Reconstruction Steps

1. Identify a number of (at least 8) point correspondances.
2. Estimate the fundamental matrix using the normalised 8-point algorithm.
3. Determine the external camera parameters (rotation and translation from one camera to the other)
    a. Calculate the essential matrix from the fundamental matrix and the camera calibration matrices.
    b. Extract the rotation and translation components from the essential matrix.

4. Determine 3D point locations.

Determining Extrinsic Camera Parameters

- Want to determine rotation and translation from one camera to the other.
- We know the camera matrices have the form

$$M_1 = C_1\begin{bmatrix} I & \vec{0} \end{bmatrix}$$
$$M_2 = C_2\begin{bmatrix} R & \vec{t} \end{bmatrix}$$

Why can we just use this as the external parameters for camera M1?
Because we are only interested in the *relative* position of the two cameras.

- First we undo the Intrinsic camera distortions by defining new *normalized* cameras

$$M_1^{norm} = C_1^{-1} M_1 \quad \text{and} \quad M_2^{norm} = C_2^{-1} M_2$$

Determining Extrinsic Camera Parameters

- The *normalized* cameras contain unknown parameters

$$M_1^{norm} = C_1^{-1} M_1 \quad \Rightarrow \quad M_1^{norm} = \begin{bmatrix} I & \vec{0} \end{bmatrix}$$

$$M_2^{norm} = C_2^{-1} M_2 \quad \Rightarrow \quad M_2^{norm} = \begin{bmatrix} R & \vec{t} \end{bmatrix}$$

- However, those parameters can be extracted from the Fundamental matrix

$$F = C_2^{-t} E C_1^{-1}$$

$$E = C_2^{t} F C_1$$

$$E = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} R = \vec{t}_{\times} R$$

Extract t and R from the Essential Matrix

How do we recover *t* and *R*? Answer: SVD of $\mathcal{E}$

$$\mathcal{E} = USV^t$$

S diagonal

U,V orthogonal and $\det() = 1$ (rotation)

$$R = UWV^t \quad \text{or} \quad R = UW^tV^t \qquad \vec{t} = u_3 \quad \text{or} \quad \vec{t} = -u_3$$

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

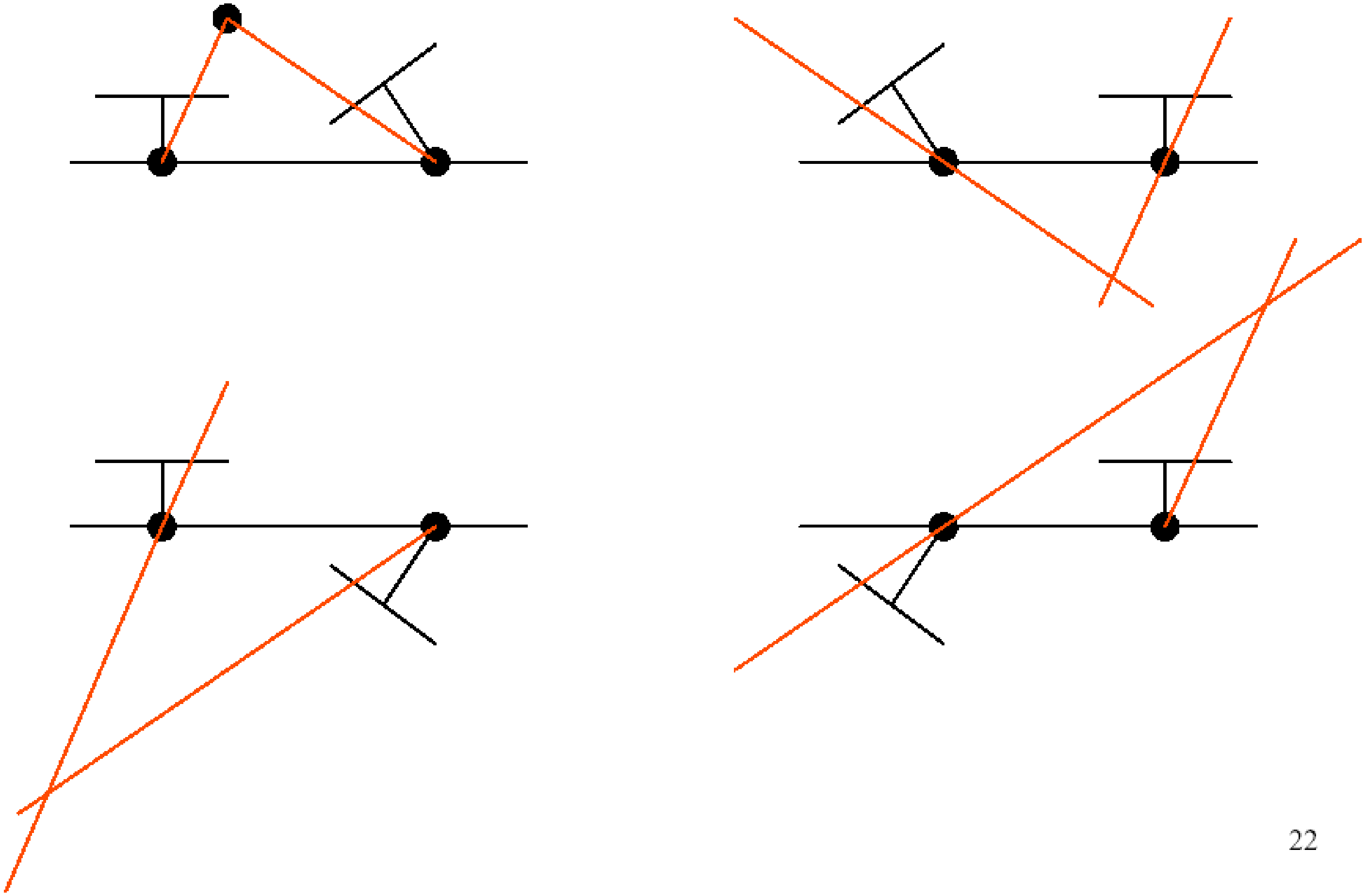
Reconstruction Ambiguity

So we have 4 possible combinations of translations and rotations giving 4 possibilities for $\boldsymbol{M}_2^{norm} = [\boldsymbol{R} \mid \boldsymbol{t}]$

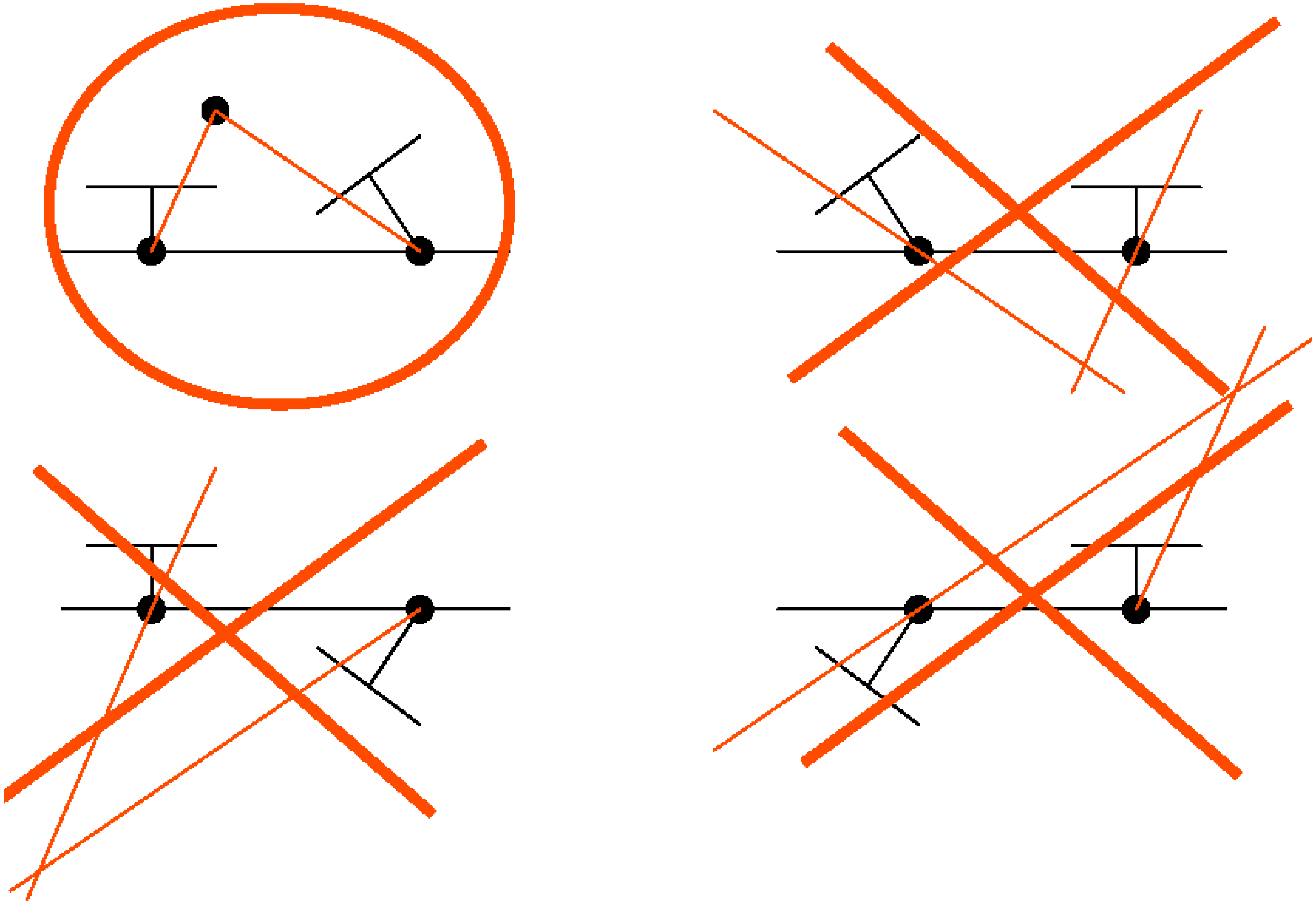
1. $\boldsymbol{M}_2^{norm} = [\boldsymbol{U}\boldsymbol{W}^t\boldsymbol{V}^t \mid \boldsymbol{t}]$
2. $\boldsymbol{M}_2^{norm} = [\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^t \mid \boldsymbol{t}]$
3. $\boldsymbol{M}_2^{norm} = [\boldsymbol{U}\boldsymbol{W}^t\boldsymbol{V}^t \mid -\boldsymbol{t}]$
4. $\boldsymbol{M}_2^{norm} = [\boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^t \mid -\boldsymbol{t}]$

Which one is right?

- We can determine which of these is correct by looking at their geometric interpretation.



Both Cameras must be facing the same direction



Which one is right?

- The correct pair will have our data points in front of both cameras.
- How do we choose the correct pair?
- Procedure:
  - Take a test point from data
  - Backproject to find 3D location
  - Determine the depth of 3D point in both cameras
  - Choose the camera pair that has a positive depth for both cameras.

How do we backproject?

$x'_{measured} = C_2 x'$ $\qquad$ $x_{measured} = C_1 x$

Knowing C_i allows us to determine the undistorted image points:

$x' = C_2^{-1} x'_{measured}$ $\qquad$ $x = C_1^{-1} x_{measured}$

Recalling the projection equations allows to relate the world point and the image points.

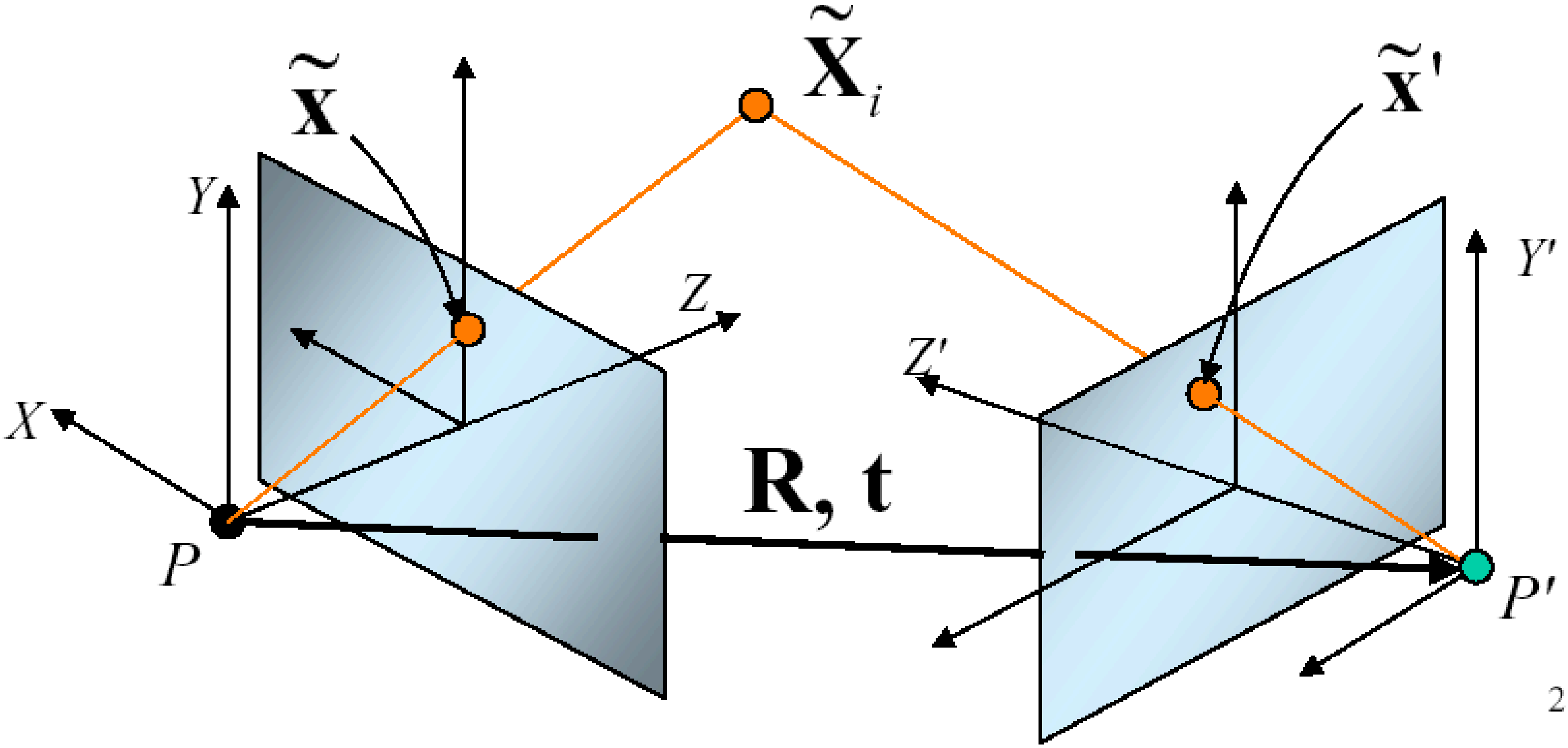
$$x' = C_2^{-1} x'_{measured} \qquad x = C_1^{-1} x_{measured}$$

$$z' x' = C_2^{-1} C_2 M_2^{norm} X \qquad zx = C_1^{-1} C_1 \left[\mathbf{I} \,|\, 0 \right] X$$

$$z' x' = M_2^{norm} X \qquad zx = \left[\mathbf{I} \,|\, 0 \right] X$$

Backprojection to 3D

We now know *x*, *x'*, *R*, and *t*
Need *X*



Solving…

$$z x_i = M^{norm} X_i$$

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} m_1^t \\ m_2^t \\ m_3^t \end{bmatrix} X_i \Rightarrow \begin{cases} z u_i = m_1^t \cdot X_i \\ z v_i = m_2^t \cdot X_i \\ z = m_3^t \cdot X_i \end{cases} \Rightarrow \begin{cases} (m_3^t \cdot X_i) u_i = m_1^t \cdot X_i \\ (m_3^t \cdot X_i) v_i = m_2^t \cdot X_i \end{cases}$$

$$\Rightarrow \begin{cases} (m_3^t \cdot X_i) u_i - m_1^t \cdot X_i = 0 \\ (m_3^t \cdot X_i) v_i - m_2^t \cdot X_i = 0 \end{cases}$$

$$\Rightarrow \begin{bmatrix} u_i (m_3^t) - m_1^t \\ v_i (m_3^t) - m_2^t \end{bmatrix} X_i = 0$$

Solving…

- Similarly for the other camera

$$\begin{bmatrix} u'_i({}^2m_3^t) - {}^2m_1^t \\ v'_i({}^2m_3^t) - {}^2m_2^t \end{bmatrix} X_i = 0$$

Where ${}^2m^t_i$ denotes the *i*th row of the second camera's normalized projection matrix.

Combining 1 & 2:

$$\begin{bmatrix} u_i(m_3^t) - m_1^t \\ v_i(m_3^t) - m_2^t \\ u'_i({}^2m_3^t) - {}^2m_1^t \\ v'_i({}^2m_3^t) - {}^2m_2^t \end{bmatrix} X_i = 0$$

$$AX_i = 0$$

It has a solvable form! Solve using minimum eigenvalue-Eigenvector approach e.g. svd (or null)

Finishing up

- Now we have the 3D point $\tilde{\mathbf{X}}_i$
- Determine the location of this point for all 4 possible camera configurations
- Next determine the depths of these points in in each camera.

A little more Linear Algebra

- Given 2 simultaneous equations we can write them in matrix form

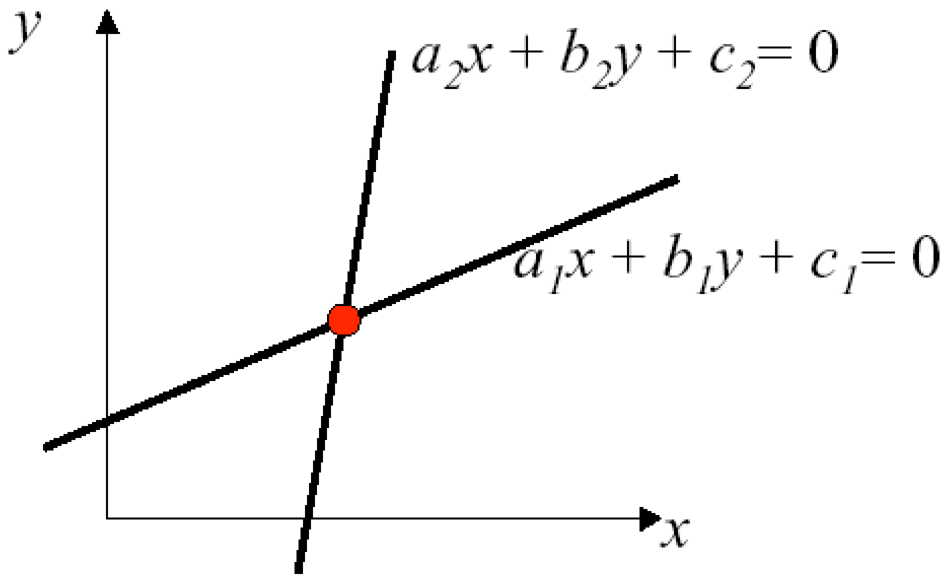
$$\left.\begin{aligned} a_1x + b_1y + c_1 = 0 \\ a_2x + b_2y + c_2 = 0 \end{aligned}\right\} \quad \Leftrightarrow \quad \mathbf{Ax} + \mathbf{c} = \mathbf{0}$$

where $\mathbf{A} = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}, \ \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}, \ \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \ \mathbf{0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

and the solution is $\mathbf{x} = -\mathbf{A}^{-1}\mathbf{c}$

Interpretation

- We are asking for a single (x,y) point that satisfies both line equations.
- Graphically this amounts to finding the point that lies on both lines.



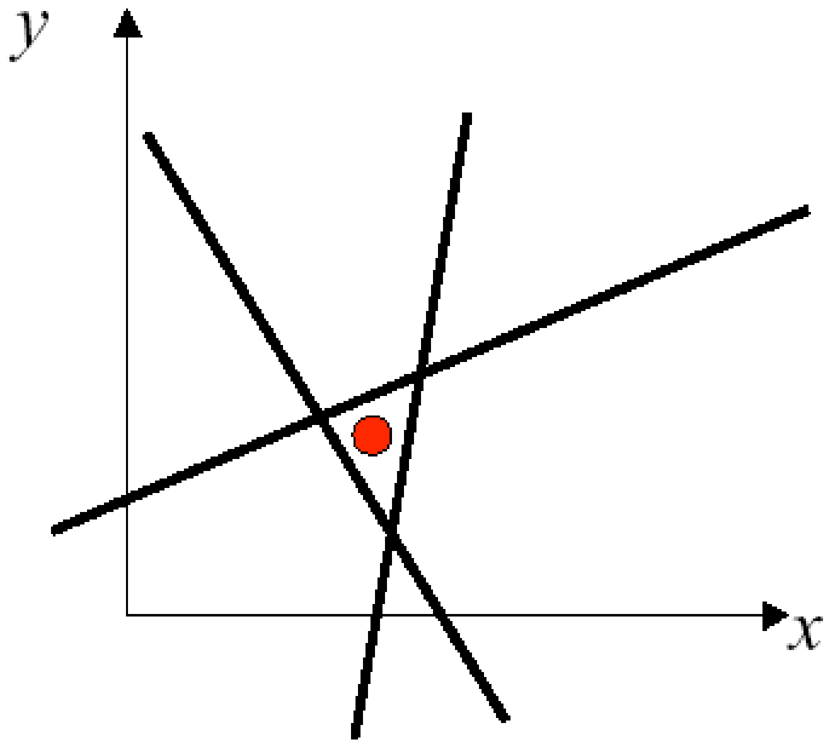
- Given *n* simultaneous equations in 2D

$$\left.\begin{aligned} a_1x + b_1y + c_1 &= 0 \\ a_2x + b_2y + c_2 &= 0 \\ &\dots \\ a_nx + b_ny + c_n &= 0 \end{aligned}\right\} \Leftrightarrow \quad \mathbf{Ax} + \mathbf{c} = \mathbf{0}$$

$$\mathbf{A} = \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_n & b_n \end{bmatrix},\ \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix},\ \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix},\ \mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- For $n>2$ this is an over-constrained system. $\mathbf{A}^{-1}$ does not exist.
- There need not be an exact solution.
- We want to find the 'best' solution.

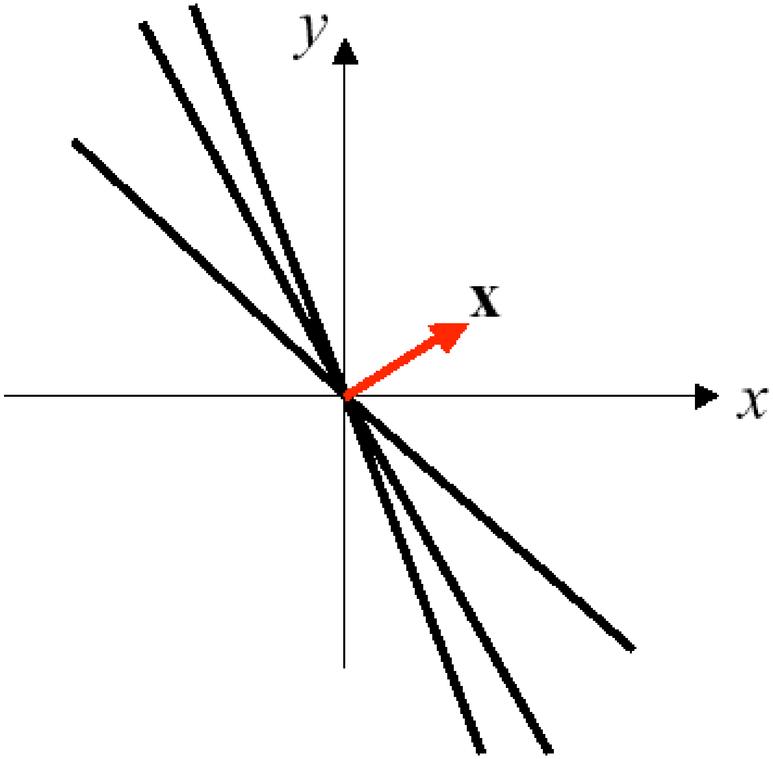
- Graphically we want to find the point that is closest to all n lines at once.



- Note that “closeness” means Euclidean distance for (unweighted) least squares solutions.

What about when c = 0?   *Case*: Ax = 0, x≠0

- Solve **Ax** = **0**, for non-zero **x**,
- Find the *direction* of **x** that *minimises* **Ax**.
- In 2D this can be interpreted as finding the **x** that is *most perpendicular* to all *n* lines (ie. most perpendicular to the rows of **A**).



Why does Ax = 0 represent a normal constraint?

The equation of a (2D) line can be written

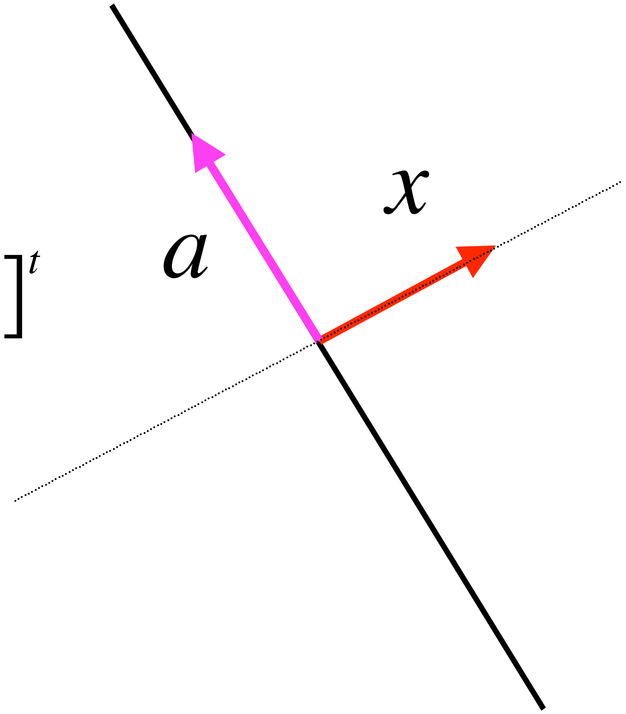
$$y = mx + b$$

$$\square \quad ax + by = \square c$$

$$\square \quad ax + by + c = 0$$

$$\left[\vec{a}^t\vec{x}\right] = 0, \quad \vec{a} = \left[a \quad b \quad c\right]^t, \quad \vec{x} = \left[x \quad y \quad 1\right]^t$$

For dimension > 2, this is a *hyperplane*



Note x is only defined up to a scale factor, because

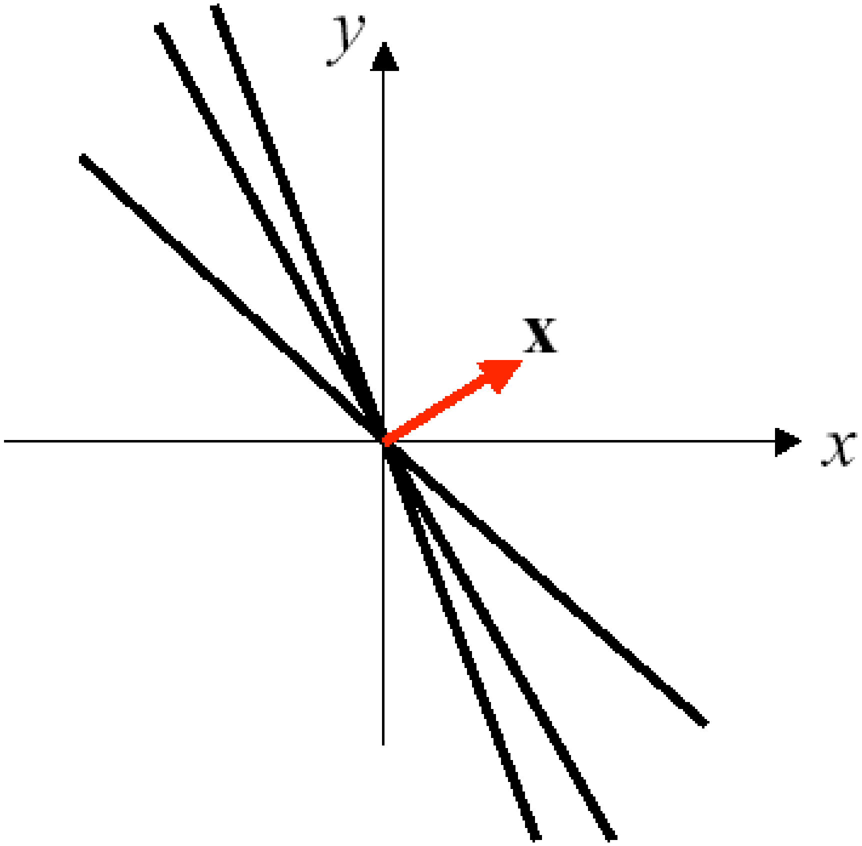
$$\vec{a}^t(\square\vec{x}) = 0$$

$$\square(\vec{a}^t\vec{x}) = 0$$

$$\vec{a}^t\vec{x} = 0$$

Solution

- Choose **x** to be the eigen vector associated with the smallest eigen value of $\mathbf{A}^T\mathbf{A}$. …Why is this?



- $\mathbf{x}$ can only be determined to a scale.
- So, choose $\mathbf{x}$ to be a unit vector,

$$||\mathbf{x}|| = 1,$$

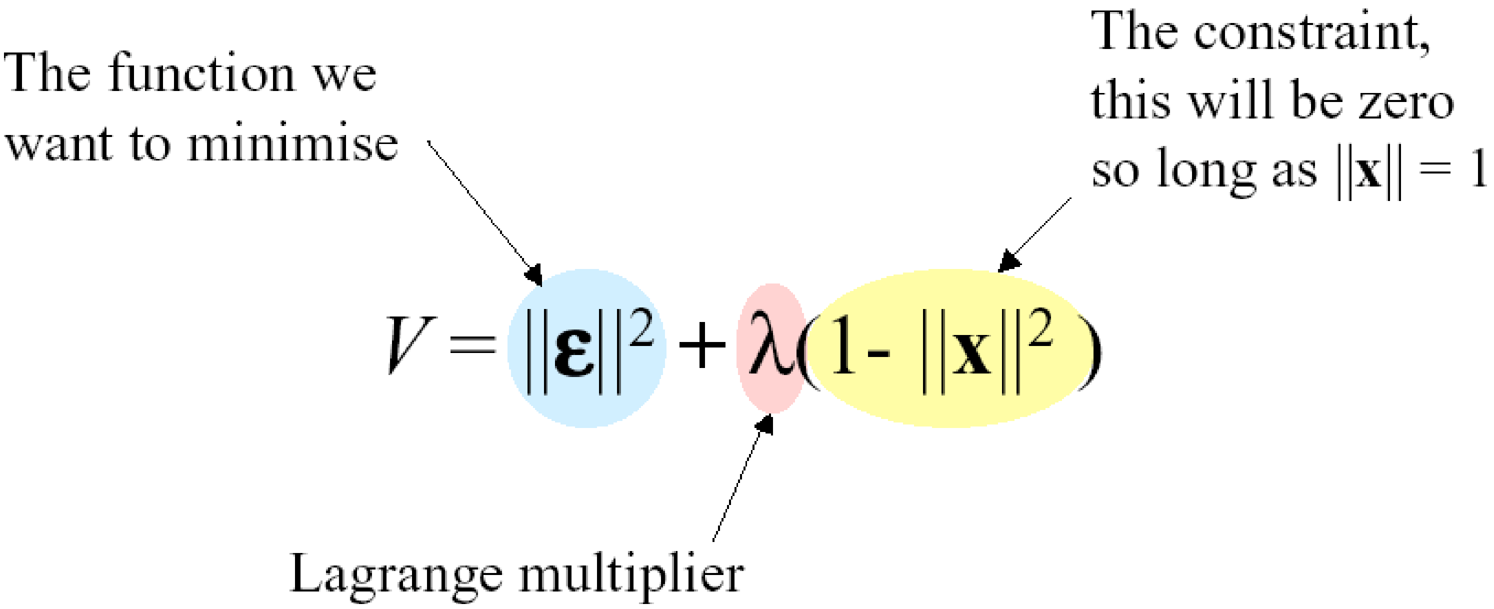
- Define $\boldsymbol{\varepsilon} = \mathbf{Ax}$,

- We want to find an $\mathbf{x}$ so that $||\boldsymbol{\varepsilon}||$ is as small as possible, and $||\mathbf{x}|| = 1$.

- We can achieve this by minimising the positive cost function $C = ||\boldsymbol{\varepsilon}||^2$ using the method of Lagrange Multipliers.

Enforce both constraints using Lagrangian multipliers

- Proceeding with method of Lagrange multipliers, define V



$$V = ||\boldsymbol{\varepsilon}||^2 + \lambda(1 - ||\mathbf{x}||^2)$$

The function we want to minimise

The constraint, this will be zero so long as $||\mathbf{x}|| = 1$

Lagrange multiplier

$$V = \|\boldsymbol{\varepsilon}\|^2 + \lambda(1 - \|\mathbf{x}\|^2)$$

V can be rewritten as

$$V = \mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} + \lambda(1\text{-}\mathbf{x}^{\mathrm{T}}\mathbf{x})$$

since

- $\|\boldsymbol{\varepsilon}\|^2 = \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon} = \mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x}$ and
- $\|\mathbf{x}\|^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x}$

- Find critical points of V, ie. where the derivative $\mathrm{d}V/\mathrm{d}\mathbf{x}$ is zero

$$V = \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} + \lambda(1-\mathbf{x}^T\mathbf{x})$$

$$\mathrm{d}V/\mathrm{d}\mathbf{x} = 2\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\lambda\mathbf{x} = 0$$

$$\Rightarrow \ \mathbf{A}^T\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

- This is the eigen equation!
- $\mathbf{x}$ must be an eigen vector of $\mathbf{A}^T\mathbf{A}$
  (so that $\mathrm{d}V/\mathrm{d}\mathbf{x} = 0$) this is a necessary, but not sufficient condition to minimise C.

- Which eigen vector to choose?

- Choose the eigen vector that minimises C.

- Let's substitute in for **x** an arbitrary unit eigen vector $\mathbf{e}_n$.

$$\begin{aligned} C &= \|\boldsymbol{\varepsilon}\|^2 \\ &= \mathbf{e}_n{}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{e}_n \\ &= \mathbf{e}_n{}^{\mathrm{T}}(\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{e}_n) \\ &= \mathbf{e}_n{}^{\mathrm{T}}\mathbf{e}_n\,\lambda \\ &= \lambda_n \end{aligned}$$

This is minimised by choosing

$$\mathbf{x} = \mathbf{e}_{\mathrm{n}}$$

where $\mathbf{e}_n$ is the eigen vector associated with the smallest eigen value λ_n.

SVD

- For an $n \times m$ matrix there exist unitary* matrices $\mathbf{U}$ and $\mathbf{V}$ such that

$$\mathbf{U} = [\mathbf{u}_1|\mathbf{u}_2|\ldots|\mathbf{u}_m] \longleftarrow m \times m \text{ matrix}$$

$$\mathbf{V} = [\mathbf{v}_1|\mathbf{v}_2|\ldots|\mathbf{v}_n] \longleftarrow n \times n \text{ matrix}$$

$$\mathbf{A} = \mathbf{USV}^{\mathrm{T}}, \text{ where } \mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{S}_1 = \begin{bmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_p \end{bmatrix}$$

and $s_1 \geq s_2 \geq \ldots \geq s_{\mathrm{p}} \geq 0, p = \min\{n,m\}$

*a real matrix $\mathbf{U}$ is unitary if $\mathbf{U}^{-1} = \mathbf{U}^{\mathrm{T}}$

SVD

- s_i is the i^{th} singular value of $\mathbf{A}$, the vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are the left and right singular vectors of $\mathbf{A}$.

- s_i^2 is an eigenvalue of $\mathbf{AA}^{\mathrm{T}}$ or $\mathbf{A}^{\mathrm{T}}\mathbf{A}$,
- $\mathbf{u}_i$ is an eigen vector of $\mathbf{AA}^{\mathrm{T}}$ and
- $\mathbf{v}_i$ is an eigen vector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

SVD

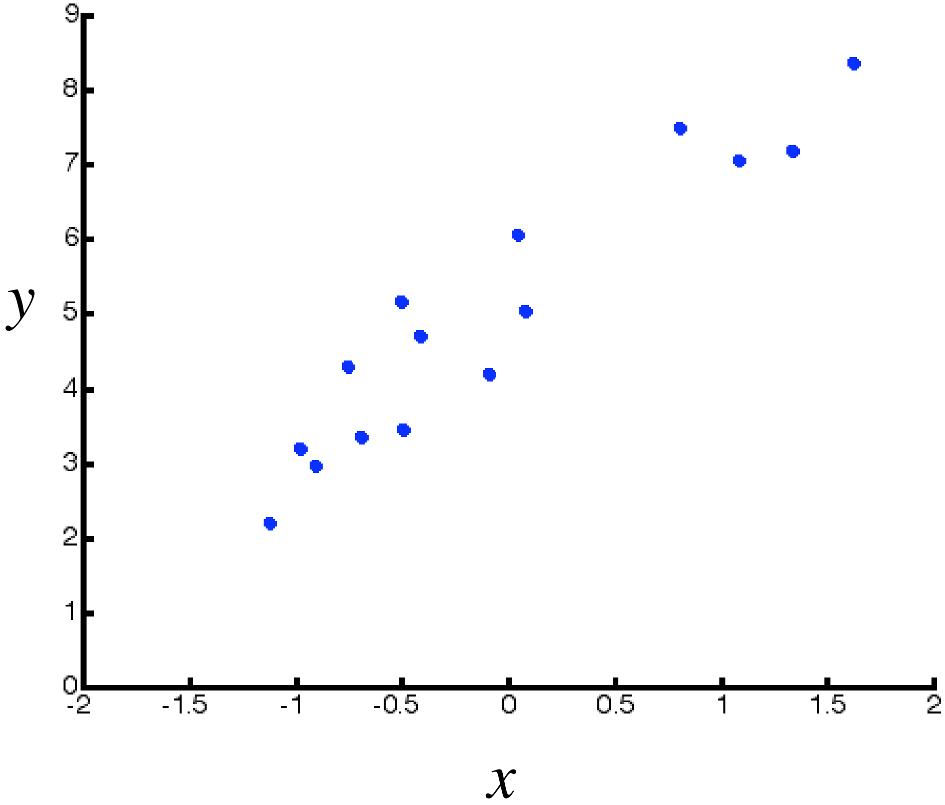
- $\mathbf{V} = [\mathbf{v}_1|\mathbf{v}_2|\ldots|\mathbf{v}_n]$
- is a matrix of eigen vectors of $\mathbf{A}^\mathrm{T}\mathbf{A}$ with associated eigen values s_i^2. The eigen vector corresponding to the smallest eigen value of $\mathbf{A}^\mathrm{T}\mathbf{A}$ is $\mathbf{v}_n$.
- Hence the non-zero $\mathbf{x}$ that minimises

$$\mathbf{Ax} = \mathbf{0}$$

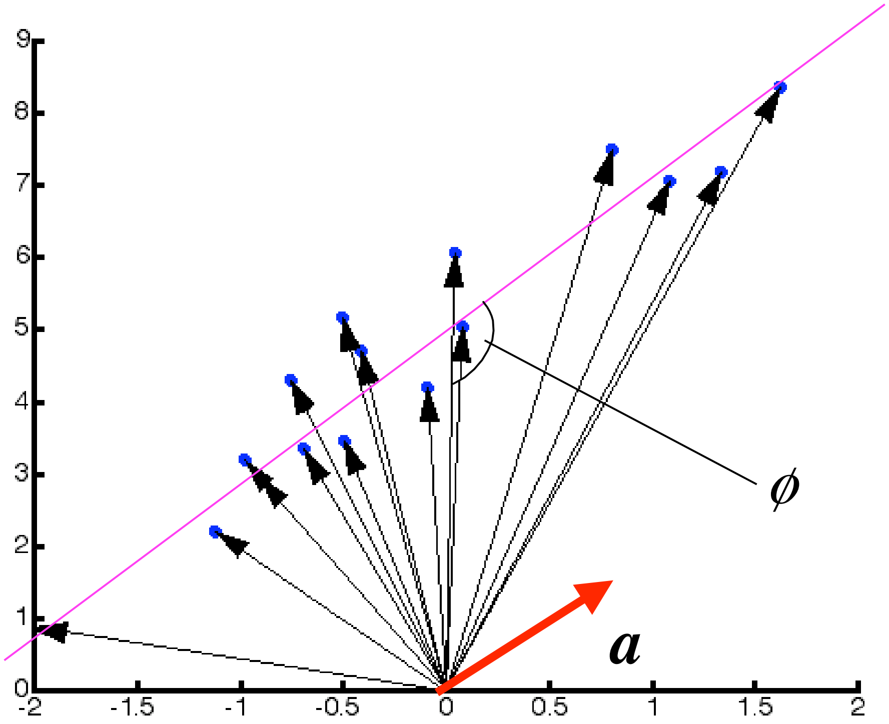
is $\mathbf{x} = \mathbf{v}_n$.

Example: Least Square Line Fitting

Data scatter



Data as 2D vectors

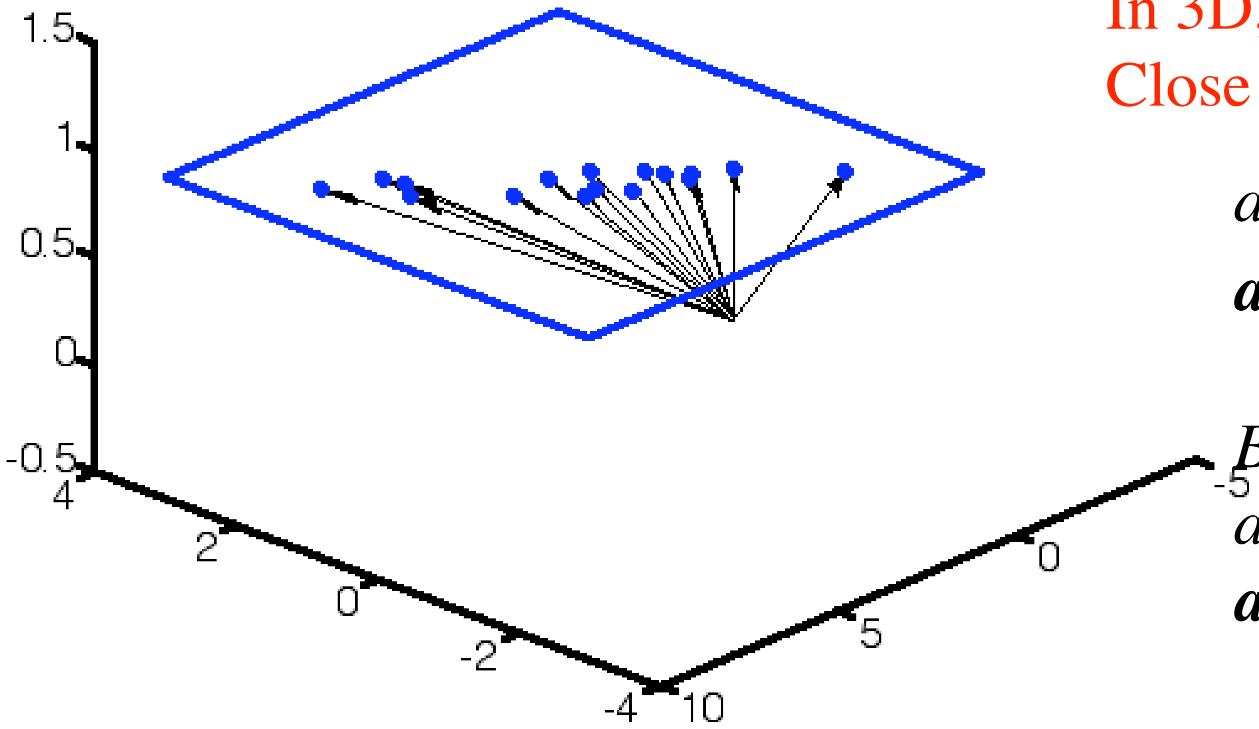


$$a\, x_i + b\, y_i = c$$

$$\boldsymbol{a}\cdot\boldsymbol{x} = \boldsymbol{c} = \|\boldsymbol{x}\|| \cos \phi$$

Introducing Homogenous Coordinates

Data as 3D homogenous vectors $p_i = [x_i \;\; y_i \;\; 1]'$



In 3D, the set of points lies
Close to a common plane

$a\, x_i + b\, y_i = c$

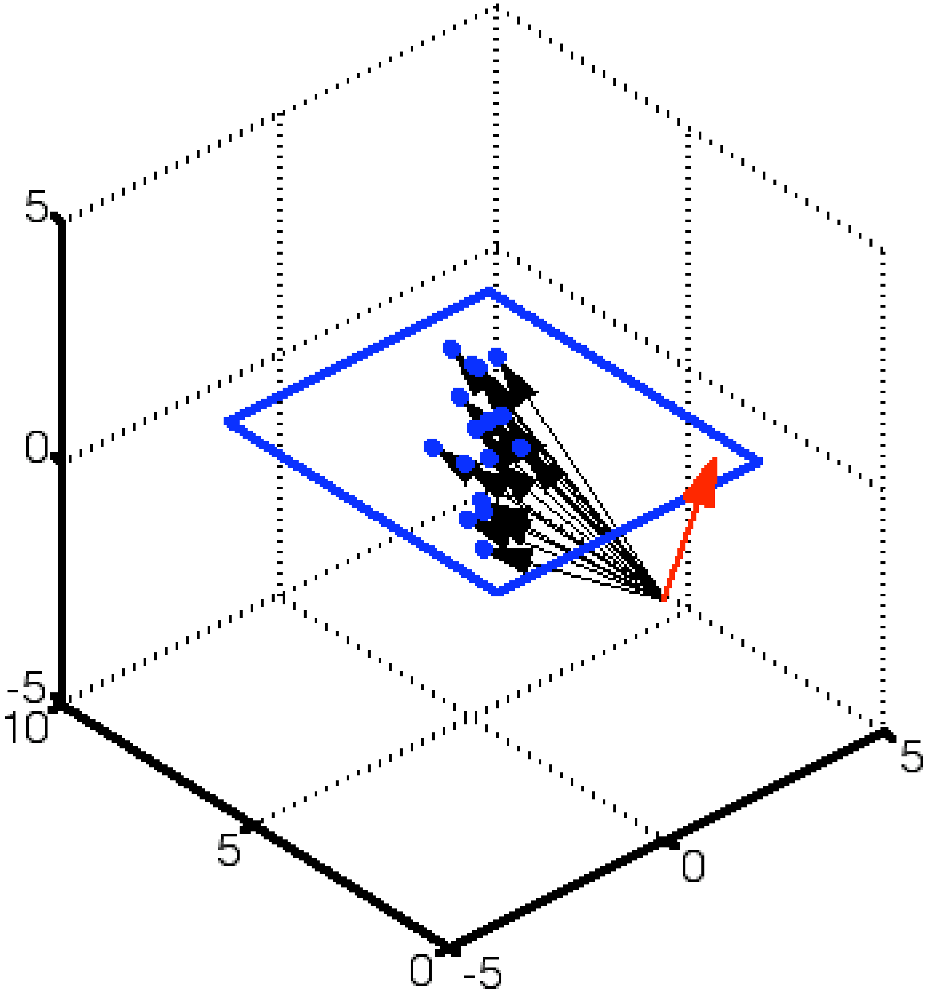
$\boldsymbol{a} \bullet \boldsymbol{x} = \boldsymbol{c} =$ **||x|||** cos ϕ

Becomes

$a\, x_i + b\, y_i + c\, 1 = 0$

$\boldsymbol{a} \cdot \boldsymbol{p}_i = \boldsymbol{0}$

Geometry of solution



5
0
-5
10
5
0
-5
0
5