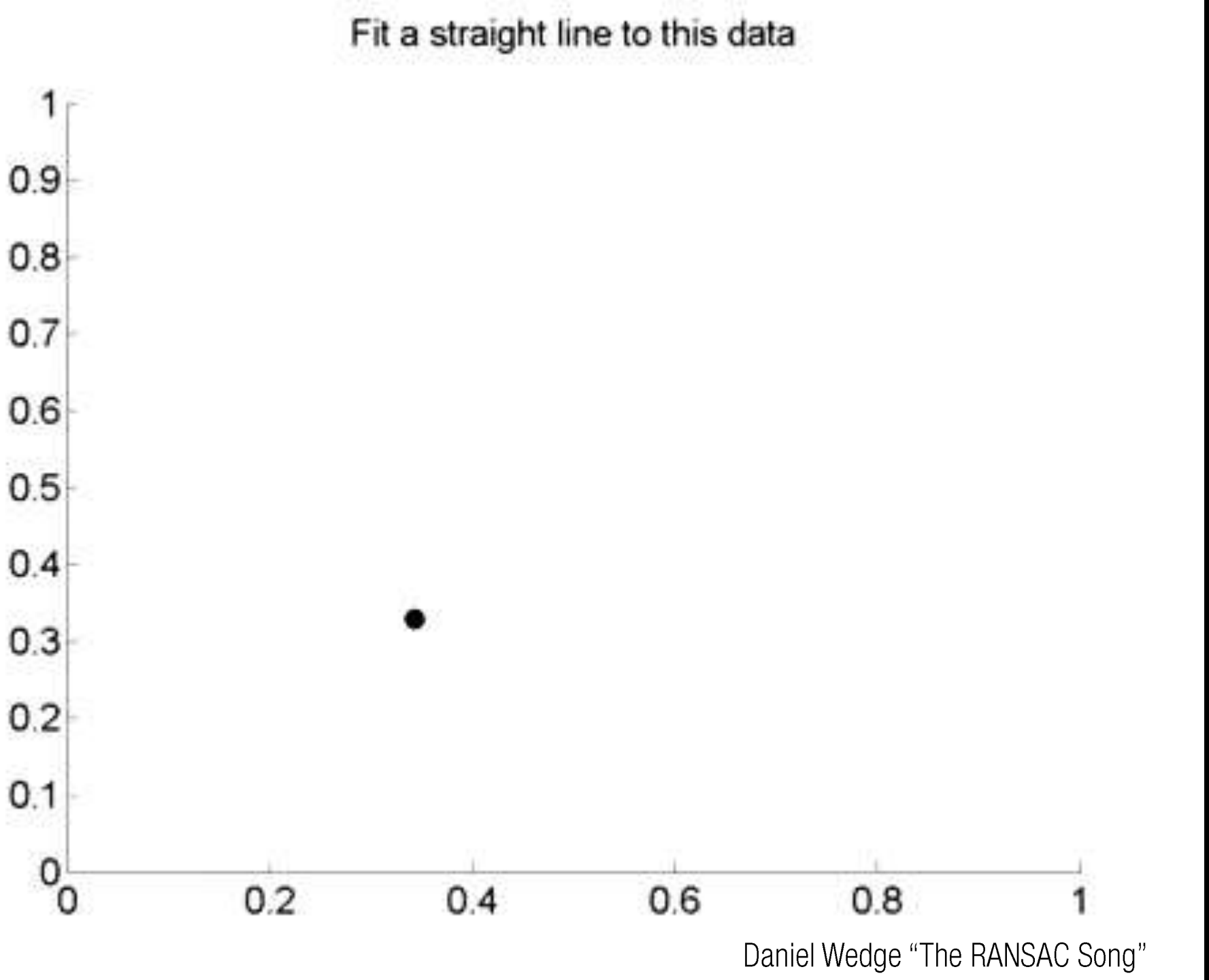
Four Camera Pose Config. From Essential Matrix











Daniel Wedge “The RANSAC Song”