Robert Collins
CMPEN454

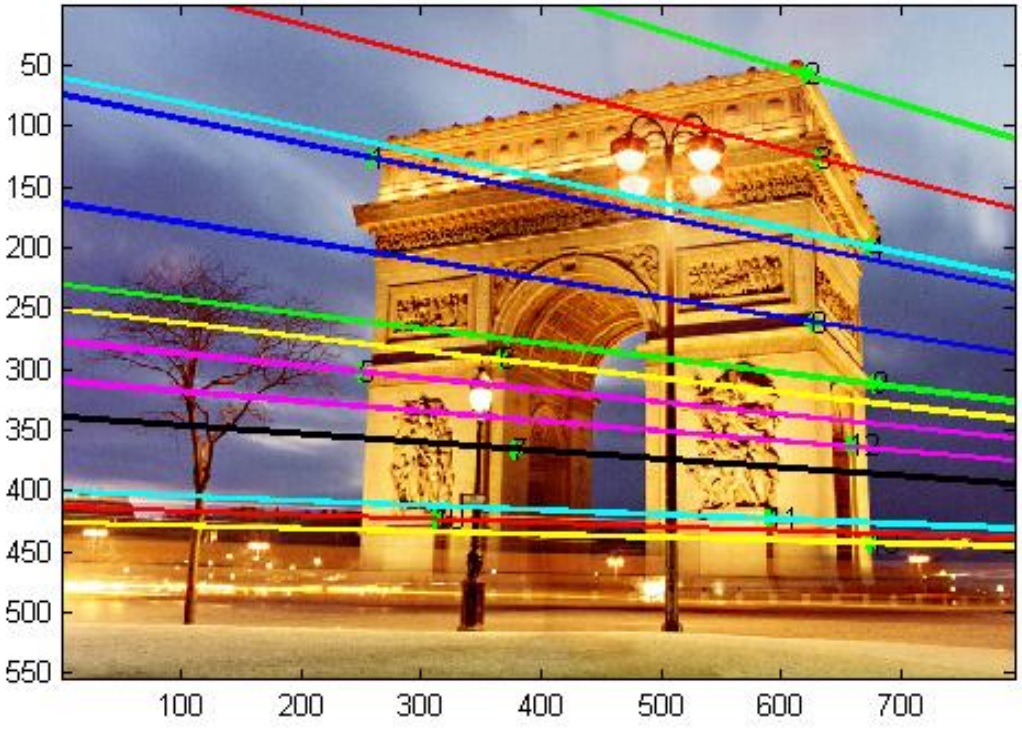
The Eight-Point Algorithm

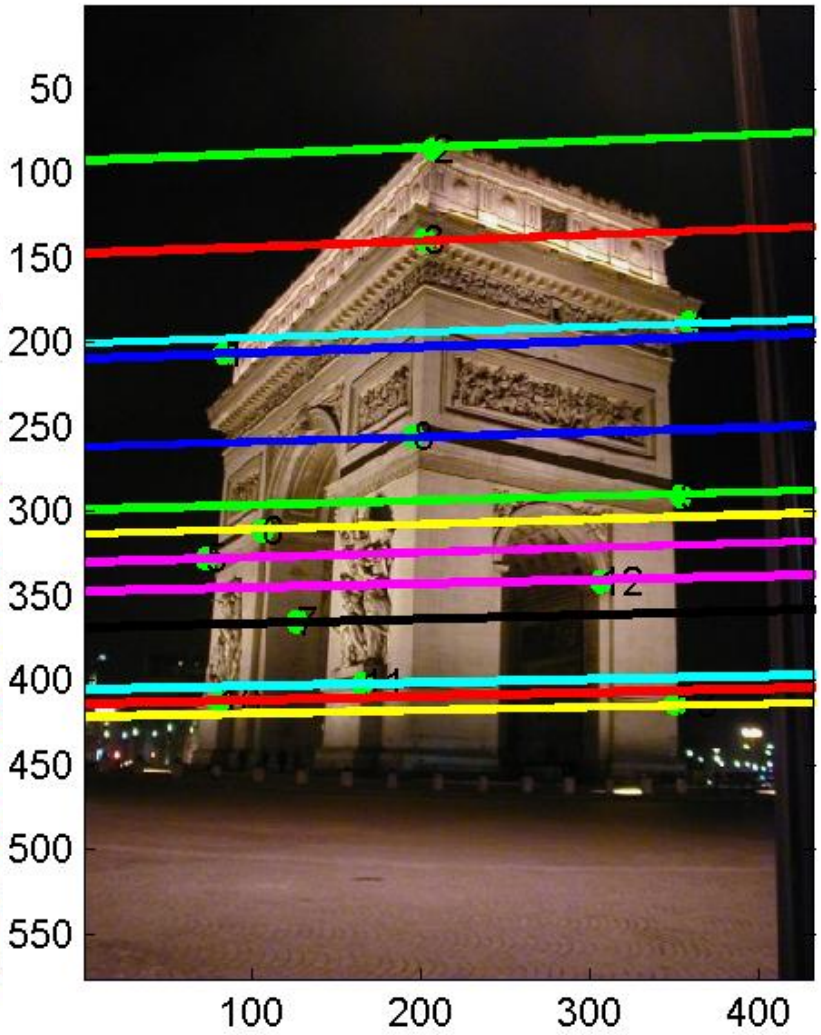
Background Readings
T&V 7.3 and 7.4
Szeliski 7.2

Robert Collins
CMPEN454

Reminder:

$$F = \begin{pmatrix} -0.00310695 & -0.0025646 & 2.96584 \\ -0.028094 & -0.00771621 & 56.3813 \\ 13.1905 & -29.2007 & -9999.79 \end{pmatrix}$$





Essential/Fundamental Matrix

The essential and fundamental matrices are 3x3 matrices that "encode" the epipolar geometry of two views.

Motivation: Given a point in one image, multiplying by the essential/fundamental matrix will tell us which epipolar line to search along in the second view.

Robert Collins
CMPEN454

E/F Matrix Summary

Longuet-Higgins equation $p_r^T E p_l = 0$

Epipolar lines: $\tilde{p_r}^T \tilde{l_r} = 0$ $\tilde{p_l}^T \tilde{l_l} = 0$

$\tilde{l_r} = E p_l$ $\tilde{l_l} = E^T p_r$

Epipoles: $e_r^T E = 0$ $E e_l = 0$

E vs F: E works in film coords (calibrated cameras)
F works in pixel coords (uncalibrated cameras)

Computing F from Point Matches

- Assume that you have m correspondences
- Each correspondence satisfies:

$$\bar{p_r}_i^T F \bar{p}_{li} = 0 \quad i = 1, \ldots, m$$

- F is a 3x3 matrix (9 entries)
- Set up a **HOMOGENEOUS** linear system with 9 unknowns

Computing F

$$\bar{p}_{li} = (x_i \;\; y_i \;\; 1)^T \quad \bar{p}_{ri} = (x'_i \;\; y'_i \;\; 1)^T$$

$$\bar{p_r}_i^T F \bar{p}_{li} = 0 \quad i = 1, \ldots, m$$

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

Computing F

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

$$x_i x'_i f_{11} + x_i y'_i f_{21} + x_i f_{31} +$$
$$y_i x'_i f_{12} + y_i y'_i f_{22} + y_i f_{32} +$$
$$x'_i f_{13} + y'_i f_{23} + f_{33} = 0$$

Robert Collins
CMPEN454

Computing F

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

Given m point correspondences…

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx'_m & x_my'_m & x_m & y_mx'_m & y_my'_m & y_m & x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

Think: how many points do we need?

How Many Points?

Although there are 9 unknowns, and each point contributes one row of constraints to the linear system of equations, there are really only 8 degree of freedom in F. (We can multiply through by any nonzero scale factor and the equations still hold). So we really only need 8 points.

Hence: The Eight Point algorithm!

Solving Homogeneous Systems

Self-study

Assume that we need the non trivial solution of:

$$Ax = 0$$

with m equations and n unknowns, m >= n – 1

Since the norm of x is arbitrary, we will look for a solution with norm ||x|| = 1

Note, x= $\begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix}$

Self-study

Least Square solution

We want Ax as close to 0 as possible and ||x|| =1:

$$\min_{\mathbf{x}} ||A\mathbf{x}||^2 \text{ s.t. } ||x||^2 = 1$$

$$||A\mathbf{x}||^2 = (A\mathbf{x})^T(A\mathbf{x}) = \mathbf{x}^T A^T A\mathbf{x}$$

$$||\mathbf{x}||^2 = \mathbf{x}^T\mathbf{x} = 1$$

Self-study

Optimization with constraints

Define the following cost:

$$\mathcal{L}(\mathbf{x}) = \mathbf{x}^T A^T A \mathbf{x} - \lambda(\mathbf{x}^T \mathbf{x} - 1)$$

This cost is called the LAGRANGIAN *cost* and λ is called the LAGRANGIAN *multiplier*

The Lagrangian incorporates the constraints into the cost function by introducing extra variables.

Self-study

Optimization with constraints

$$\min_{\mathbf{x}} \left\{ \mathcal{L}(\mathbf{x}) = \mathbf{x}^T A^T A \mathbf{x} - \lambda(\mathbf{x}^T \mathbf{x} - 1) \right\}$$

Taking derivatives wrt to x and λ:

$$A^T A \mathbf{x} - \lambda \mathbf{x} = 0$$

$$\mathbf{x}^T \mathbf{x} - 1 = 0$$

- The first equation is an eigenvector problem
- The second equation is the original constraint

Self-study

Optimization with constraints

$$A^T A\mathbf{x} - \lambda\mathbf{x} = 0$$
$$A^T A\mathbf{x} = \lambda\mathbf{x}$$

- x is an eigenvector of A^TA with eigenvalue λ: e_λ

$$\mathcal{L}(\mathbf{e}_\lambda) = \mathbf{e}_\lambda{}^T A^T A\mathbf{e}_\lambda - \lambda(\mathbf{e}_\lambda{}^T\mathbf{e}_\lambda - 1)$$
$$\mathcal{L}(\mathbf{e}_\lambda) = \lambda\mathbf{e}_\lambda{}^T\mathbf{e}_\lambda = \lambda$$

- We want the eigenvector with <u>smallest</u> eigenvalue

Computing F: The 8 pt Algorithm

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ & & & & & & & & \\ & & & & \vdots & & & & \\ & & & & & & & & \\ x_8x'_8 & x_8y'_8 & x_8 & y_8x'_8 & y_8y'_8 & y_8 & x'_8 & y'_8 & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

$$\mathbf{Ax} = 0$$ A has rank 8

$$\min_{\mathbf{x}} ||A\mathbf{x}||^2 \text{ s.t. } ||x||^2 = 1$$

- Find eigenvector of A^TA with smallest eigenvalue!

Algorithm EIGHT_POINT

The input is formed by m point correspondences*, m >= 8

- Construct the m x 9 matrix A
- Find the eigenvalues/eigenvectors of $A^T A$
- The entries of F are the components of the eigenvector corresponding to the smallest eigenvalue

*Important: the point correspondences should not all lie on one line or one plane in the world... they should span the 3D space of the scene.

Algorithm EIGHT_POINT

F must be singular (remember, it is rank 2, since it is important for it to have a left and right nullspace, i.e. the epipoles). To enforce rank 2 constraint:

- Find the SVD* of F: $F = U_f D_f V_f^T$
- Set smallest s.v. of F to 0 to create D'_f
- Recompute F: $F = U_f D'_f V_f^T$

*SVD = Singular Value Decomposition

Numerical Details

- The coordinates of corresponding points can have a wide range leading to numerical instabilities.
- It is better to first normalize them so they have average 0 and std dev 1 and denormalize F at the end:

$$\hat{x}_i = T x_i \qquad \hat{x}'_i = T' x'_i$$

$$F = (T')^{-1} F_n T$$

$$T = \begin{bmatrix} \frac{1}{\sigma^2} & 0 & -\mu_x \\ 0 & \frac{1}{\sigma^2} & -\mu_y \\ 0 & 0 & 1 \end{bmatrix}$$

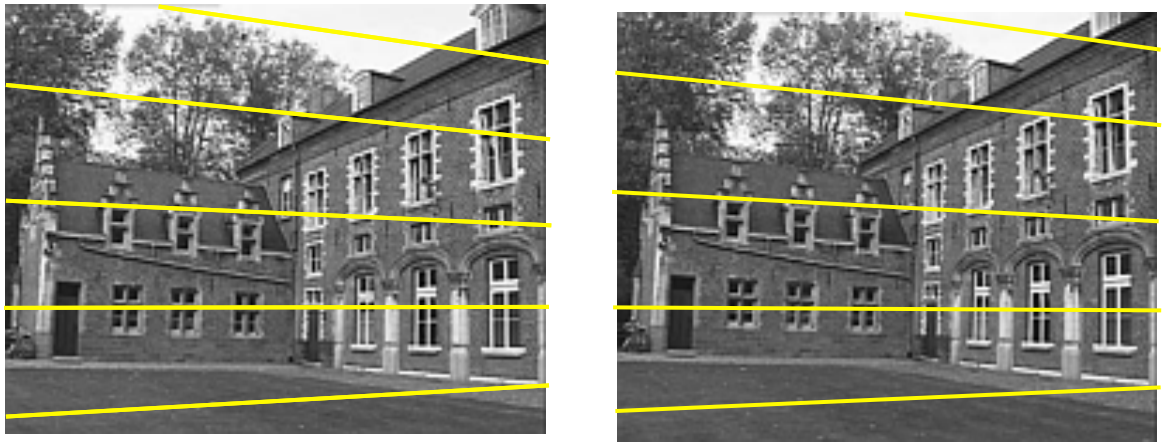
This method of preconditioning the data prior to running the 8-point algorithm is due to Richard Hartley, in a paper called "In Defense of the Eight-Point Algorithm."

DEMO

- Running Matlab code eightpoint.m

Stereo Rectification

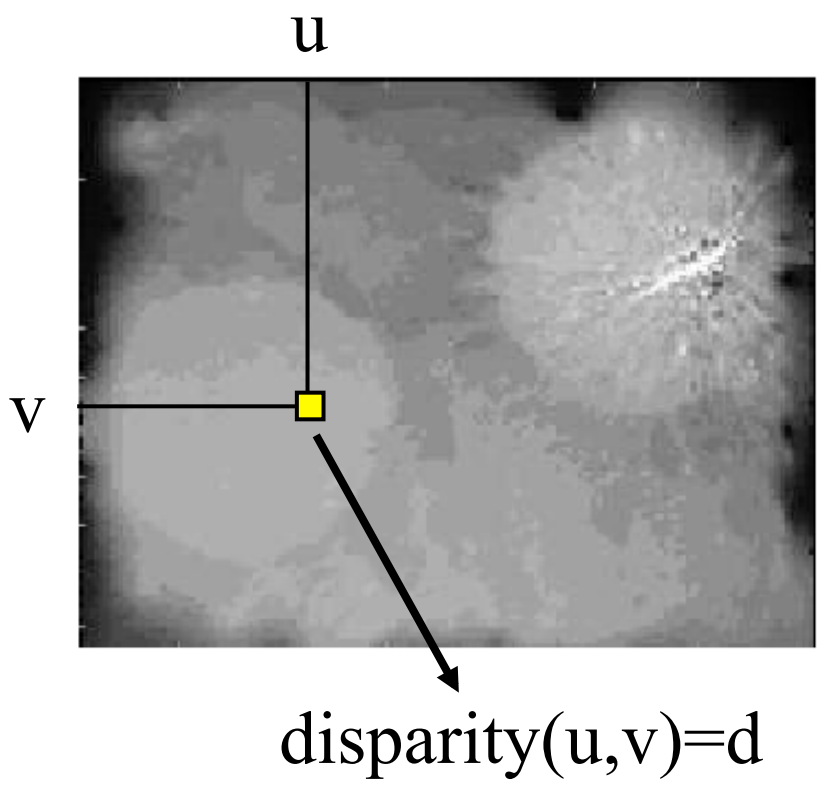
- Assume we know the epipolar geometry, which tells us conjugate epipolar lines to use when searching for matching image patches.
- In general, these lines are not parallel in the images, so search is not as efficient as in simple stereo (where you search along rows).
- Stereo Rectification seeks to transform both images so that conjugate epipolar lines are horizontal (rows) in both images.



We would prefer these epipolar lines to be parallel along corresponding rows of the two images, like in the case of simple stereo.

Reconstruction

Although we have a dense disparity map, when talking about recovering 3D scene structure, we could consider it to just be a set of point matches.



<u>point match:</u>

(u,v) in left image
matches
(u-d,v) in right image

Robert Collins
CMPEN454

Stereo Reconstruction

Given point correspondences, how to compute 3D point positions using triangulation.

Results depend on how calibrated the system is:

1) Intrinsic and extrinsic parameters known
   Can compute metric 3D geometry
2) Only intrinsic parameters known
   Unknown scale factor
3) Neither intrinsic nor extrinsic known
   Recover structure up to an unknown projective transformation of the scene

Fully Calibrated Stereo

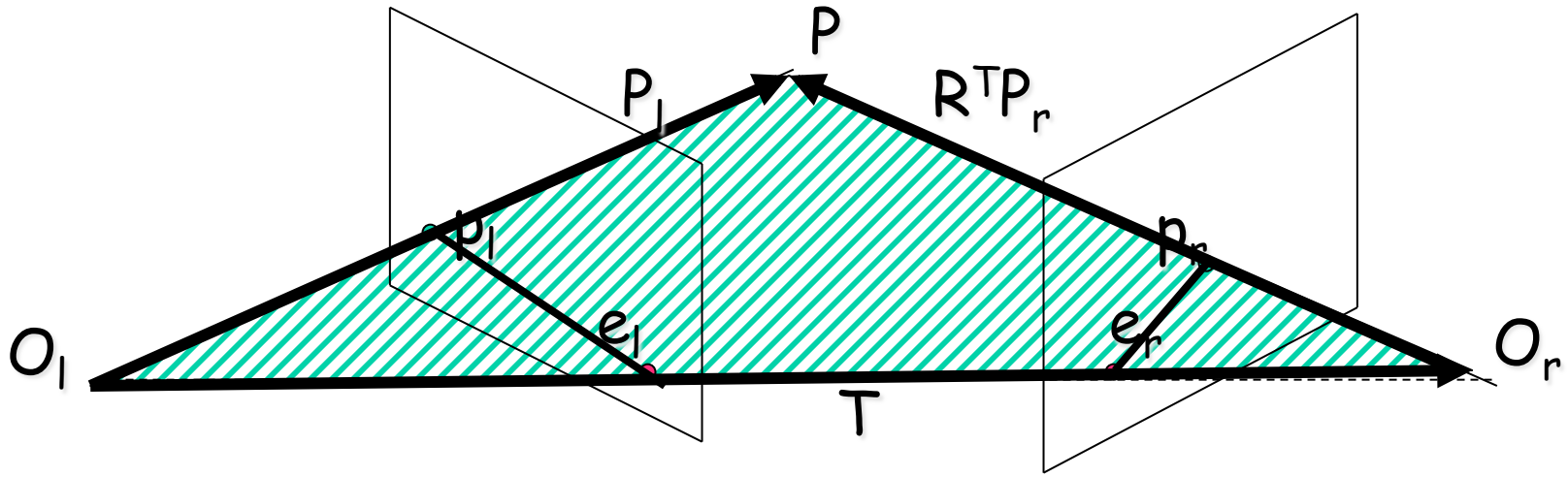
Known intrinsics -- can compute viewing rays in camera coordinate system

Know extrinsics -- know how rays from both cameras are positioned in 3D space

Reconstruction: triangulation of viewing rays

Robert Collins
CMPEN454

Calibrated Triangulation



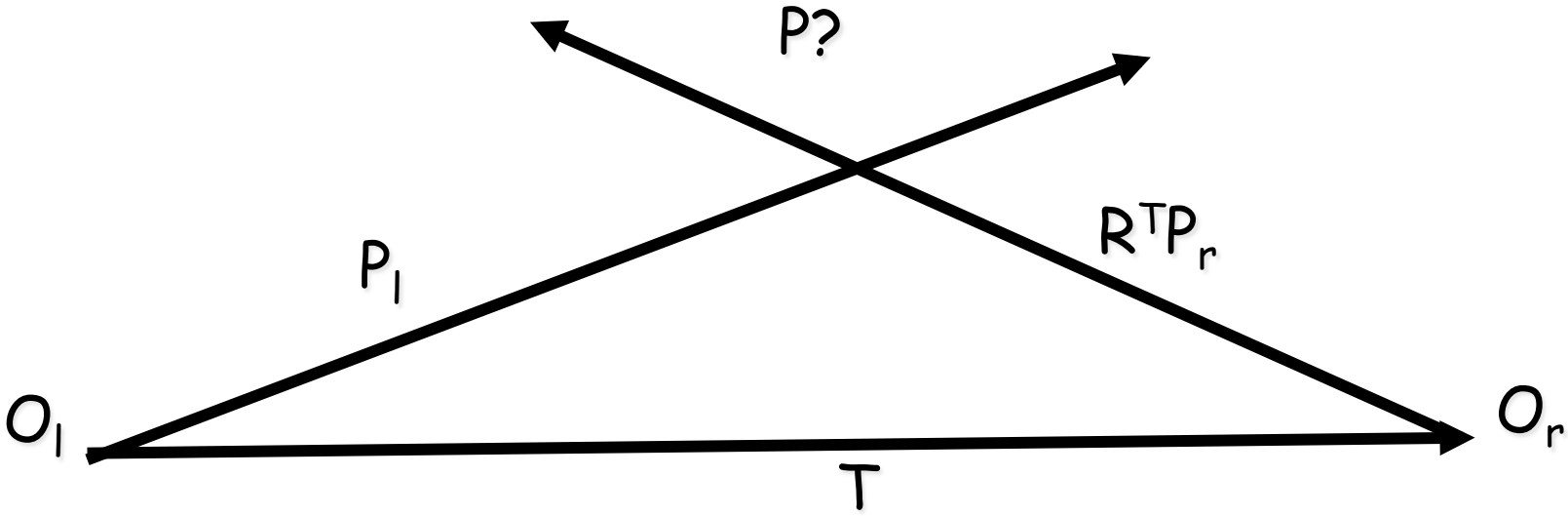
ideally, P is the point of intersection of two 3D rays:

ray through O_l with direction P_l

ray through O_r with direction $R^T P_r$

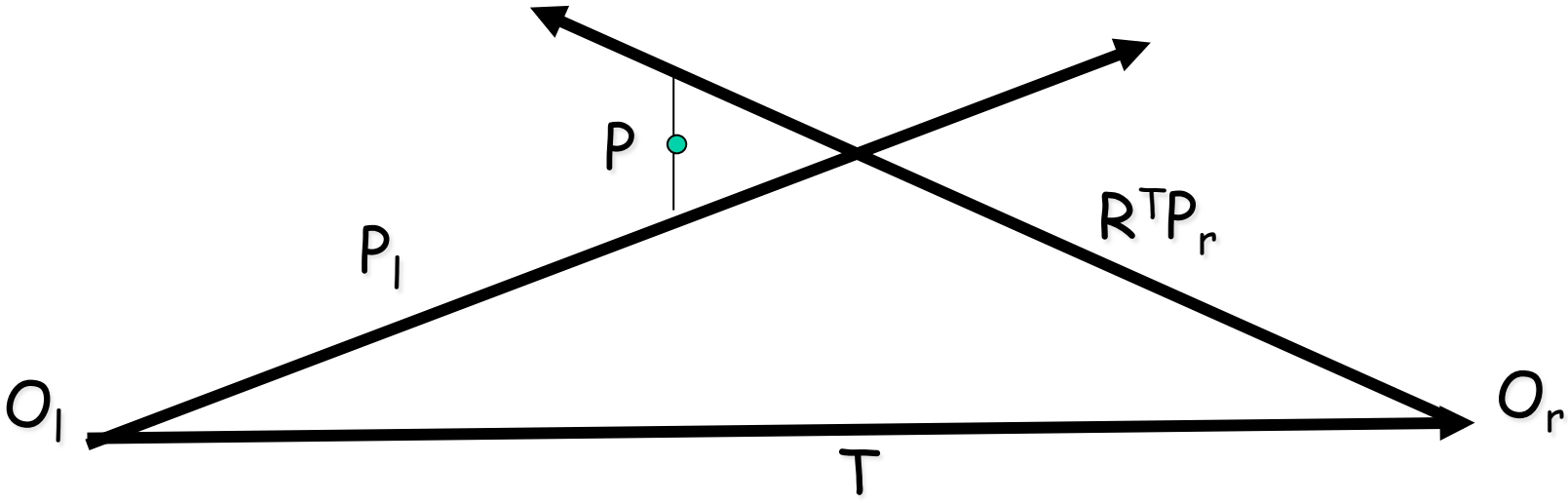
Triangulation with Noise

Unfortunately, these rays typically don't intersect due to noise in point locations and calibration params



Triangulation with Noise

Unfortunately, these rays typically don't intersect due to noise in point locations and calibration params

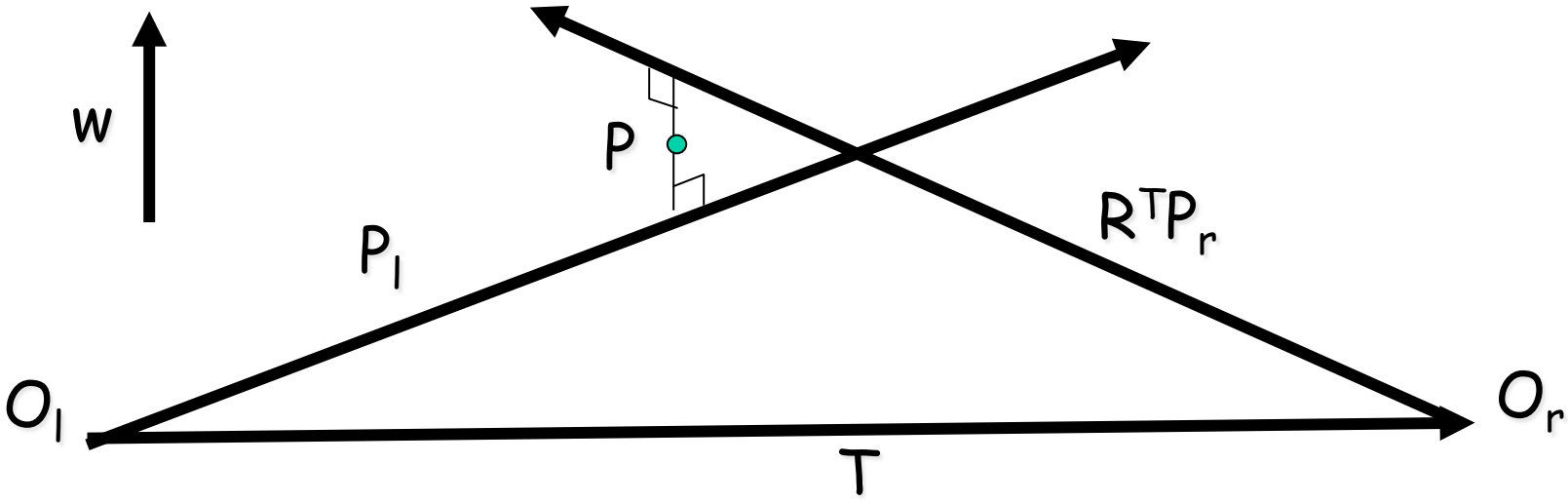


Solution: Choose P as the "pseudo-intersection point". This is point that minimizes the sum of squared distance to both rays. (The SSD is 0 if the rays exactly intersect)

Solution from T&V Book

P is midpoint of the segment perpendicular to P_l and $R^T P_r$

Let $w = P_l \times R^T P_r$ (this is perpendicular to both)

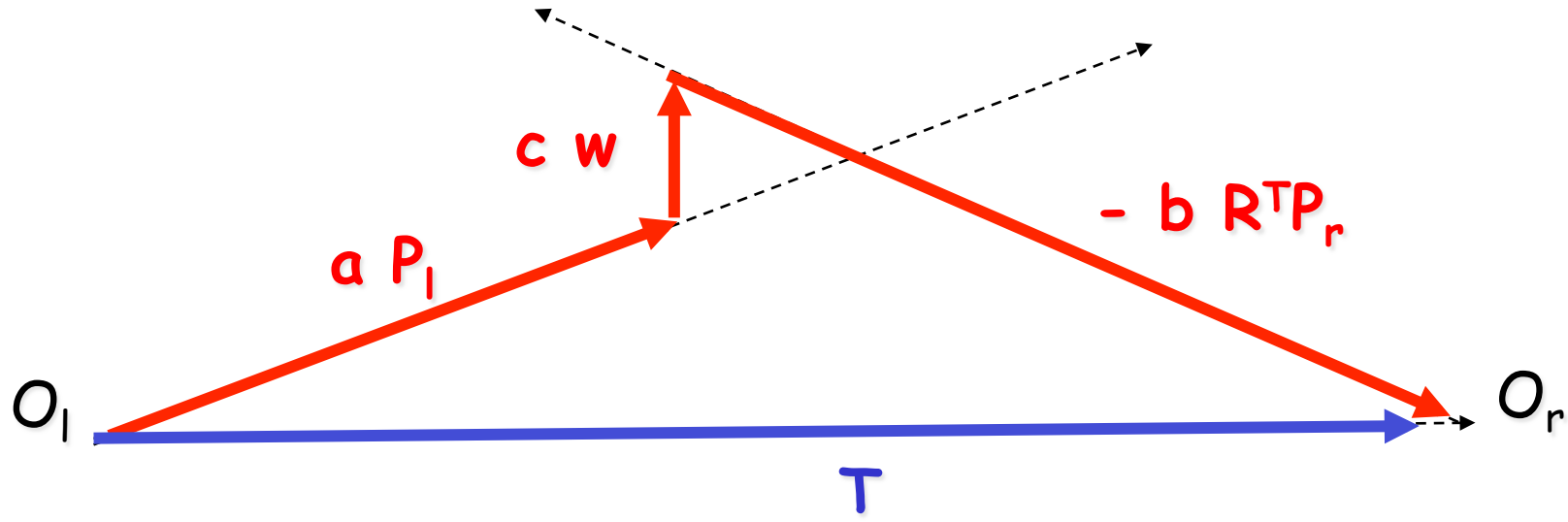


Introducing three unknown scale factors a,b,c we note we can write down the equation of a "circuit"

Solution from T&V Book

Writing vector "circuit diagram" with unknowns a,b,c

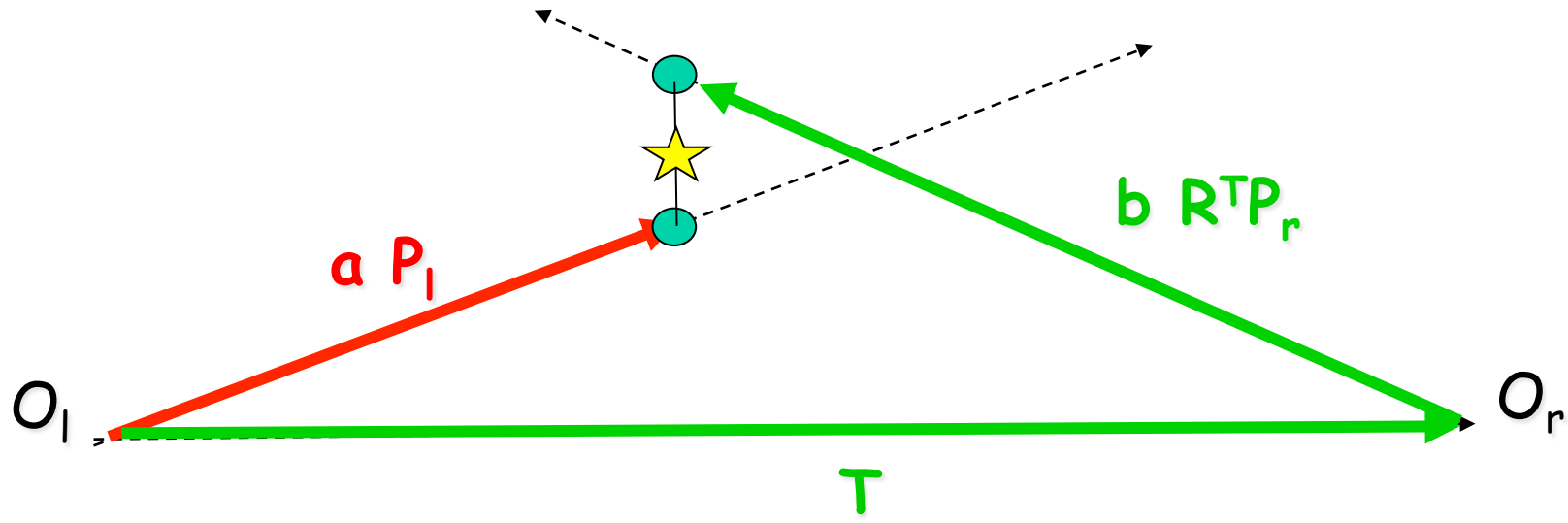
$$a\,P_l + c\,(P_l \,X\, R^T P_r) - b\,R^T P_r = T$$



note: this is three linear equations in three unknowns a,b,c
=> can solve for a,b,c

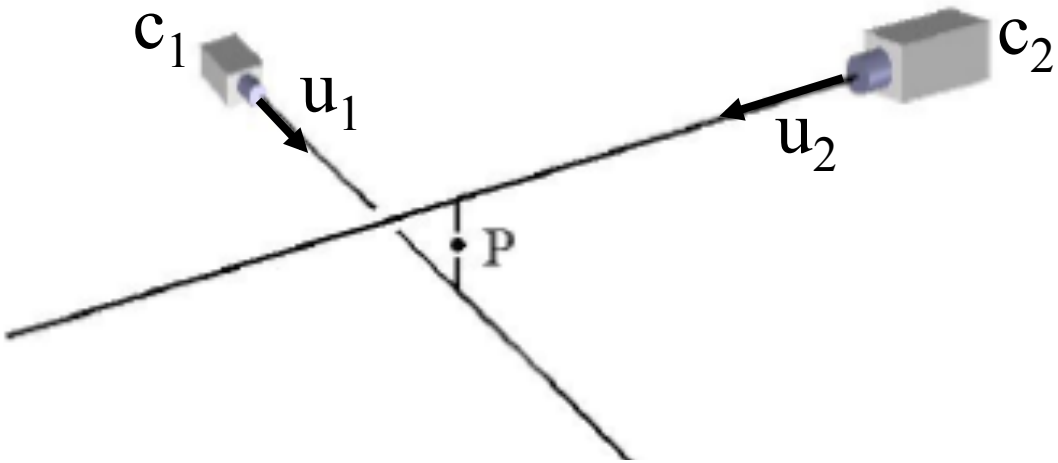
Solution from T&V Book

After finding a,b,c , solve for midpoint of line segment between points $O_l + a P_l$ and $O_l + T + b R^T P_r$



Alternate Solution

I prefer an alternate solution based on using least squares to solve for an unknown point P that minimizes SSD to viewing rays. Why? it generalizes readily to N cameras.



w_i = weight or certainty of point i's measurement

$$[\sum_{i}^{n} w_i(I - u_i u_i')]\, P = \sum_{i}^{n} w_i(I - u_i u_i')c_i$$