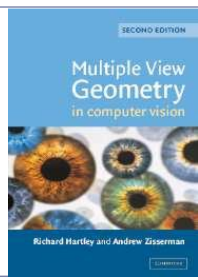
Marc Pollefeys
Full Professor
Computer Vision and Geometry lab
Institute of Visual Computing
Department of Computer Science
ETH Zurich

Tel: +41 44 632 31 05
Fax: +41 44 632 17 39
E-mail: marc.pollefeys@inf.ethz.ch
Web:http://www.inf.ethz.ch/personal/pomarc

http://www.cs.unc.edu/~marc/

Adjunct/Research Professor
Department of Computer Science
University of North Carolina at Chapel Hill

Tel: (919) 962 1845
Fax: (919) 962 1699
E-mail: marc@cs.unc.edu



Multiple View Geometry in Computer Vision

2011.09.
lbg@dongseo.ac.kr

Figure 1.1: An image of a scene

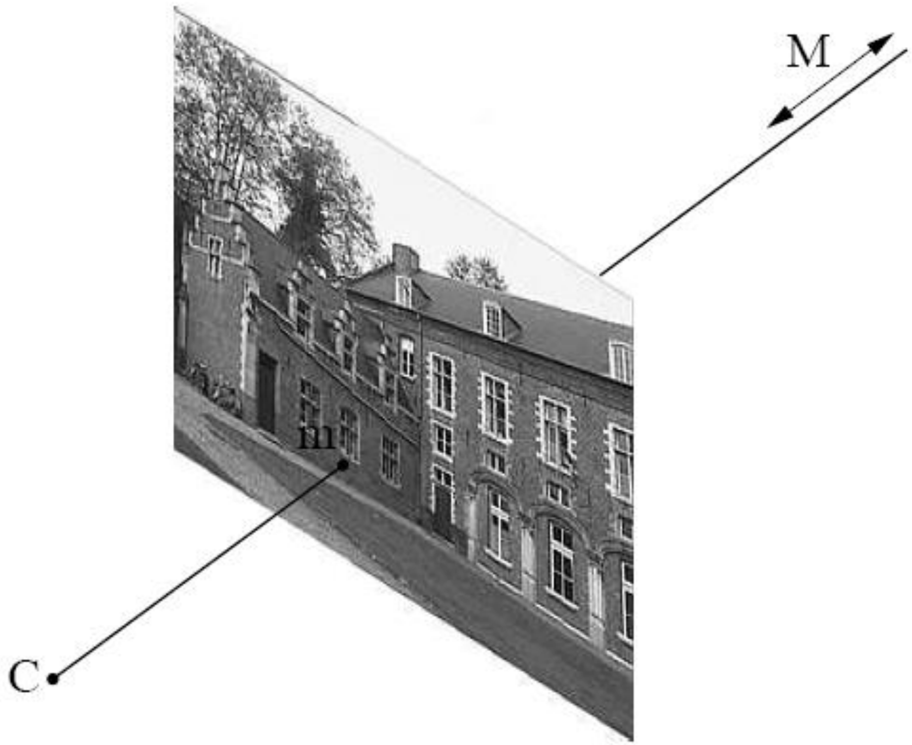


Figure 1.2: Back-projection of a point along the line of sight.

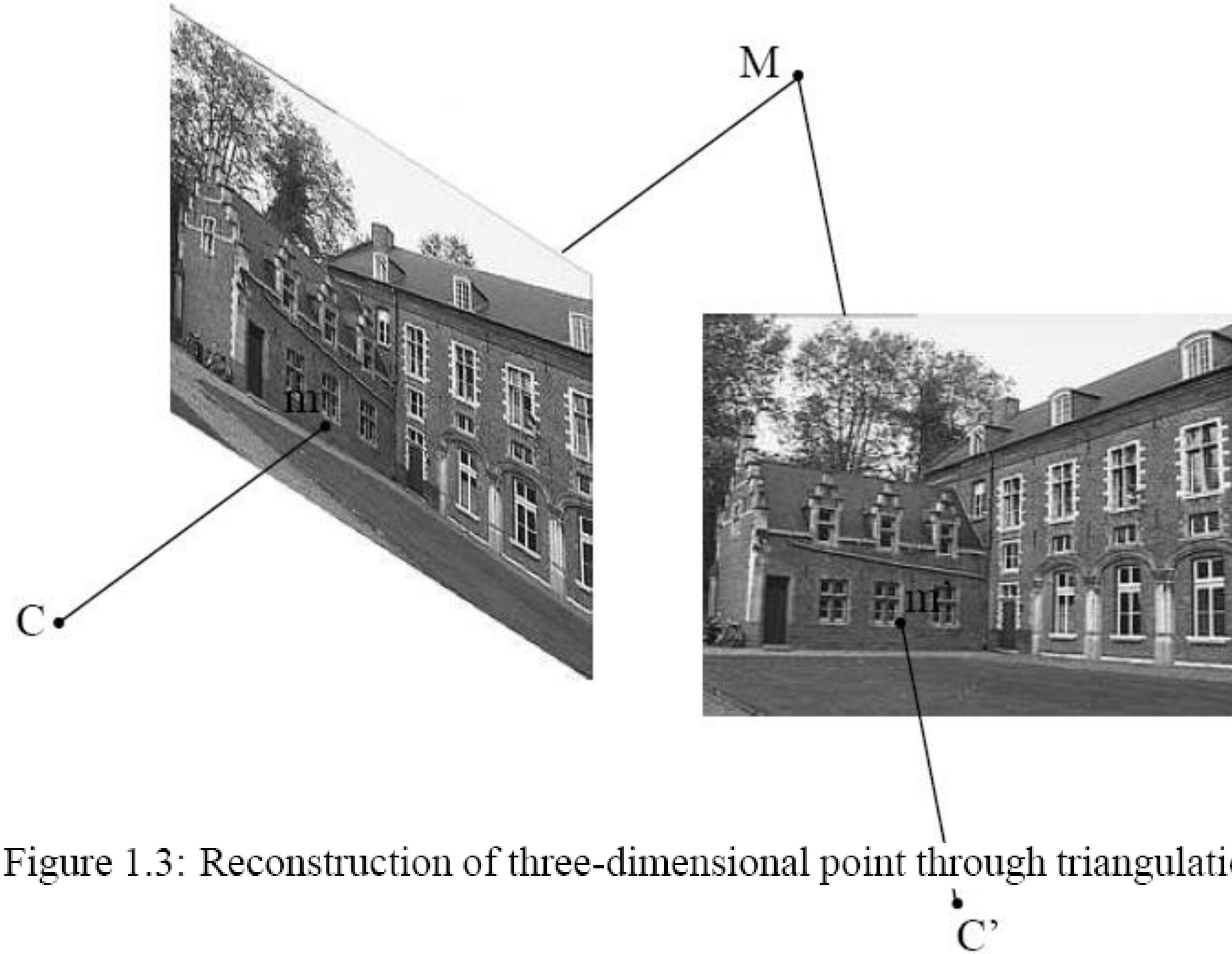
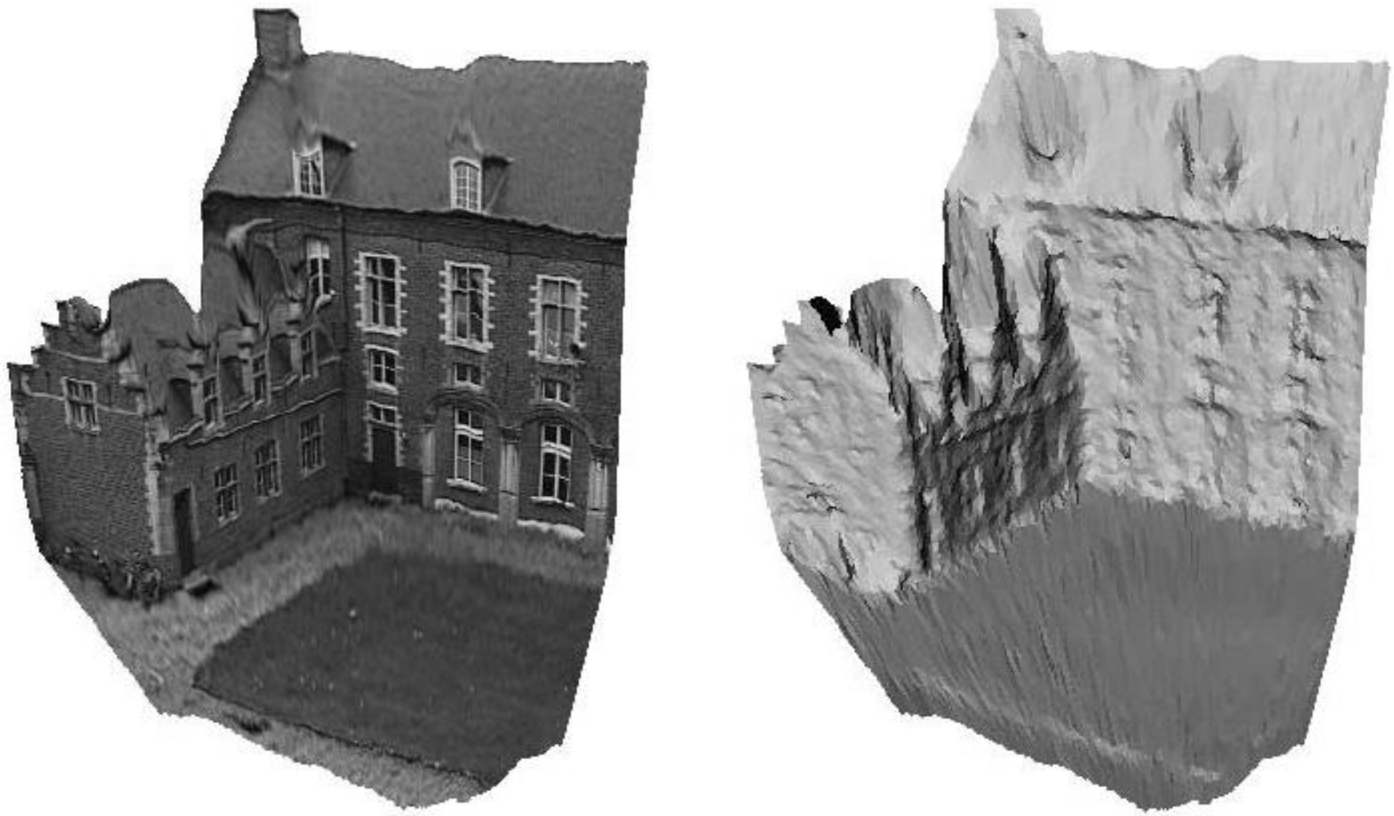


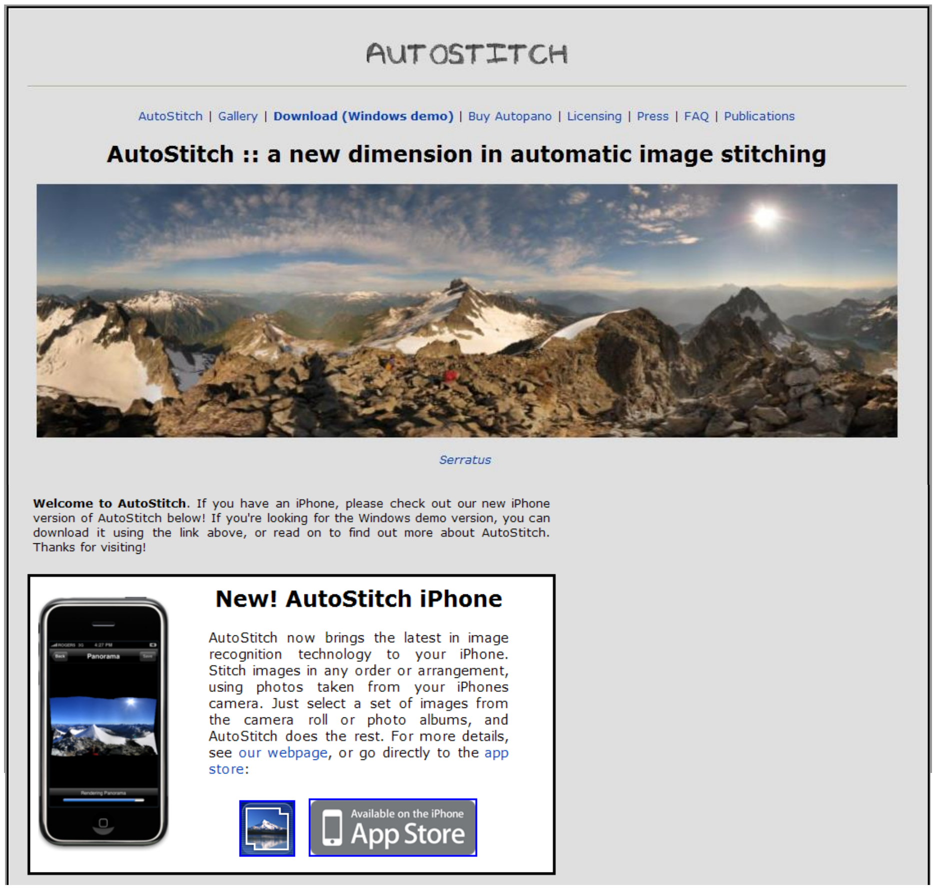
Figure 1.3: Reconstruction of three-dimensional point through triangulation.

Visual 3D Modeling from Images



AutoStitch

- http://cs.bath.ac.uk/brown/autostitch/autostitch.html



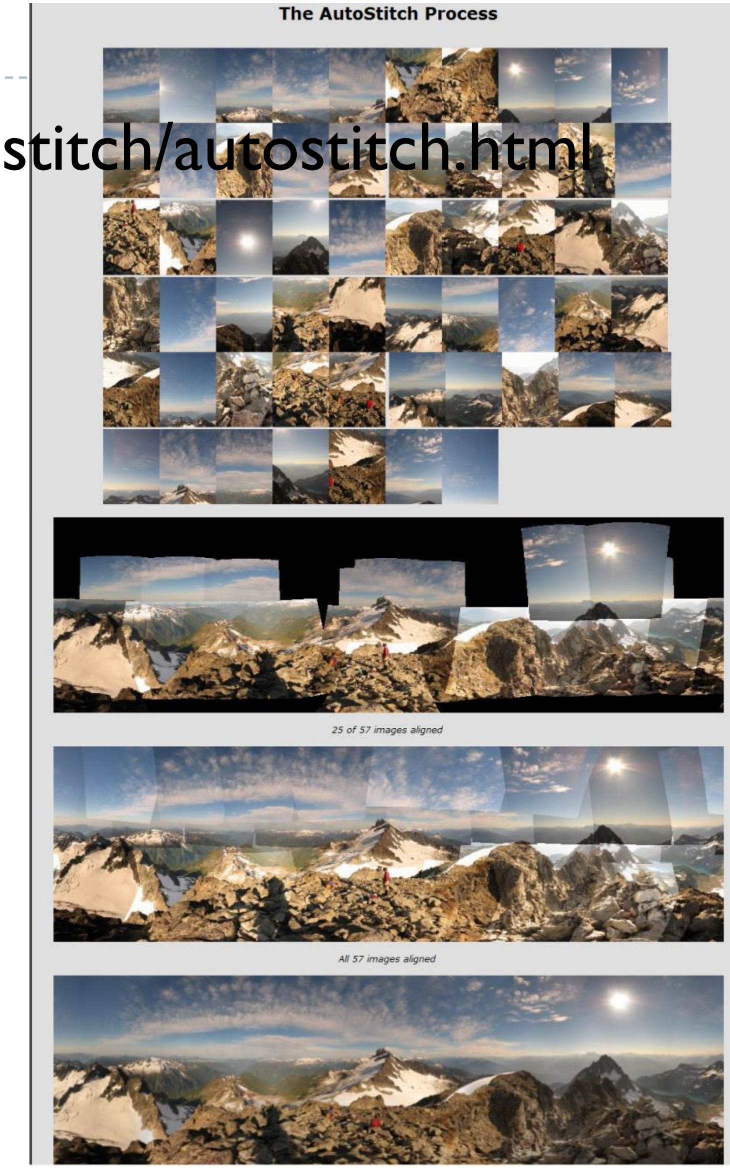
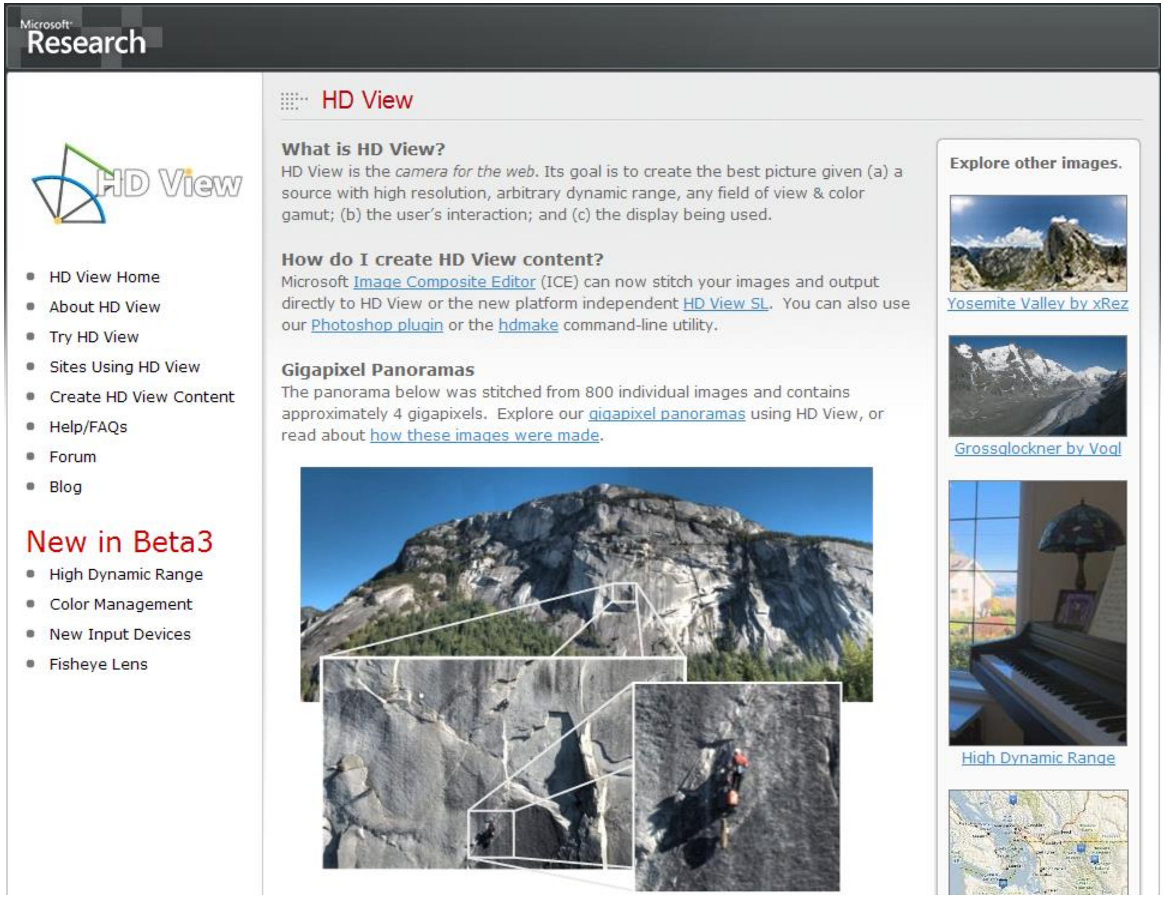


Image Composite Editor

- http://research.microsoft.com/en-us/um/redmond/groups/ivm/ice/

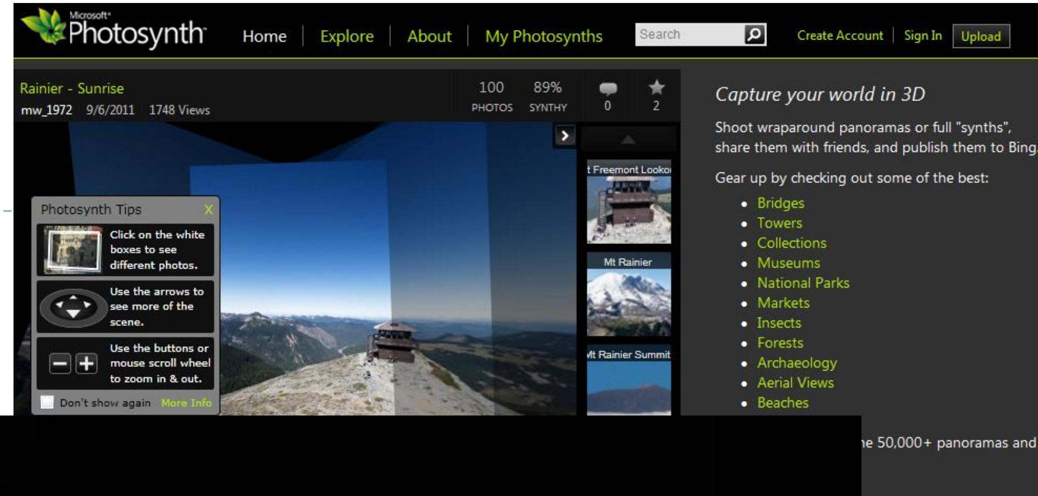
Deep Zoom & HD View

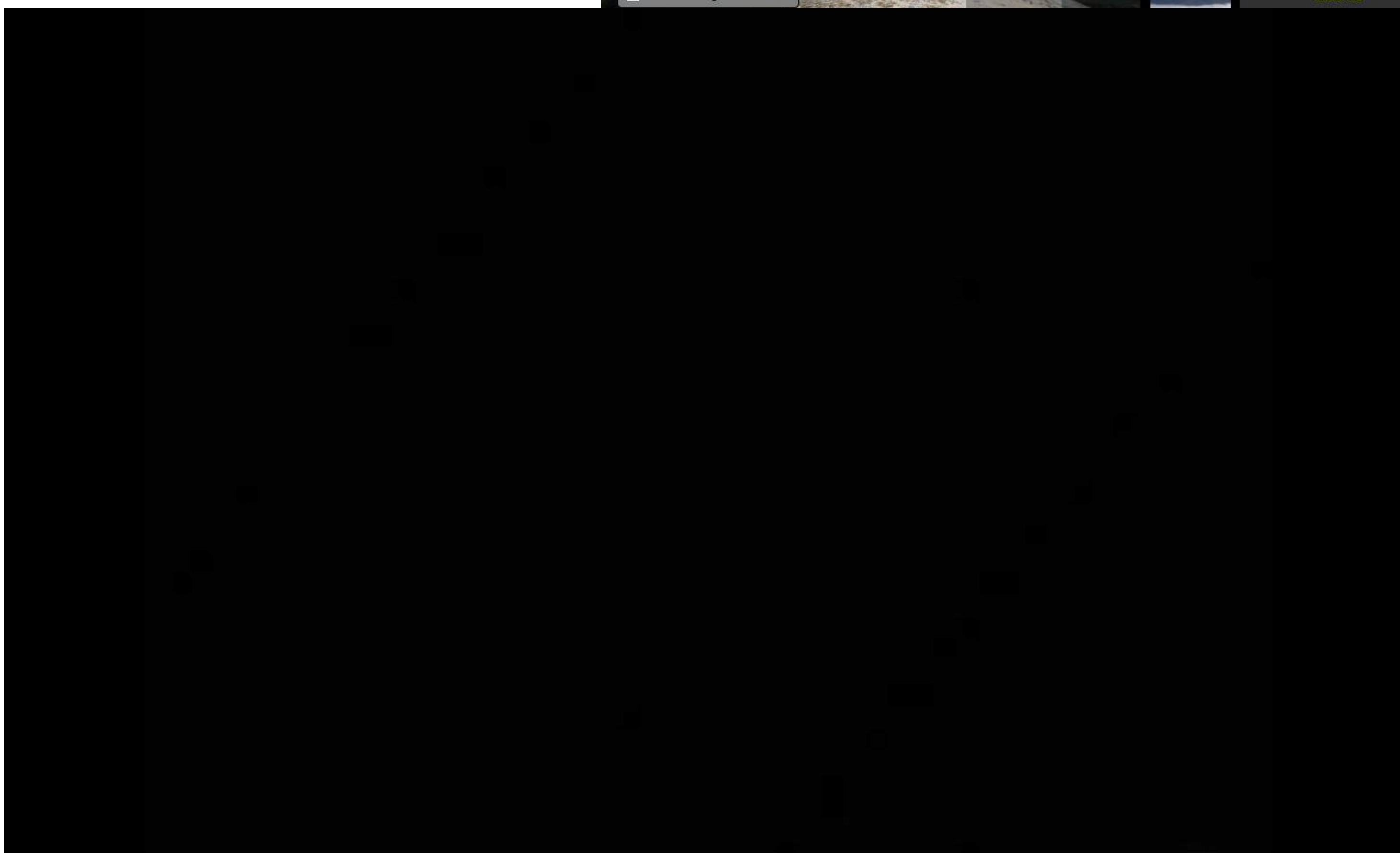
- http://research.microsoft.com/en-us/um/redmond/groups/ivm/HDView/



Photosynth

- http://photosynth.net/





PhotoCity

- http://photocitygame.com/

Projective Transformations

lbg@dongseo.ac.kr

Homogeneous coordinates

Homogeneous representation of lines

$$ax + by + c = 0 \qquad (a,b,c)^{\mathsf{T}}$$

$$(ka)x + (kb)y + kc = 0, \forall k \neq 0 \qquad (a,b,c)^{\mathsf{T}} \sim k(a,b,c)^{\mathsf{T}}$$

equivalence class of vectors, any vector is representative

Set of all equivalence classes in $\mathbf{R}^3 - (0,0,0)^{\mathsf{T}}$ forms $\mathbf{P}^2$

Homogeneous representation of points

$\mathrm{x} = (x, y, 1)^{\mathsf{T}}$ on $\mathrm{l} = (a,b,c)^{\mathsf{T}}$ if and only if $ax + by + c = 0$

$$(x,y,1)(a,b,c)^{\mathsf{T}} = (x,y,1)\mathrm{l} = 0 \qquad (x, y, 1)^{\mathsf{T}} \sim k(x, y, 1)^{\mathsf{T}}, \forall k \neq 0$$

The point **x** lies on the line **l** if and only if $\mathbf{x}^{\mathsf{T}}\mathbf{l} = \mathbf{l}^{\mathsf{T}}\mathbf{x} = 0$

Homogeneous coordinates $(x, y, z)^{\mathsf{T}}$ but only 2DOF

Inhomogeneous coordinates $(x, y)^{\mathsf{T}}$

Points from lines and vice-versa

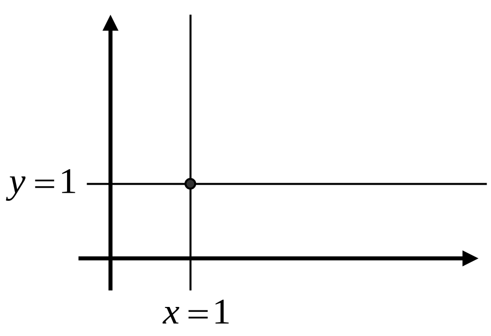
Intersections of lines

The intersection of two lines l and l' is $\mathrm{x} = \mathrm{l} \times \mathrm{l}'$

Line joining two points

The line through two points x and x' is $\mathrm{l} = \mathrm{x} \times \mathrm{x}'$

Example



$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

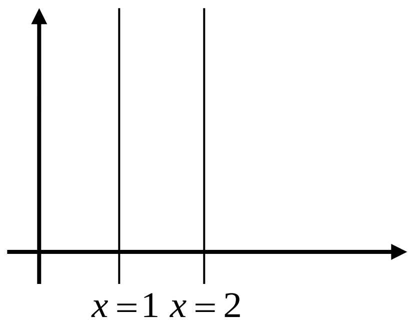
$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} \mathbf{i} - \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} \mathbf{j} + \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \mathbf{k}$$

Ideal points and the line at infinity

Intersections of parallel lines

$\mathrm{l} = (a, b, c)^\mathsf{T}$ and $\mathrm{l}' = (a, b, c')^\mathsf{T}$ $\quad \mathrm{l} \times \mathrm{l}' = (b, -a, 0)^\mathsf{T}$

Example



$(b, -a)$ tangent vector

(a, b) normal direction

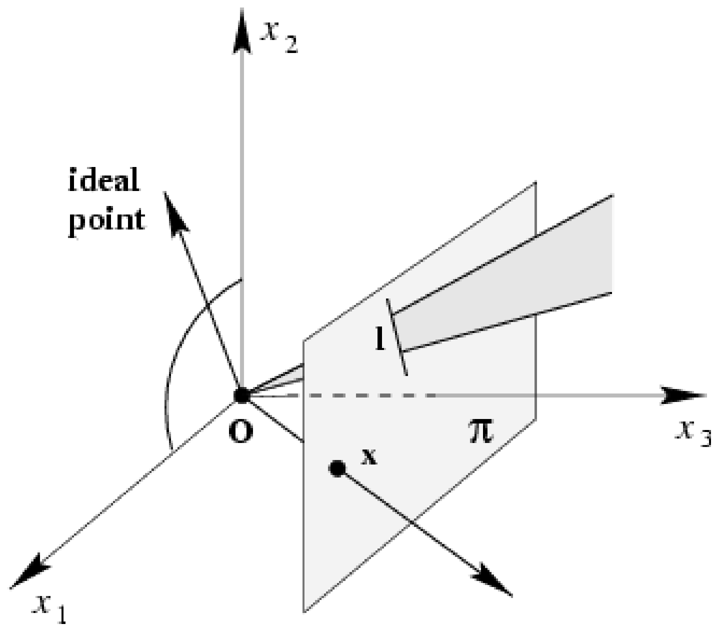
Ideal points $(x_1, x_2, 0)^\mathsf{T}$

Line at infinity $\mathrm{l}_\infty = (0,0,1)^\mathsf{T}$

$\mathbf{P}^2 = \mathbf{R}^2 \cup \mathrm{l}_\infty$

Note that in $\mathbf{P}^2$ there is no distinction between ideal points and others

A model for the projective plane



exactly one line through two points

exactly one point at intersection of two lines

Duality

$$\mathbf{x} \longleftrightarrow \mathbf{l}$$

$$\mathbf{x}^{\mathsf{T}}\mathbf{l} = 0 \longleftrightarrow \mathbf{l}^{\mathsf{T}}\mathbf{x} = 0$$

$$\mathbf{x} = \mathbf{l} \times \mathbf{l}' \longleftrightarrow \mathbf{l} = \mathbf{x} \times \mathbf{x}'$$

Duality principle:

To any theorem of 2-dimensional projective geometry there corresponds a dual theorem, which may be derived by interchanging the role of points and lines in the original theorem

Projective 2D Geometry



Projective transformations

Definition:

A *projectivity* is an invertible mapping h from P^2 to itself such that three points x_1,x_2,x_3 lie on the same line if and only if $h(x_1)$,$h(x_2)$,$h(x_3)$ do.

Theorem:

A mapping h:$P^2 \rightarrow P^2$ is a projectivity if and only if there exist a non-singular 3x3 matrix **H** such that for any point in P^2 reprented by a vector **x** it is true that $h(\mathbf{x})=\mathbf{H}\mathbf{x}$

Definition: Projective transformation

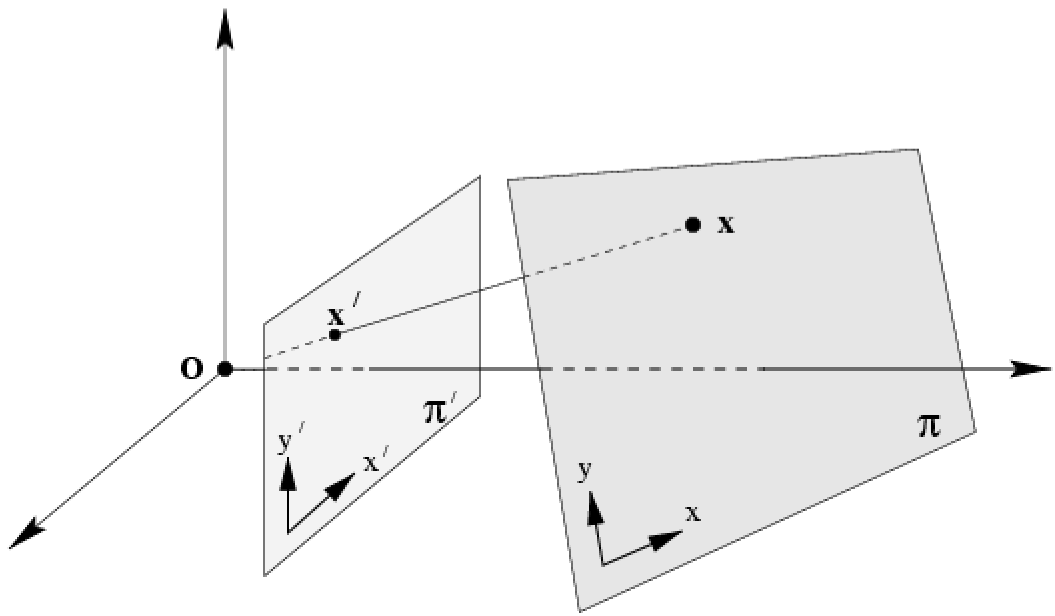
$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

or $\mathbf{x}' = \mathbf{H}\,\mathbf{x}$

8DOF

projectivity=collineation=projective transformation=homography

Mapping between planes



central projection may be expressed by x'=Hx
(application of theorem)

Removing projective distortion

select four points in a plane with know coordinates

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

$$x'\left(h_{31}x + h_{32}y + h_{33}\right) = h_{11}x + h_{12}y + h_{13}$$
$$y'\left(h_{31}x + h_{32}y + h_{33}\right) = h_{21}x + h_{22}y + h_{23}$$

(linear in h_{ij})

(2 constraints/point, 8DOF $\Rightarrow$ 4 points needed)

Appears in: Proceedings of International Conference on Computer Vision, 2001

Smarter Presentations: Exploiting Homography in Camera-Projector Systems

Rahul Sukthankar[1,2], Robert G. Stockton[1], Matthew D. Mullin[1]

[1]Just Research, 4616 Henry Street, Pittsburgh, PA 15213
[2]The Robotics Institute, Carnegie Mellon, Pittsburgh, PA 15213

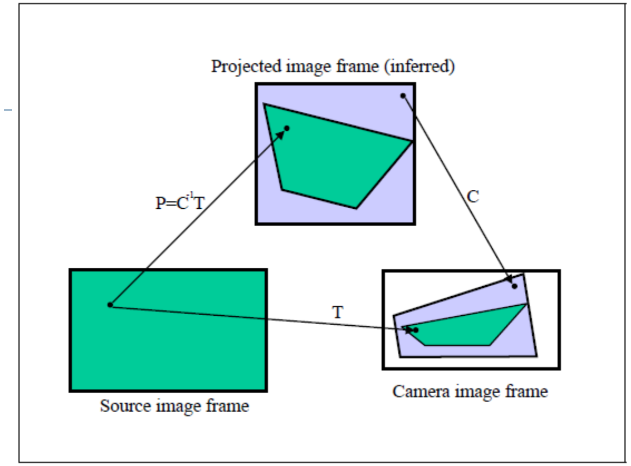
{rahuls,rgs,mdm}@justresearch.com *

At first glance, it may appear that this mapping is impossible to determine in the presence of so many unknowns. Fortunately, we can exploit the fact that all of the observed points in the scene lie on some unknown plane (the flat projection screen), and this establishes a homography between the camera and projector frames of reference. Thus, we can show that the compounded transforms mapping (x, y) in the projector frame, to some unknown point on the projection screen, and then to pixel (X, Y) in the camera frame, can be expressed by a *single* projective transform,

$$(x,y) = \left(\frac{p_1X + p_2Y + p_3}{p_7X + p_8Y + p_9}, \frac{p_4X + p_5Y + p_6}{p_7X + p_8Y + p_9}\right),$$

with eight degrees of freedom, $\vec{p} = (p_1 \dots p_9)^T$ constrained by $|\vec{p}| = 1$. The same transform is more concisely expressed in homogeneous coordinates as:

$$\begin{pmatrix} xw \\ yw \\ w \end{pmatrix} = \begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & p_9 \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$



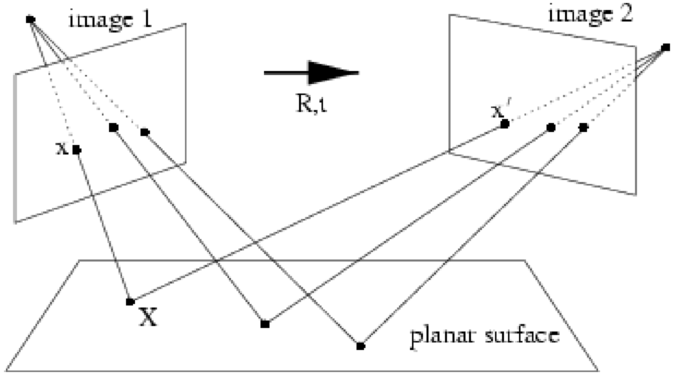
$\vec{p}$ can be determined from as few as four pixel correspondences[2]; when more than four correspondences are available, the system finds the best estimate in a least-squares sense. Given n feature point matches, $\{(x_i, y_i), (X_i, Y_i)\}$,

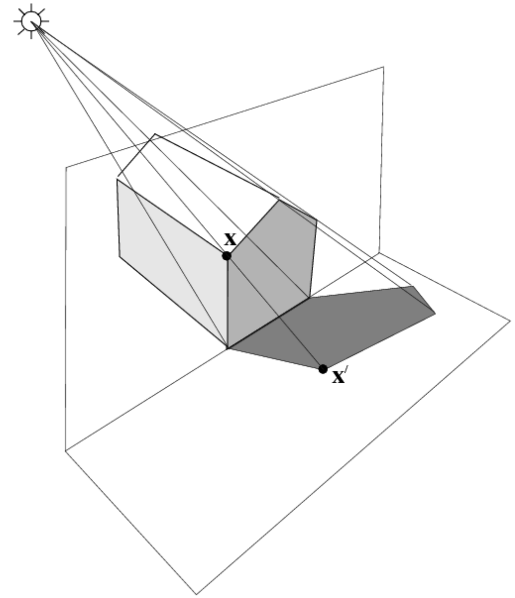
let A be the following $2n \times 9$ matrix:

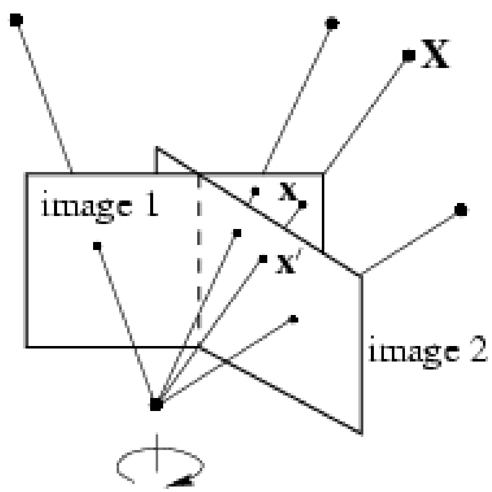
$$\begin{pmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -X_1x_1 & -Y_1x_1 & -x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -X_1y_1 & -Y_1y_1 & -y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -X_2x_2 & -Y_2x_2 & -x_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -X_2y_2 & -Y_2y_2 & -y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & 1 & 0 & 0 & 0 & -X_nx_n & -Y_nx_n & -x_n \\ 0 & 0 & 0 & X_n & Y_n & 1 & -X_ny_n & -Y_ny_n & -y_n \end{pmatrix}$$

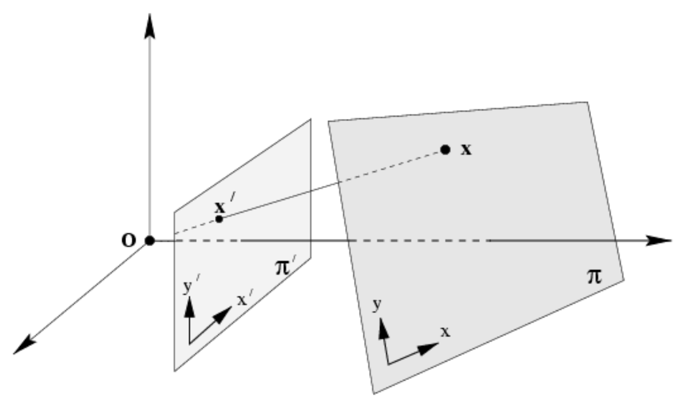
The goal is to find the unit vector $\vec{p}$ that minimizes $|A\vec{p}|$, and this is given by the eigenvector corresponding to the smallest eigenvalue of A^TA.

More Examples









Transformation for lines

For a point transformation

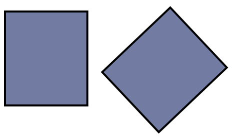
$$\mathrm{x}' = \mathbf{H}\,\mathrm{x}$$

Transformation for lines

$$\mathrm{l}' = \mathbf{H}^{-T}\,\mathrm{l}$$

Isometries

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \varepsilon\cos\theta & -\sin\theta & t_x \\ \varepsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \varepsilon = \pm 1$$



orientation preserving: $\varepsilon = 1$
orientation reversing: $\varepsilon = -1$

$$\mathrm{x}' = \mathbf{H}_E\,\mathrm{x} = \begin{bmatrix} \mathbf{R} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix} \mathrm{x} \qquad \mathbf{R}^{\mathsf{T}}\mathbf{R} = \mathbf{I}$$

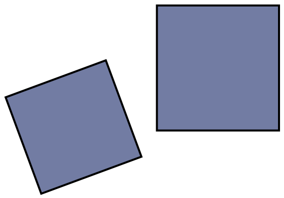
3DOF (1 rotation, 2 translation)

special cases: pure rotation, pure translation

Invariants: length, angle, area

Similarities

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



$$\mathrm{x}' = \mathbf{H}_S\,\mathrm{x} = \begin{bmatrix} s\mathbf{R} & \mathrm{t} \\ 0^\mathsf{T} & 1 \end{bmatrix} \mathrm{x}$$

$$\mathbf{R}^\mathsf{T}\mathbf{R} = \mathbf{I}$$

4DOF (1 scale, 1 rotation, 2 translation)

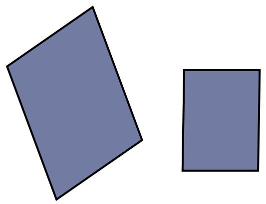
also know as *equi-form* (shape preserving)

metric structure = structure up to similarity (in literature)

Invariants: ratios of length, angle, ratios of areas, parallel lines

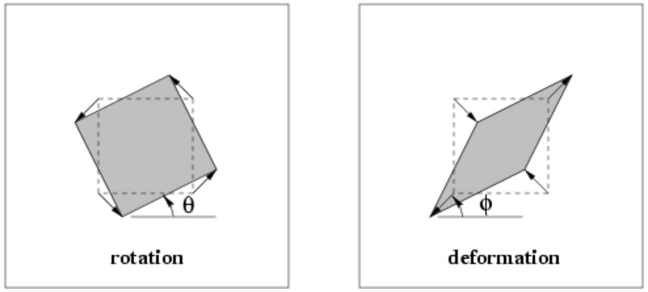
Affine transformations

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



$$\mathrm{x}' = \mathbf{H}_A \, \mathrm{x} = \begin{bmatrix} \mathbf{A} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix} \mathrm{x}$$

$$\mathbf{A} = \mathbf{R}(\theta)\mathbf{R}(-\phi)\mathbf{D}\mathbf{R}(\phi) \qquad \mathbf{D} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



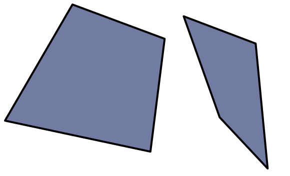
6DOF (2 scale, 2 rotation, 2 translation)

non-isotropic scaling! (2DOF: scale ratio and orientation)

Invariants: parallel lines, ratios of parallel lengths, ratios of areas

Projective transformations

$$x' = \mathbf{H}_P\, x = \begin{bmatrix} \mathbf{A} & t \\ v^{\mathsf{T}} & v \end{bmatrix} x \qquad v = (v_1, v_2)^{\mathsf{T}}$$



8DOF (2 scale, 2 rotation, 2 translation, 2 line at infinity)

Action non-homogeneous over the plane

Invariants: cross-ratio of four points on a line (ratio of ratio)

$$\mathbf{H} = \mathbf{H}_S \mathbf{H}_A \mathbf{H}_P = \begin{bmatrix} s\mathbf{R} & t \\ 0^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{K} & 0 \\ 0^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ v^{\mathsf{T}} & v \end{bmatrix} = \begin{bmatrix} \mathbf{A} & t \\ v^{\mathsf{T}} & v \end{bmatrix}$$

$$\mathbf{A} = s\mathbf{RK} + t v^{\mathsf{T}}$$

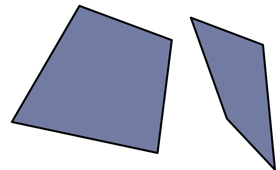
decomposition unique (if chosen s>0)

$\mathbf{K}$ upper-triangular, $\det \mathbf{K} = 1$

Overview Transformations

Projective
8dof

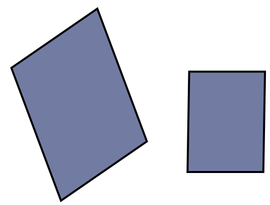
$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$



Concurrency, collinearity, order of contact (intersection, tangency, inflection, etc.), cross ratio

Affine
6dof

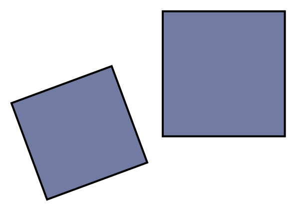
$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



Parallellism, ratio of areas, ratio of lengths on parallel lines (e.g midpoints), linear combinations of vectors (centroids).
The line at infinity l_∞

Similarity
4dof

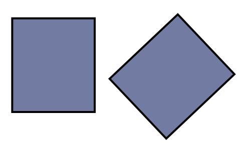
$$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



Ratios of lengths, angles.
The circular points I,J

Euclidean
3dof

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



lengths, areas.

Line at infinity

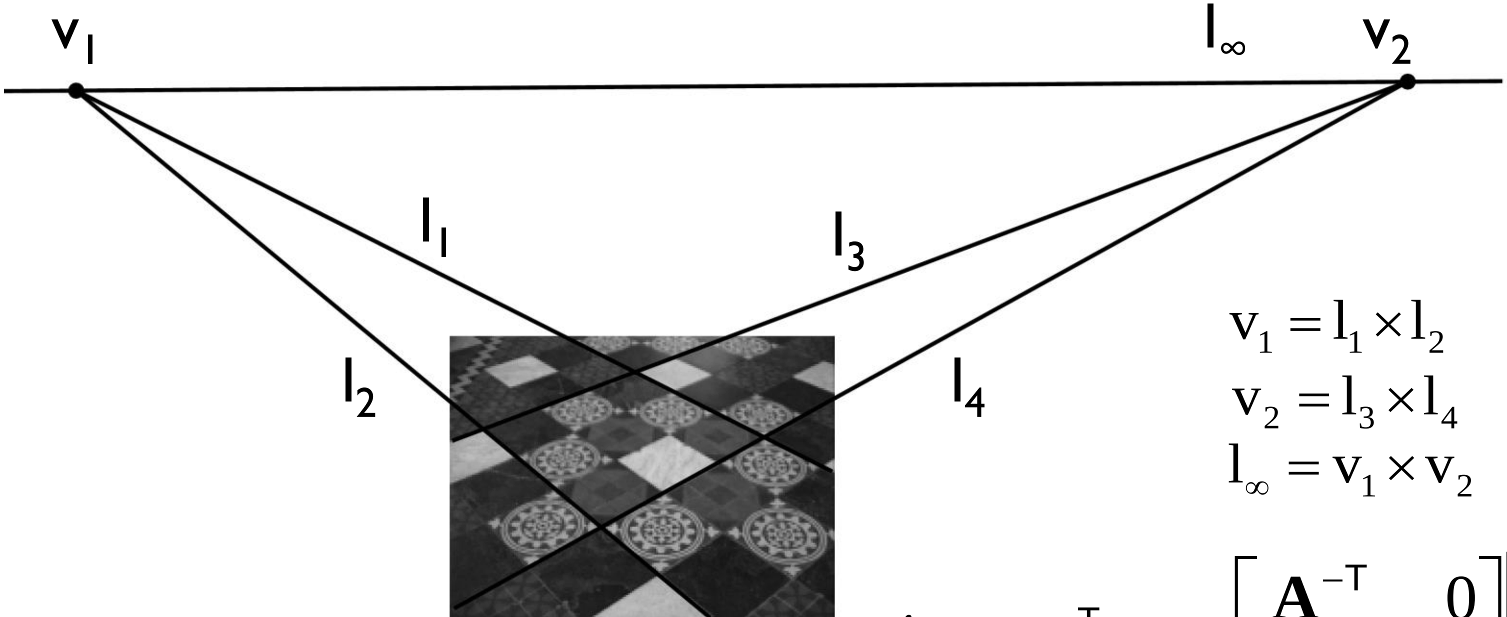
$$\begin{bmatrix} \mathbf{A} & \mathrm{t} \\ 0^{\mathsf{T}} & v \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{A} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ 0 \end{pmatrix}$$

Line at infinity stays at infinity

$$\begin{bmatrix} \mathbf{A} & \mathrm{t} \\ \mathrm{v}^{\mathsf{T}} & v \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} \mathbf{A} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ v_1 x_1 + v_2 x_2 \end{pmatrix}$$

Line at infinity becomes finite,
allows to observe vanishing points, horizon,

The line at infinity

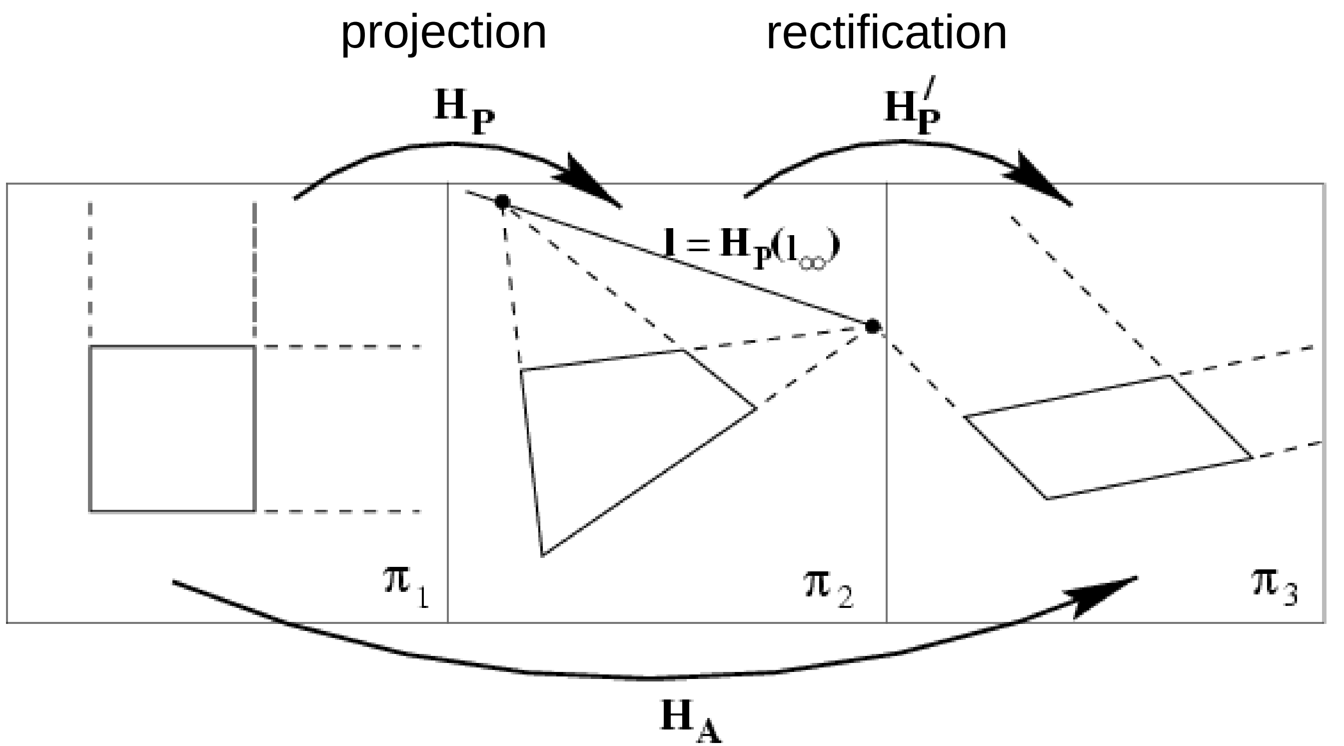


$$\mathbf{v}_1 = \mathbf{l}_1 \times \mathbf{l}_2$$
$$\mathbf{v}_2 = \mathbf{l}_3 \times \mathbf{l}_4$$
$$\mathbf{l}_\infty = \mathbf{v}_1 \times \mathbf{v}_2$$

$$\mathbf{l}'_\infty = \mathbf{H}_A^{-\mathsf{T}} \mathbf{l}_\infty = \begin{bmatrix} \mathbf{A}^{-\mathsf{T}} & 0 \\ -\mathbf{A}\,\mathbf{t} & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \mathbf{l}_\infty$$

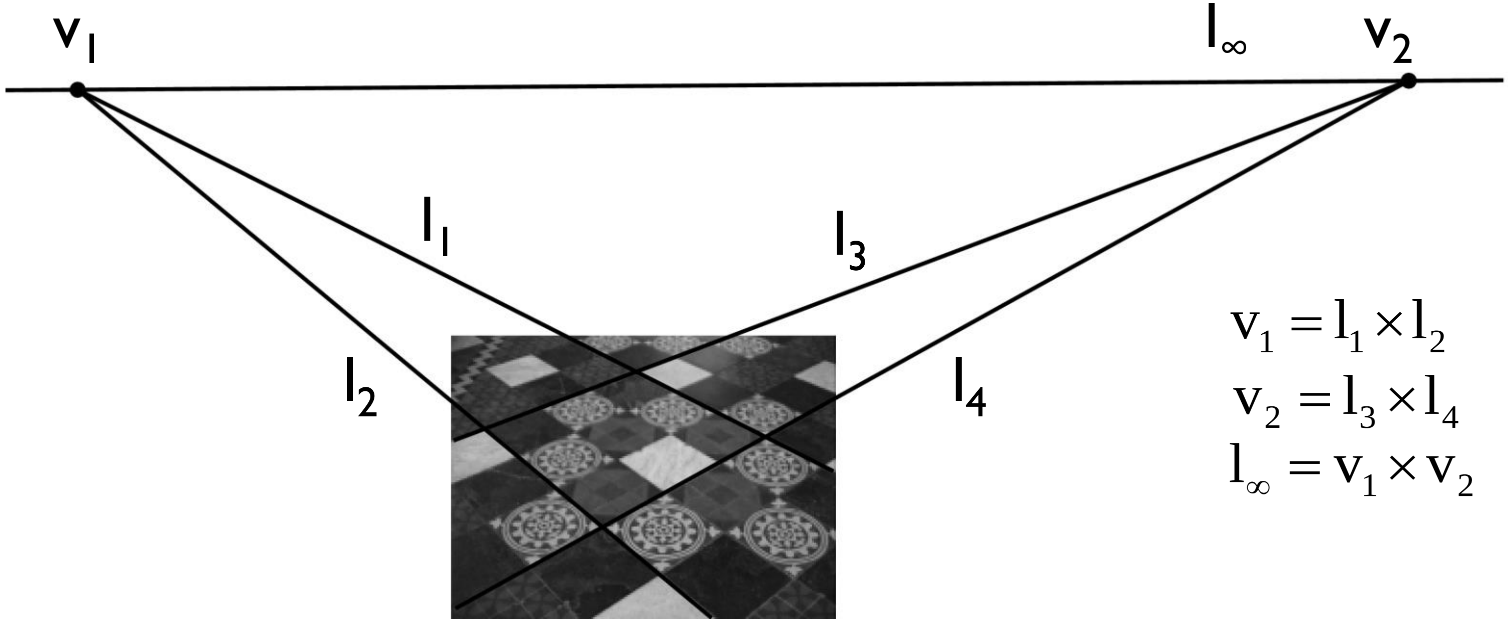
The line at infinity l_∞ is a fixed line under a projective transformation H if and only if H is an affinity

Affine properties from images

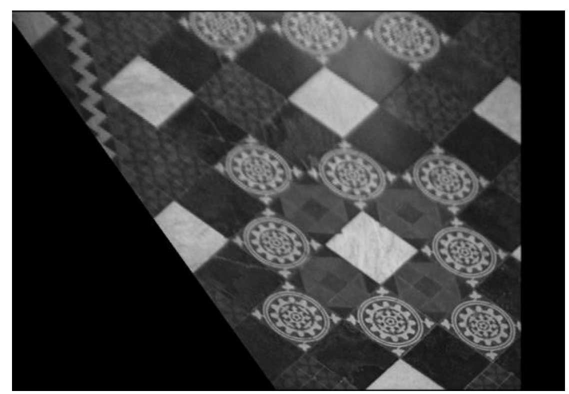


$$\mathbf{H}_{PA} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix} \mathbf{H}_A \qquad \mathrm{l}_\infty = \begin{bmatrix} l_1 & l_2 & l_3 \end{bmatrix}^\mathrm{T}, l_3 \neq 0$$

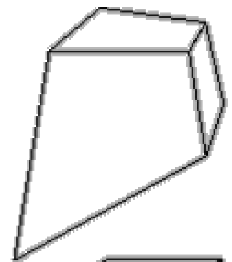
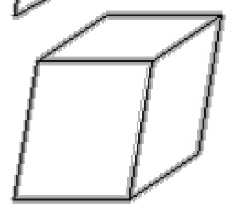
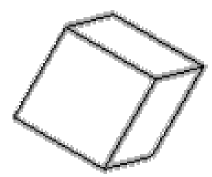
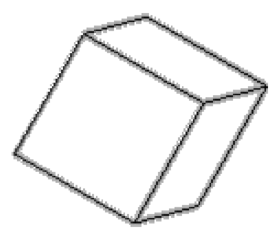
Affine Rectification

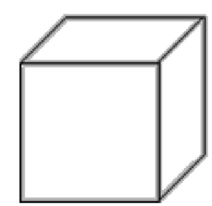


$$v_1 = l_1 \times l_2$$
$$v_2 = l_3 \times l_4$$
$$l_\infty = v_1 \times v_2$$



Projective 3D geometry

Projective 15dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		Intersection and tangency
Affine 12dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Parallellism of planes, Volume ratios, centroids, **The plane at infinity π_∞**
Similarity 7dof	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$		**The absolute conic Ω_∞**
Euclidean 6dof	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Volume



Camera Calibration

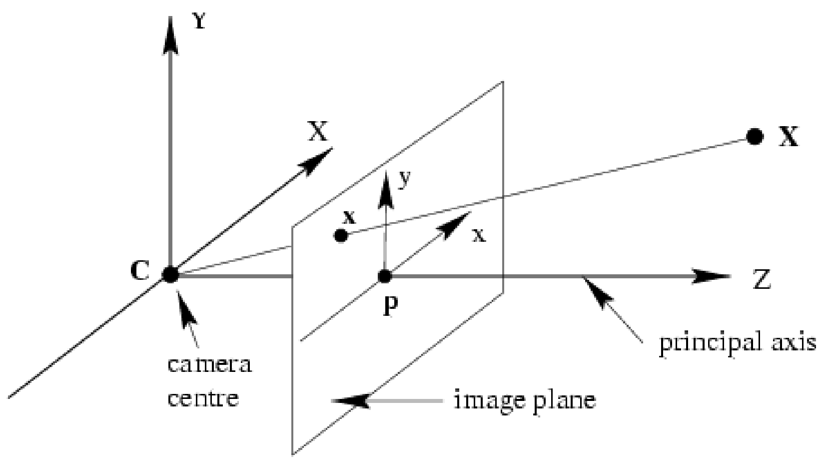
lbg@dongseo.ac.kr

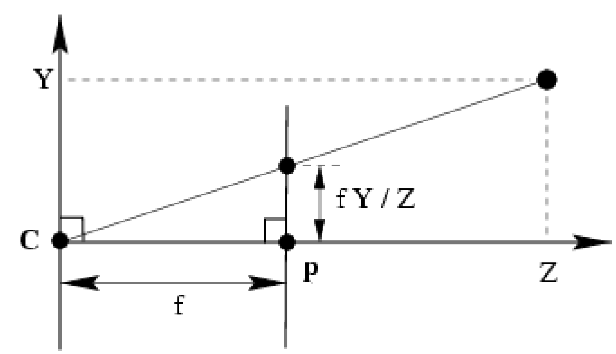
Pinhole Camera Model

$$(X,Y,Z)^T \mapsto (fX/Z, fY/Z)^T$$

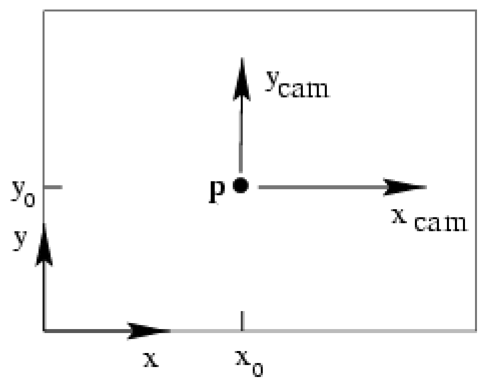
$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$





Pinhole Camera Model



Principal point offset

$$\mathrm{x} = \mathrm{K}[\mathrm{I} \mid 0]\mathrm{X}_{\mathrm{cam}}$$

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

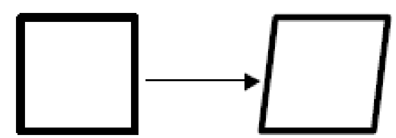
Calibration Matrix

Internal Camera Parameters

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix} \quad \text{with} \quad \begin{matrix} \alpha_x = f\,k_x \\ \alpha_y = -f\,k_y \end{matrix} \qquad \begin{matrix} x_{pix} = u'/w' \\ y_{pix} = v'/w' \end{matrix}$$

$$\begin{bmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \mathbf{K}\left[\mathbf{I}_3 \quad | \quad \mathbf{0}_3\right]$$

- α_x and α_y "focal lengths" in pixels
- x_0 and y_0 coordinates of image center in pixels
- Added parameter s is skew parameter



- **K** is called *calibration matrix.* **Five degrees of freedom.**
  - **K** is a 3x3 upper triangular matrix

Camera rotation and translation

$$\tilde{X}_{cam} = R(\tilde{X} - \tilde{C})$$

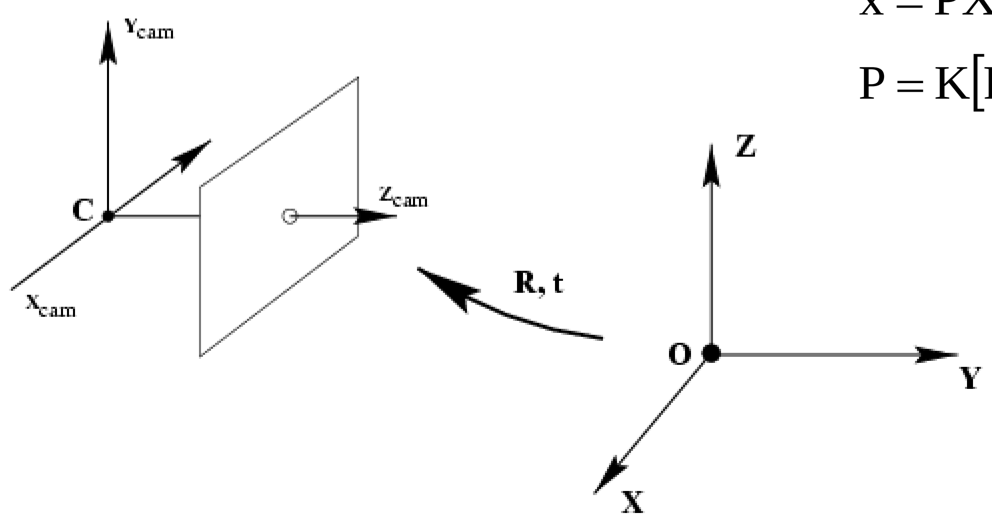
$$X_{cam} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -RC \\ 0 & 1 \end{bmatrix} X$$

$$x = K[I \mid 0]X_{cam}$$

$$x = KR[I \mid -\tilde{C}]X$$

$$x = PX$$

$$P = K[R \mid t] \quad t = -R\tilde{C}$$



Camera Parameter Matrix P

- Further simplification of **P**:

$$\mathbf{x} = \mathbf{K}[\mathbf{I}_3 \mid \mathbf{0}_3]\begin{bmatrix} \mathbf{R} & -\mathbf{R}\widetilde{\mathbf{C}} \\ \mathbf{0}_3^T & 1 \end{bmatrix}\mathbf{X}$$

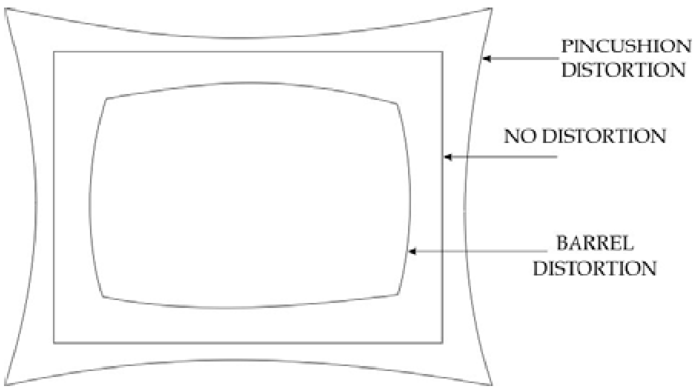
$$[\mathbf{I}_3 \mid \mathbf{0}_3]\begin{bmatrix} \mathbf{R} & -\mathbf{R}\widetilde{\mathbf{C}} \\ \mathbf{0}_3^T & 1 \end{bmatrix} = [\mathbf{R} \quad -\mathbf{R}\widetilde{\mathbf{C}}] = \mathbf{R}[\mathbf{I}_3 \mid -\widetilde{\mathbf{C}}]$$

$$\mathbf{x} = \mathbf{K}\mathbf{R}[\mathbf{I}_3 \mid -\widetilde{\mathbf{C}}]\mathbf{X}$$

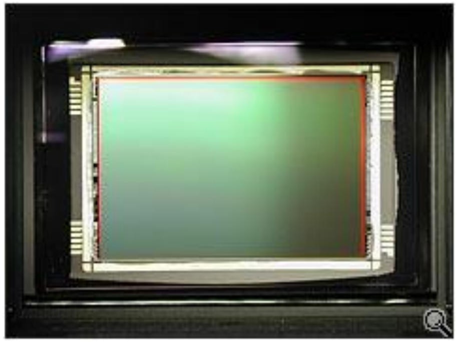
$$\boxed{\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}_3 \mid -\widetilde{\mathbf{C}}]}$$

- **P** has 11 degrees of freedom:
  - 5 from triangular calibration matrix **K**, 3 from **R** and 3 from $\widetilde{\mathbf{C}}$
- **P** is a fairly general 3 x 4 matrix
  - left 3x3 submatrix **KR** is non-singular

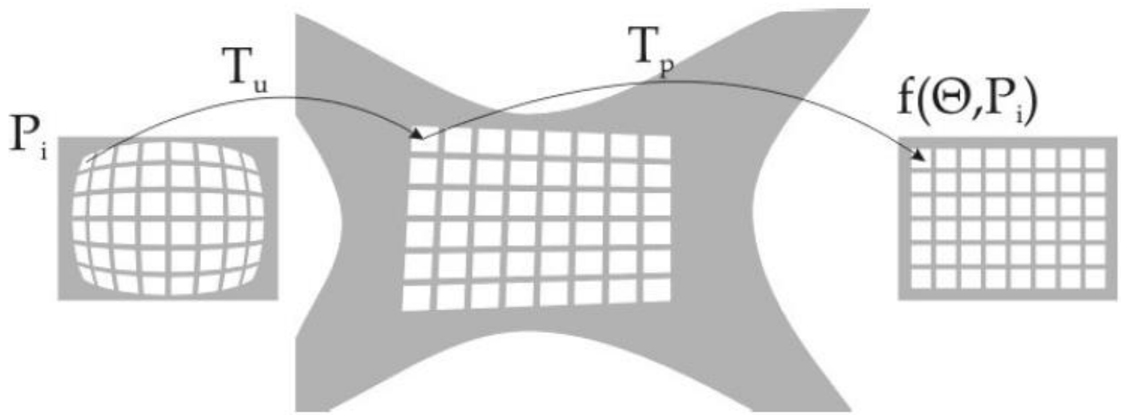
CCD Cameras


PINCUSHION DISTORTION
NO DISTORTION
BARREL DISTORTION


element CCD



Correcting Radial Distortion of Cameras



$$
\begin{aligned}
x_u &= c_x + (x_d - c_x)\, f_2(r_d^2) \\
&= c_x + (x_d - c_x)(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6) \\
y_u &= c_y + (y_d - c_y)\, f_2(r_d^2) \\
&= c_y + (y_d - c_y)(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6) \\
r_d^2 &= (x_d - c_x)^2 + (y_d - c_y)^2
\end{aligned}
$$

$$
x_p = \frac{m_0 x_u + m_1 y_u + m_2}{m_6 x_u + m_7 y_u + 1}
$$

$$
y_p = \frac{m_3 x_u + m_4 y_u + m_5}{m_6 x_u + m_7 y_u + 1}
$$

Correcting Radial Distortion of Cameras with Wide Angle Lens Using Point Correspondences

Leonardo Romero and Cuauhtemoc Gomez
División de Estudios de Postgrado, Facultad de Ingeniería Eléctrica
Universidad Michoacana de San Nicolás de Hidalgo
Morelia, Michoacán, México

$$e_{xk} = x_p(\Theta, x_{dk}, y_{dk}) - x_{rk}$$
$$e_{yk} = y_p(\Theta, x_{dk}, y_{dk}) - y_{rk}$$
$$E(\Theta) = \sum_{k=1}^{n} \left(e_{xk}^{\ 2} + e_{yk}^{\ 2} \right)$$
$$\Theta = \operatorname{argmin} E(\Theta)$$

5.1 Non-Linear Optimization

The Gauss-Newton-Levenberg-Marquardt method (GNLM) (Press et al., 1986) is a non-linear iterative technique specifically designated for minimizing functions which has the form of sum of square functions, like E. At each iteration the increment of parameters, vector $\delta\Theta$, is computed solving the following linear matrix equation:

$$[A+\lambda I]\delta\Theta = B \tag{8}$$

If there is n point correspondences and q parameters in Θ, A is a matrix of dimension qxq and matrix B has dimension $qx1$ and $\delta\Theta = [\,\delta\theta_1, \delta\theta_1, \ldots, \delta\theta_q\,]^t$. λ is a parameter which is allowed to vary at each iteration. After a little algebra, the elements of A and B can be computed using the following formulas,

$$a_{i,j} = \sum_{k=1}^{n} \left(\frac{\partial x_{pk}}{\partial \theta_i} \frac{\partial x_{pk}}{\partial \theta_j} + \frac{\partial y_{pk}}{\partial \theta_i} \frac{\partial y_{pk}}{\partial \theta_j} \right) \quad b_i = -\sum_{k=1}^{n} \left(\frac{\partial x_{pk}}{\partial \theta_i} e_{xk} + \frac{\partial y_{pk}}{\partial \theta_i} e_{yk} \right)$$

Calibration Process

5.4 The Calibration Process

The calibration process starts with one image from the camera, I_d, another image from the calibration pattern, I_r, and initial values for parameters Θ. In the following algorithm, Θ and $\delta\Theta$ are considered as vectors. We start with (c_x,c_y) at the center of the image, $k_1=k_2=k_3=0$ and the identity matrix for M. The calibration algorithm is as follows:

1. From the reference image, compute the reference feature points (x_{rk},y_{rk}), $(k=1,...n)$.
2. From Θ and the distorted image, compute a corrected image.
3. From the corrected image compute the set of feature points (x_{pk},y_{pk}), $(k=1,...n)$.
4. From $(x_{pk},y_{pk})(k=1,...n)$ and Θ compute $(x_{dk},y_{dk})(k=1,...n)$.
5. Find the best Θ that minimize E using the GNLM algorithm:
   (a) Compute the total error, E(Θ) (eq. 7).
   (b) Pick a modest value for λ, say λ=0.001.
   (c) Solve the linear system of equations (8), and calculate $E(\Theta+\delta\Theta)$.
   (d) If $E(\Theta+\delta\Theta)$ >= E(Θ), increase λ by a factor of 10, and go to the previous step. If λ grows very large, it means that there is no way to improve the solution Θ.
   (e) If $E(\Theta+\delta\Theta)$ < E(Θ), decrease λ by a factor of 10, replace Θ by $\Theta+\delta\Theta$, and go to step 5a.
6. Repeat steps 2-5 until E(Θ) does not decrease.

When $\lambda=0$, the GNLM method is a Gauss-Newton method, and when λ tends to infinity, δΘ turns to so called steepest descent direction and the size of $\delta\theta_i$ tends to zero.

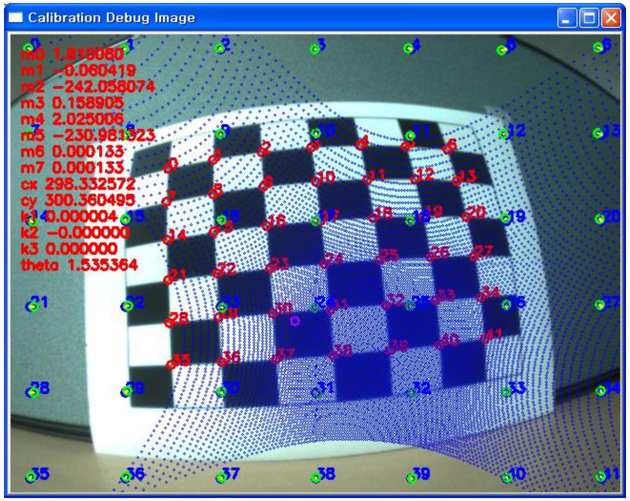
The calibration algorithm apply several times the GNLM algorithm to get better solutions. At the beginning, the clusters of the distorted image are not perfect squares and so point features can not match exactly the feature points computed using the reference image. Once a corrected image is ready, point features can be better estimated.

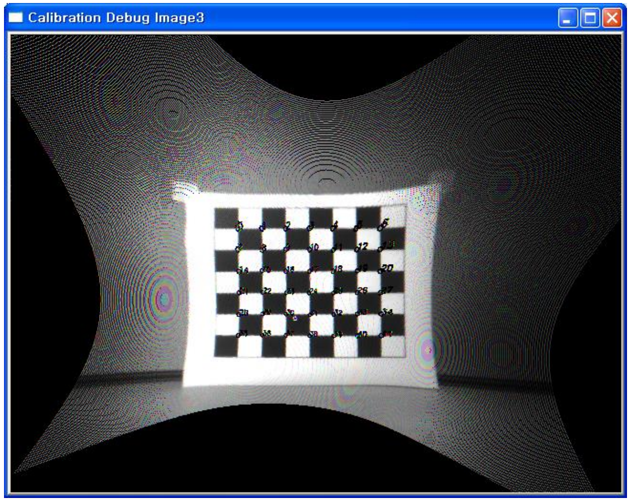
$$x_u = c_x + (x_d - c_x)(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6)\cos\theta - (y_d - c_y)(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6)\sin\theta$$

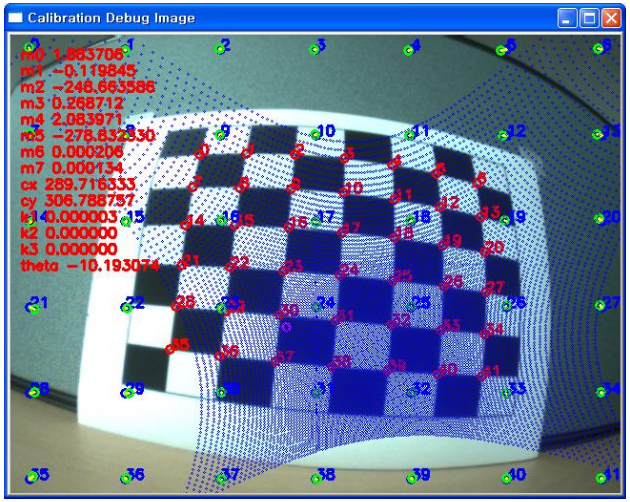
$$y_u = c_y + (x_d - c_x)(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6)\sin\theta + (y_d - c_y)(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6)\cos\theta$$

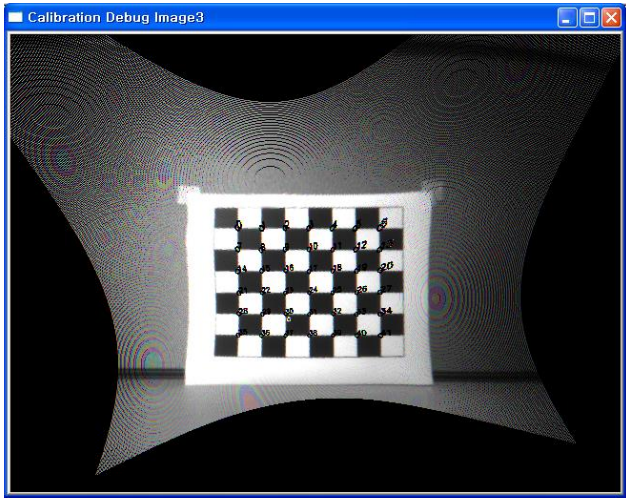
$$x_p = \frac{m_0 x_u + m_1 y_u + m_2}{m_6 x_u + m_7 y_u + 1}$$

$$y_p = \frac{m_3 x_u + m_4 y_u + m_5}{m_6 x_u + m_7 y_u + 1}$$



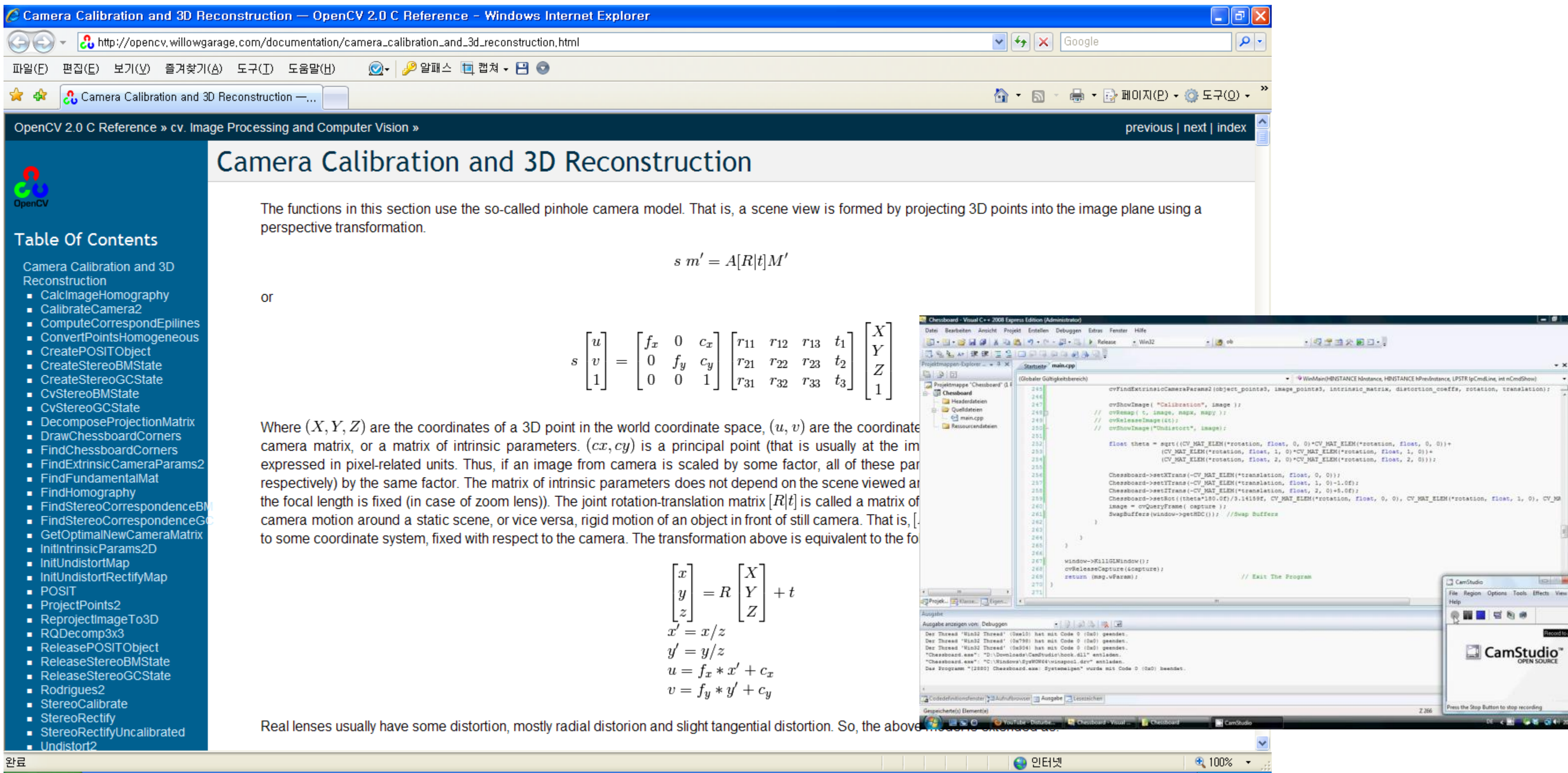






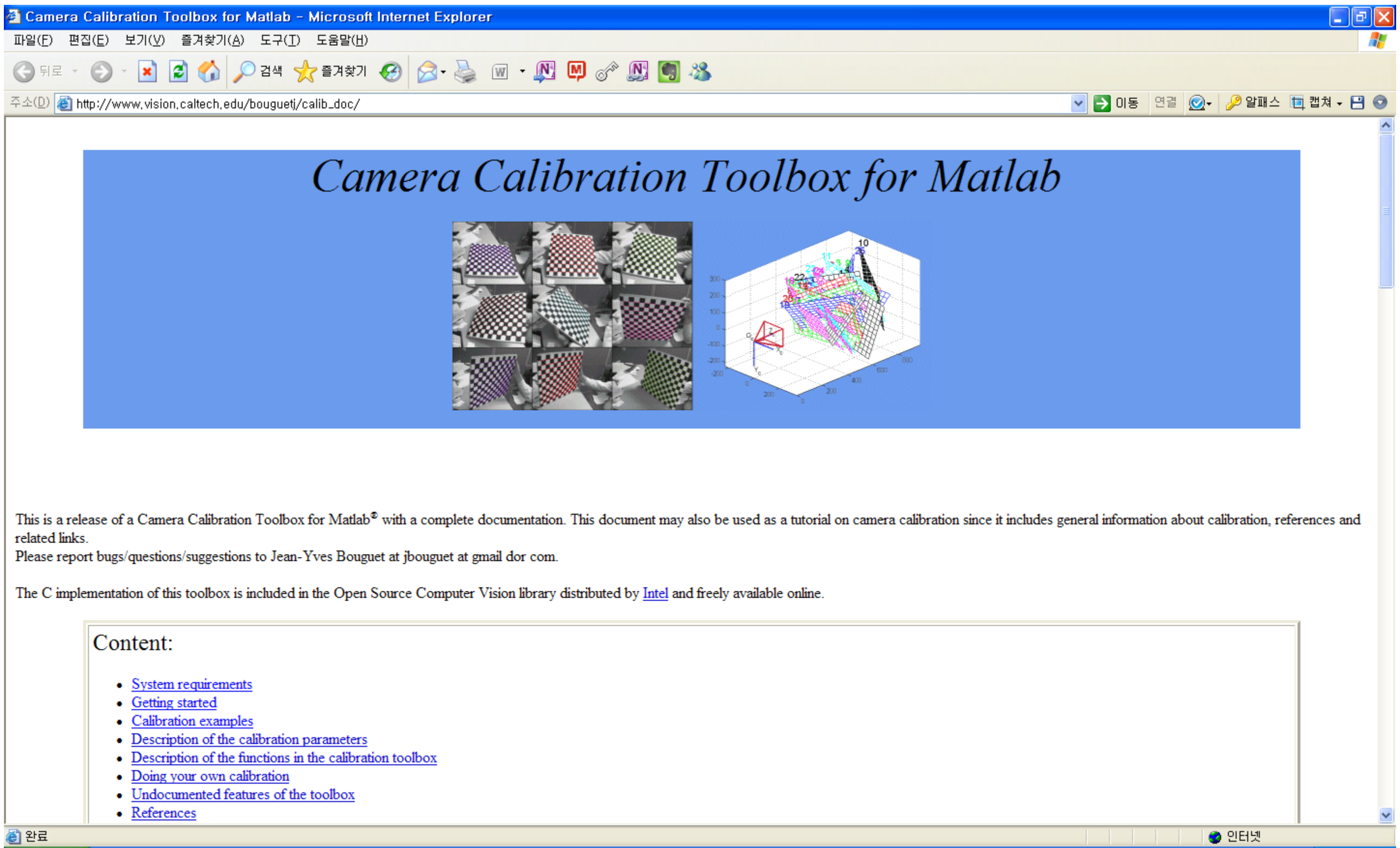
OpenCV

- http://opencv.willowgarage.com/documentation/camera_calibration_and_3d_reconstruction.html

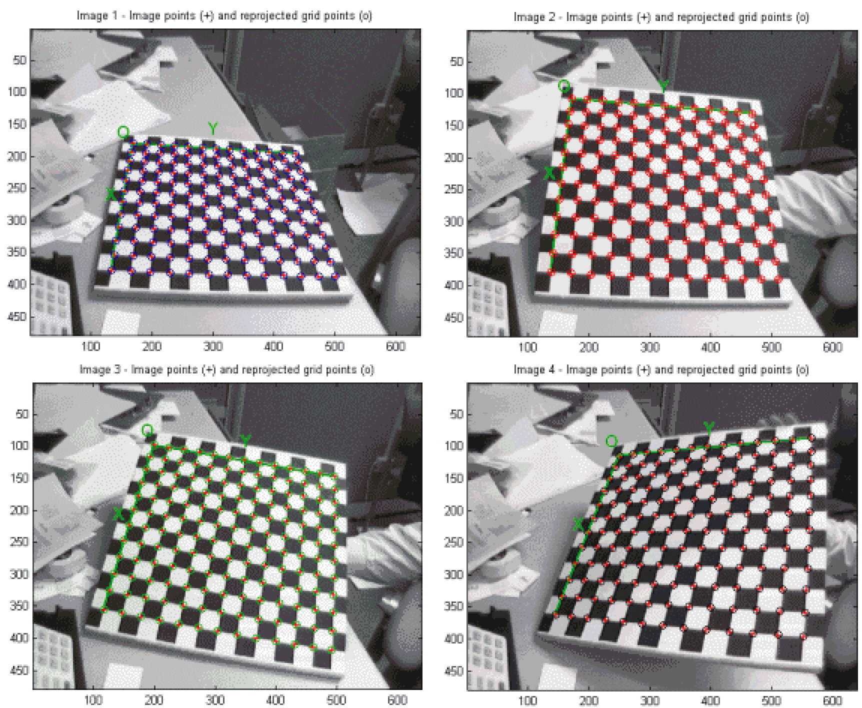


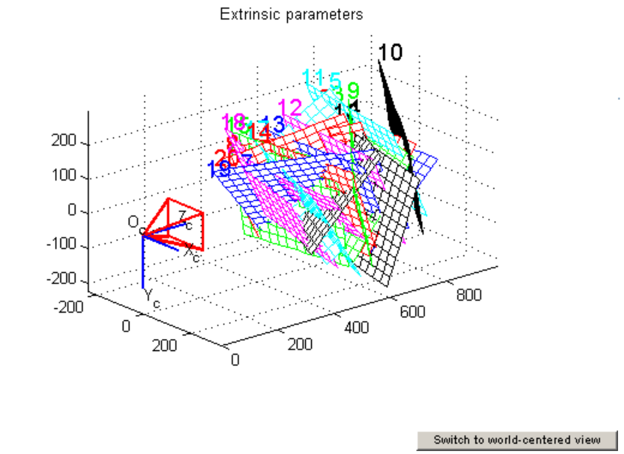
Camera Calibration Toolbox

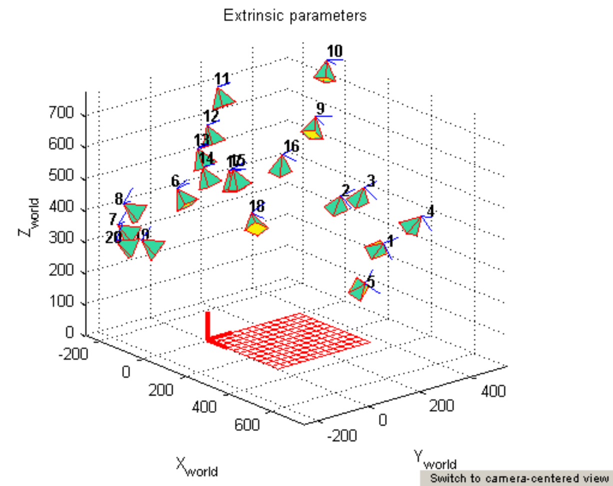
- http://www.vision.caltech.edu/bouguetj/calib_doc/



Camera Calibration

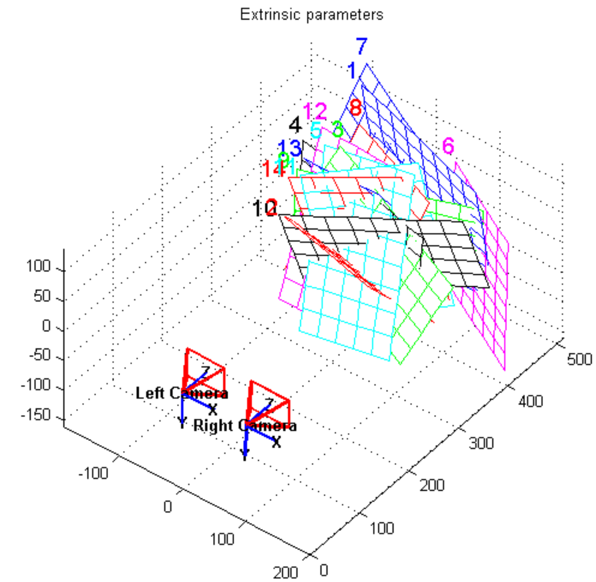







```
Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 661.67001   662.82858 ] ± [ 1.17913   1.26567 ]
Principal point:       cc = [ 306.09590   240.78987 ] ± [ 2.38443   2.17481 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.26425   0.22645   0.00020   0.00023   0.00000 ] ± [ 0.00934   0.03826   0.00052   0.00053   0.00000 ]
Pixel error:          err = [ 0.45330   0.38916 ]
```

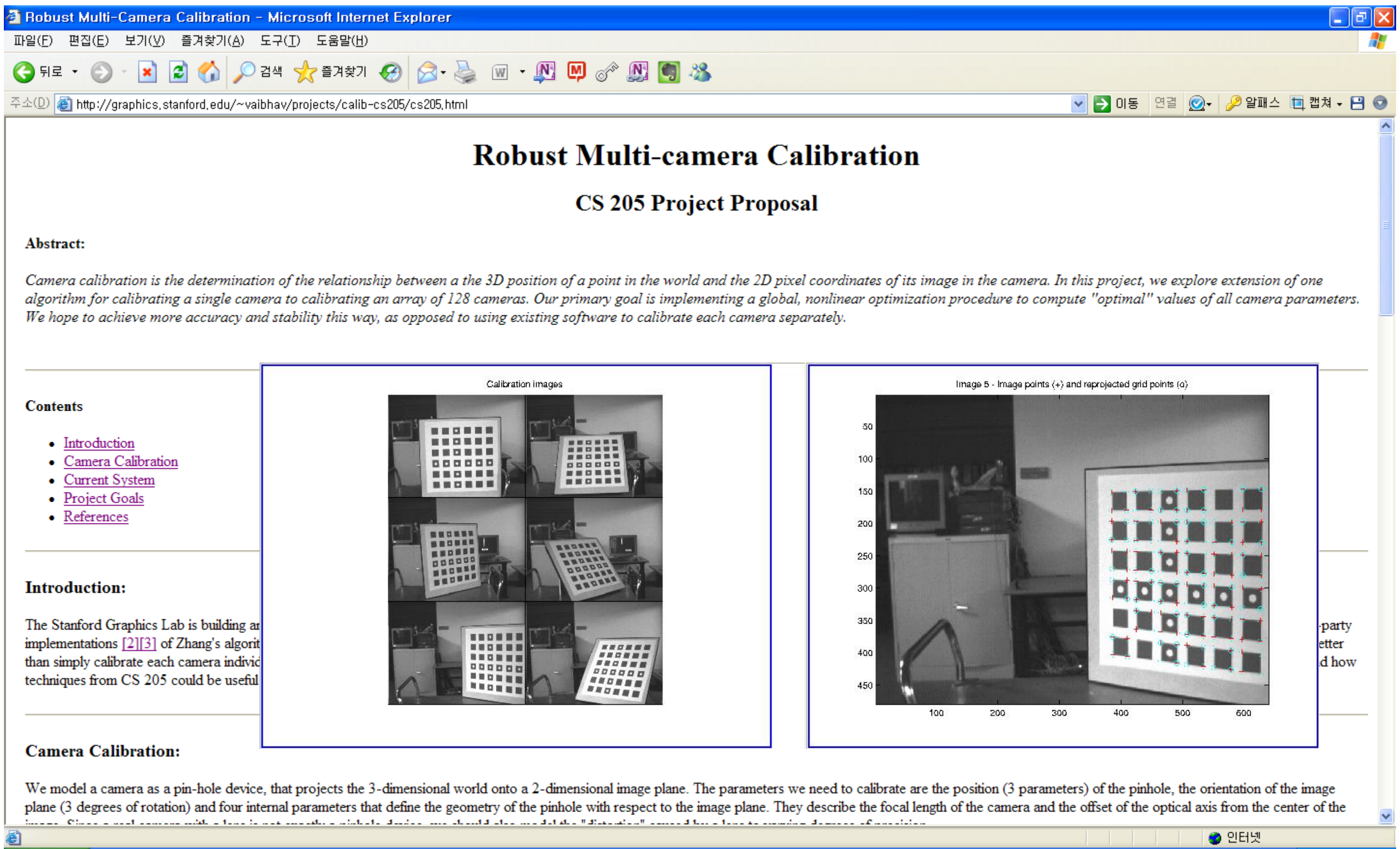
Calibrating a Stereo System




```
Stereo calibration parameters after loading the individual calibration files:


Intrinsic parameters of left camera:

Focal Length:          fc_left = [ 533.00371   533.15260 ] ± [ 1.07629   1.10913 ]
Principal point:       cc_left = [ 341.58612   234.25940 ] ± [ 1.24041   1.33065 ]
Skew:             alpha_c_left = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc_left = [ -0.28947   0.10326   0.00103   -0.00029  0.00000 ] ± [ 0.00596   0.02055   0.00030   0.00037  0.00000 ]


Intrinsic parameters of right camera:

Focal Length:          fc_right = [ 536.98262   536.56938 ] ± [ 1.19786   1.15677 ]
Principal point:       cc_right = [ 326.47209   249.33257 ] ± [ 1.36588   1.34252 ]
Skew:             alpha_c_right = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes = 90.00000 ± 0.00000 degrees
Distortion:            kc_right = [ -0.28936   0.10677   -0.00078   0.00020  0.00000 ] ± [ 0.00488   0.00866   0.00027   0.00062  0.00000 ]


Extrinsic parameters (position of right camera wrt left camera):

Rotation vector:             om = [ 0.00611   0.00409  -0.00359 ]
Translation vector:           T = [ -99.84929   0.82221   0.43647 ]
```

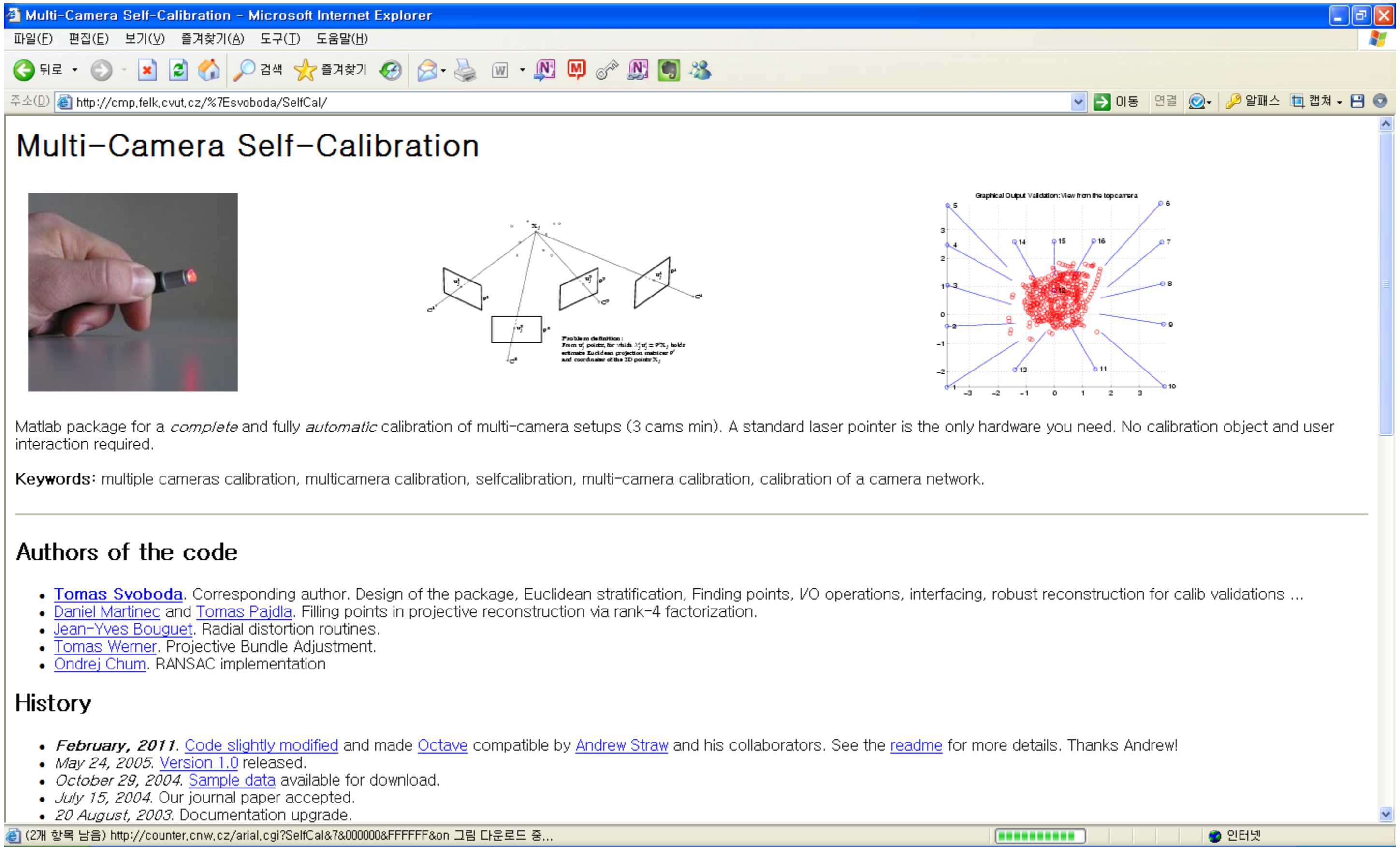
Robust Multi-camera Calibration

- http://graphics.stanford.edu/~vaibhav/projects/calib-cs205/cs205.html



Multi-Camera Self Calibration

- http://cmp.felk.cvut.cz/~Esvoboda/



Multi-Camera Self-Calibration – Microsoft Internet Explorer

http://cmp.felk.cvut.cz/%7Esvoboda/SelfCal/

Multi–Camera Self–Calibration

Matlab package for a *complete* and fully *automatic* calibration of multi–camera setups (3 cams min). A standard laser pointer is the only hardware you need. No calibration object and user interaction required.

Keywords: multiple cameras calibration, multicamera calibration, selfcalibration, multi–camera calibration, calibration of a camera network.

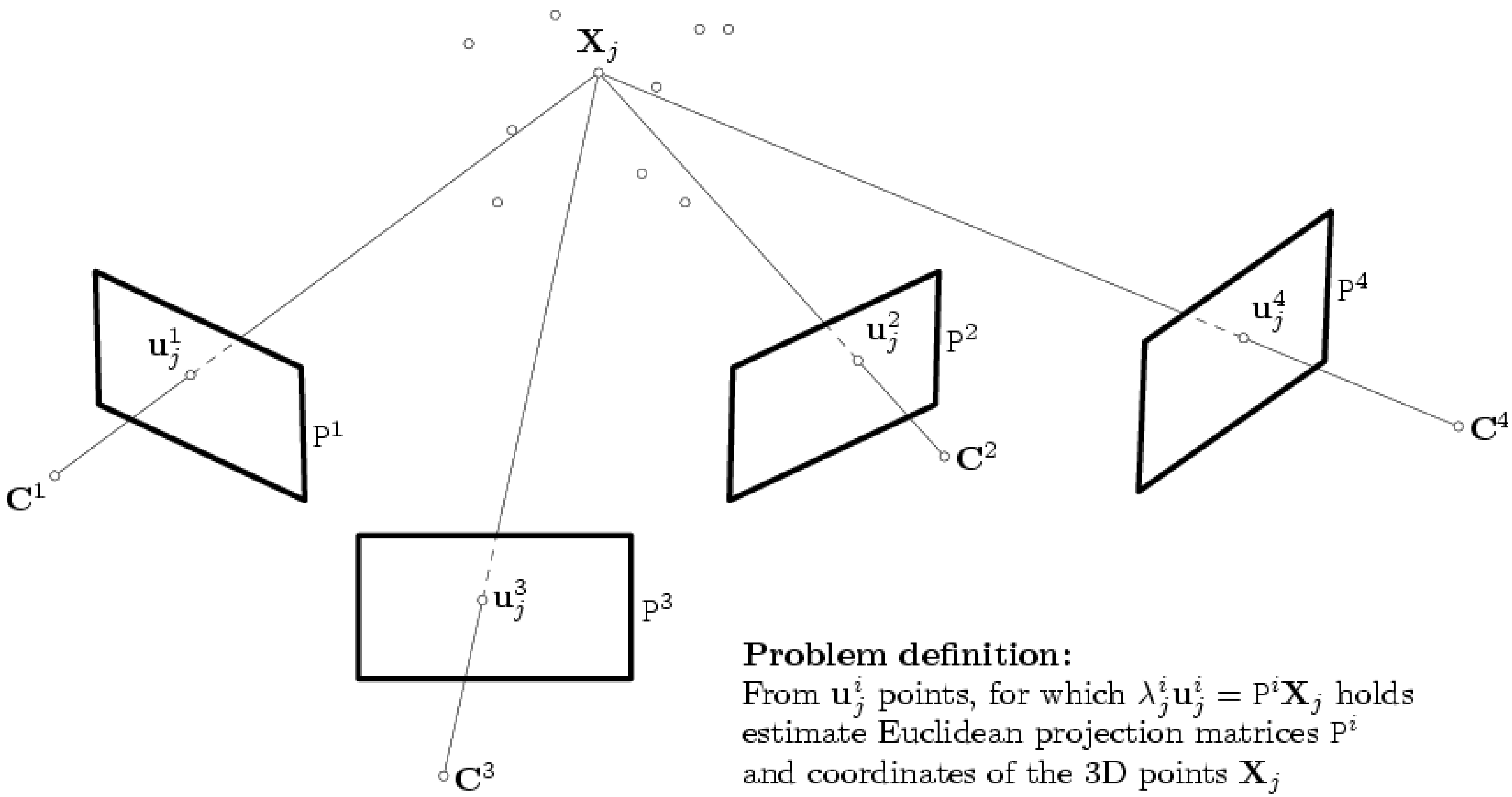
Authors of the code

- Tomas Svoboda. Corresponding author. Design of the package, Euclidean stratification, Finding points, I/O operations, interfacing, robust reconstruction for calib validations ...
- Daniel Martinec and Tomas Pajdla. Filling points in projective reconstruction via rank–4 factorization.
- Jean–Yves Bouguet. Radial distortion routines.
- Tomas Werner. Projective Bundle Adjustment.
- Ondrej Chum. RANSAC implementation

History

- ***February, 2011***. Code slightly modified and made Octave compatible by Andrew Straw and his collaborators. See the readme for more details. Thanks Andrew!
- *May 24, 2005*. Version 1.0 released.
- *October 29, 2004*. Sample data available for download.
- *July 15, 2004*. Our journal paper accepted.
- *20 August, 2003*. Documentation upgrade.

Multi-Camera Self Calibration



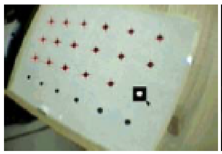
Problem definition:
From $\mathbf{u}_j^i$ points, for which $\lambda_j^i \mathbf{u}_j^i = P^i \mathbf{X}_j$ holds
estimate Euclidean projection matrices P^i
and coordinates of the 3D points $\mathbf{X}_j$

ARToolKit Camera Calibration

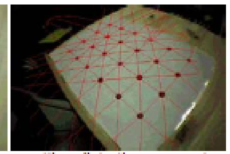
Accurate two-step method (2/3)

17

- Step 1: Getting distortion parameters: 'calib_dist'



selecting dots with mouse



getting distortion parameters by automatic line-fitting

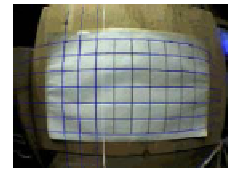
  - Take pattern pictures as large as possible
  - Slant in various directions with big angle
  - 4 times or more

Augmented Reality - Spring 2007 Univ. of Incheon, CSE

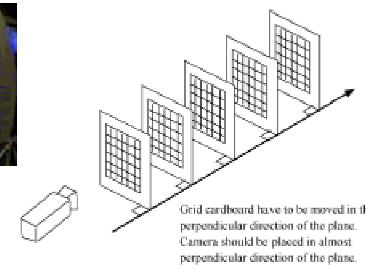
Accurate two-step method (3/3)

18

- Step 2: Getting perspective projection matrix: 'calib_cparam'



Manual line-fitting



Augmented Reality - Spring 2007 Univ. of Incheon, CSE

Easy one-step method: 'calib_camera2'

19

- Same operation as 'calib_dist'
- Getting all camera parameters including distortion parameters and perspective projection matrix
- Not require careful setup
- Accuracy is good enough for image overlay
  - [But, Not good enough for 3D measurement.]

Camera Parameter Implementation

20

- Camera parameter structure

```
typedef struct {
  int xsize, ysize;
  double mat[3][4];
  double dist_factor[4];
} ARParam;
```

- Adjust camera parameter for the input image size

```
int arParamChangeSize(ARParam *source,
  int xsize, int ysize, ARParam *newparam);
```

- Read camera parameters from the file

```
int arParamLoad(char *filename, int num, ARParam *param, …);
```

Epipolar Geometry and 3D Reconstruction.

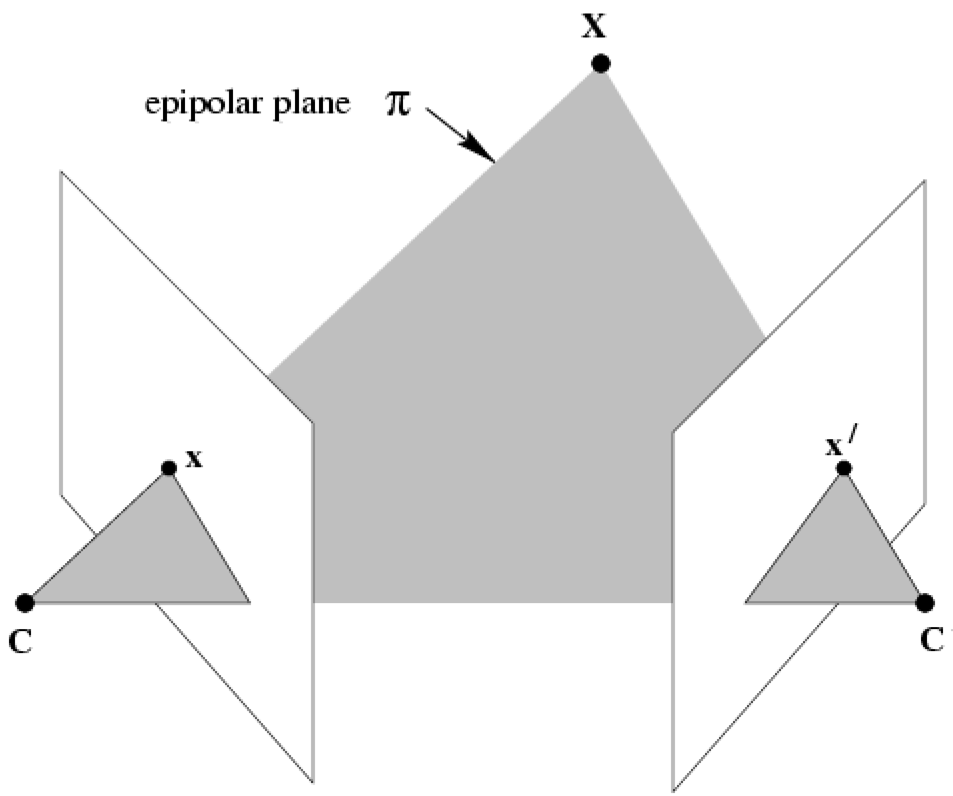
lbg@dongseo.ac.kr

Introduction

Computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras. Computer vision is, in some ways, the inverse of computer graphics. While computer graphics produces image data from 3D models, computer vision often produces 3D models from image data. Today one of the major problems in Computer vision is **correspondence search**.

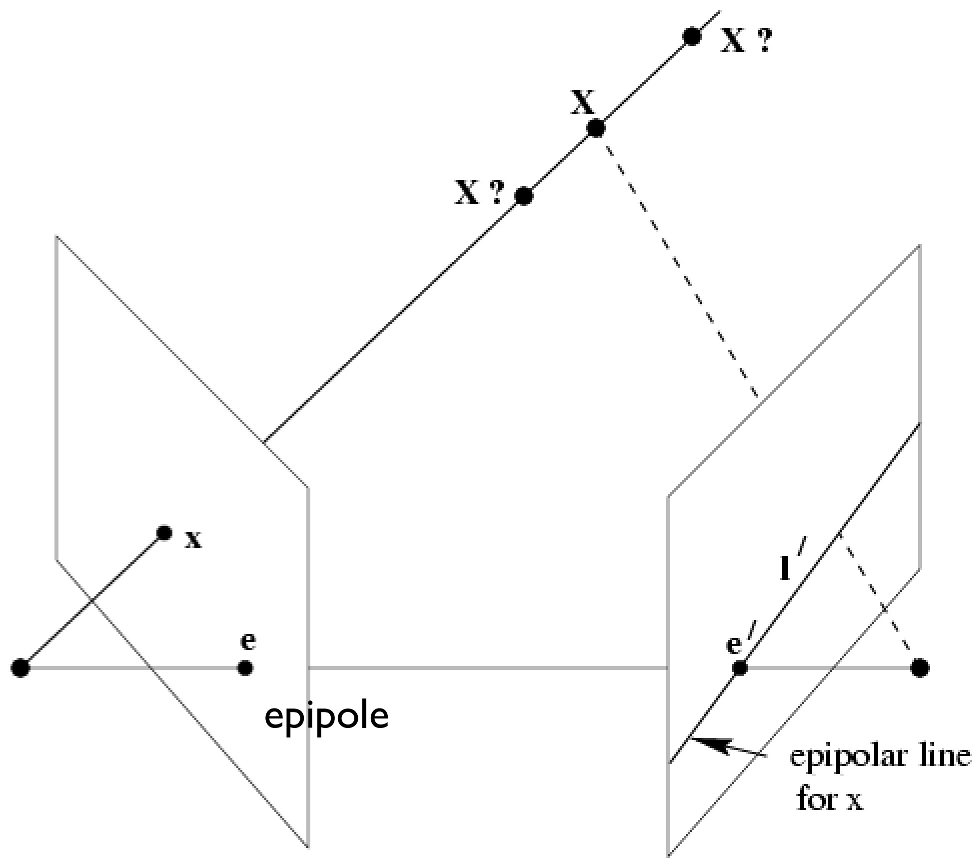
Epipolar Geometry

- http://www.ai.sri.com/~luong/research/Meta3DViewer/EpipolarGeo.html

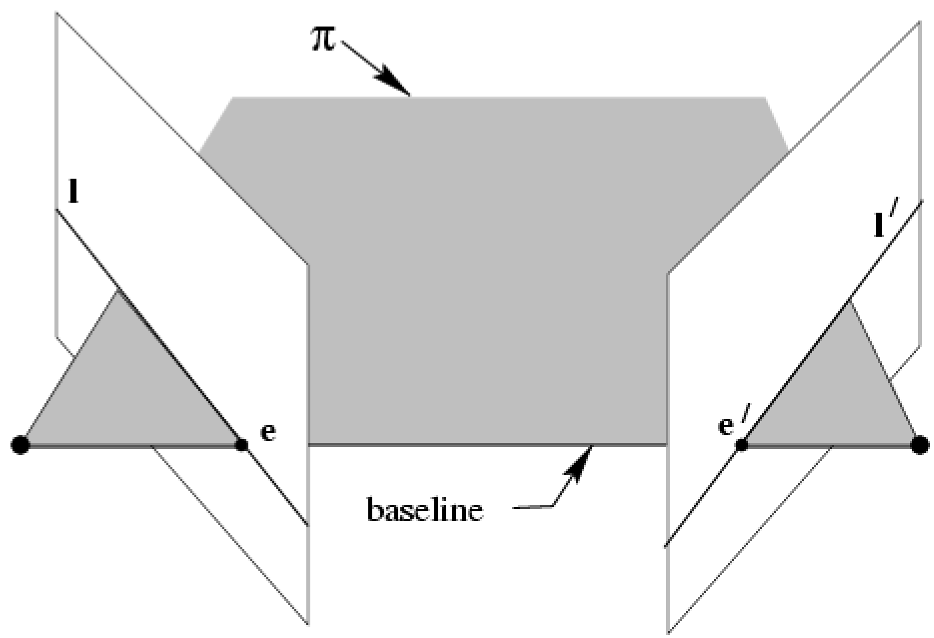


C,C',x,x' and X are coplanar

Epipolar Geometry

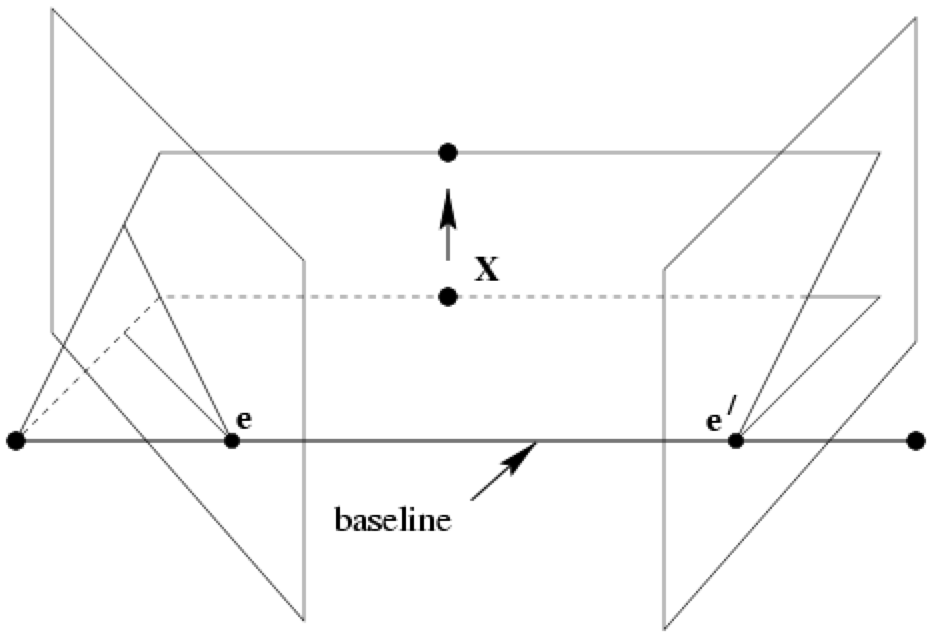


Epipolar Geometry



All points on p project on l and l’

Epipolar Geometry

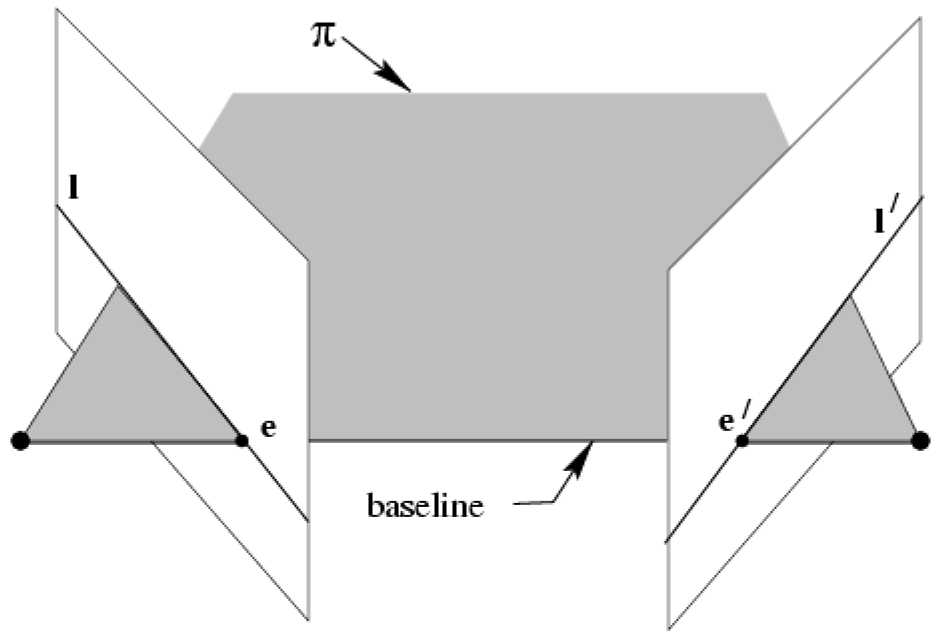


Family of planes p and lines l and l'
Intersection in e and e'

Epipolar Geometry

epipoles e,e'
= intersection of baseline with image plane
= projection of projection center in other image
= vanishing point of camera motion direction

an epipolar plane = plane containing baseline (1-D family)

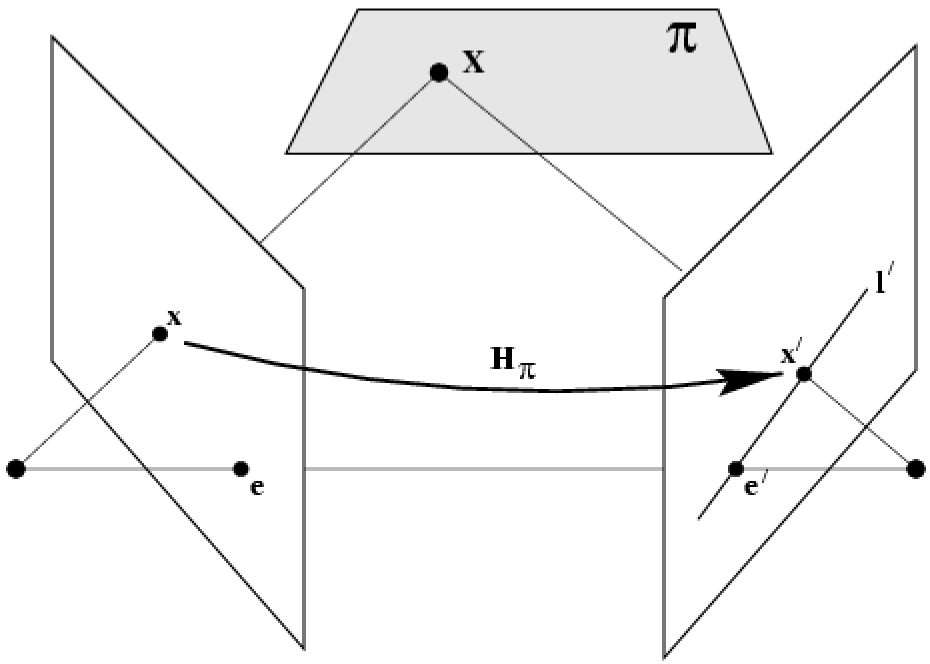


an epipolar line = intersection of epipolar plane with image
(always come in corresponding pairs)

The Fundamental Matrix F

$$x' = H_\pi x$$

$$l' = e' \times x' = [e']_\times H_\pi x = Fx$$



H : projectivity=collineation=projective transformation=homography

Cross Products

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

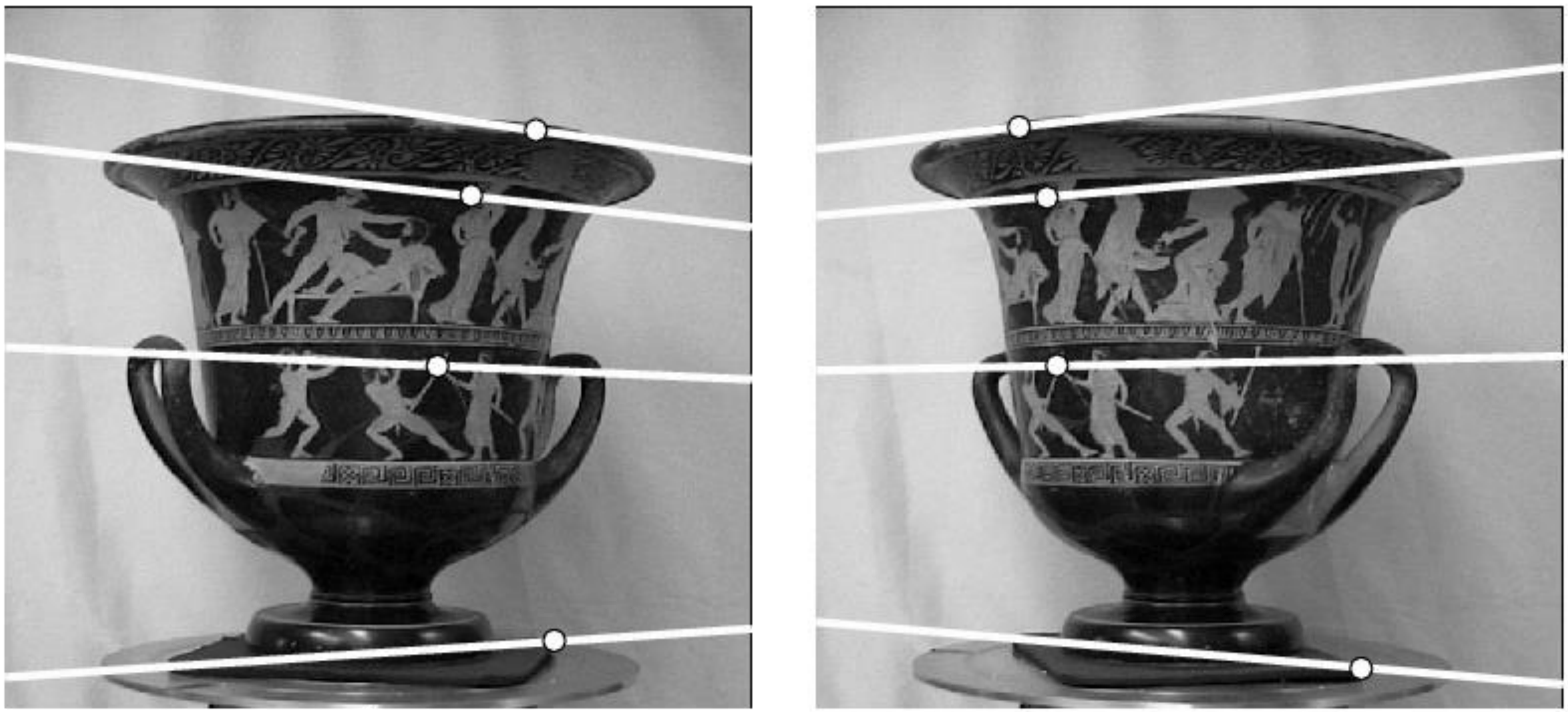
$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix} \mathbf{i} - \begin{vmatrix} a_1 & a_3 \\ b_1 & b_3 \end{vmatrix} \mathbf{j} + \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \mathbf{k}$$

The vector cross product also can be expressed as the product of a skew-symmetric matrix and a vector:

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_{\times} \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$[\mathbf{a}]_{\times} \stackrel{\text{def}}{=} \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

Epipolar Lines



Three Questions

(i) **Correspondence geometry:** Given an image point x in the first view, how does this constrain the position of the corresponding point x' in the second image?

(ii) **Camera geometry (motion):** Given a set of corresponding image points $\{x_i \leftrightarrow x'_i\}$, i=1,…,n, what are the cameras P and P' for the two views?

(iii) **Scene geometry (structure):** Given corresponding image points $x_i \leftrightarrow x'_i$ and cameras P, P', what is the position of (their pre-image) X in space?

Parameter estimation

- 2D homography

  Given a set of (x_i, x_i'), compute H $(x_i' = Hx_i)$

- 3D to 2D camera projection

  Given a set of (X_i, x_i), compute P $(x_i = PX_i)$

- Fundamental matrix

  Given a set of (x_i, x_i'), compute F $(x_i'^T F x_i = 0)$

The fundamental matrix F

Algebraic Derivation

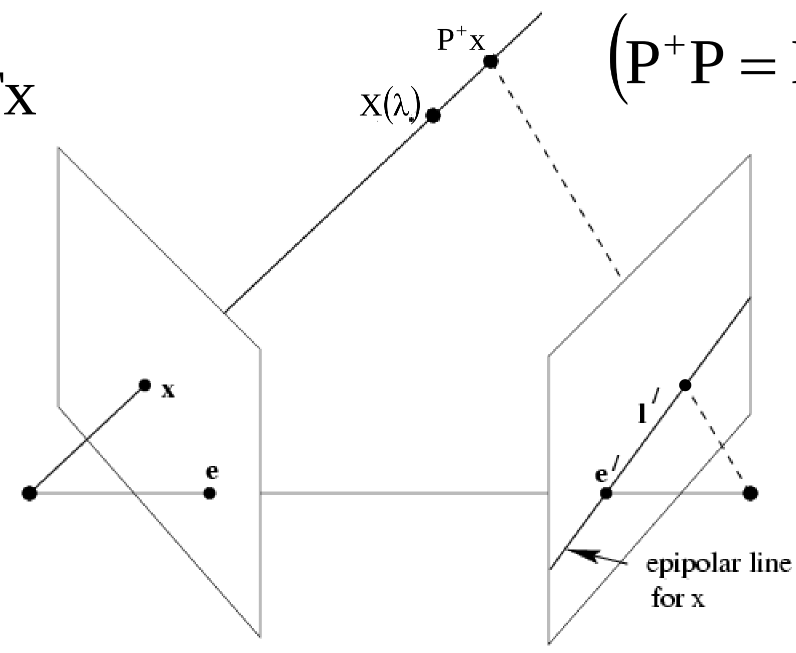
$$X(\lambda) = P^{+}x + \lambda C$$

$$l' = e' \times x' = [e']_{\times} H_{\pi} x = Fx$$

$$l' = P'C \times P'P^{+}x$$

$$F = [e']_{\times} P'P^{+}$$

$$(P^{+}P = I)$$



(note: doesn't work for C=C' $\Rightarrow$ F=0)

The fundamental matrix F

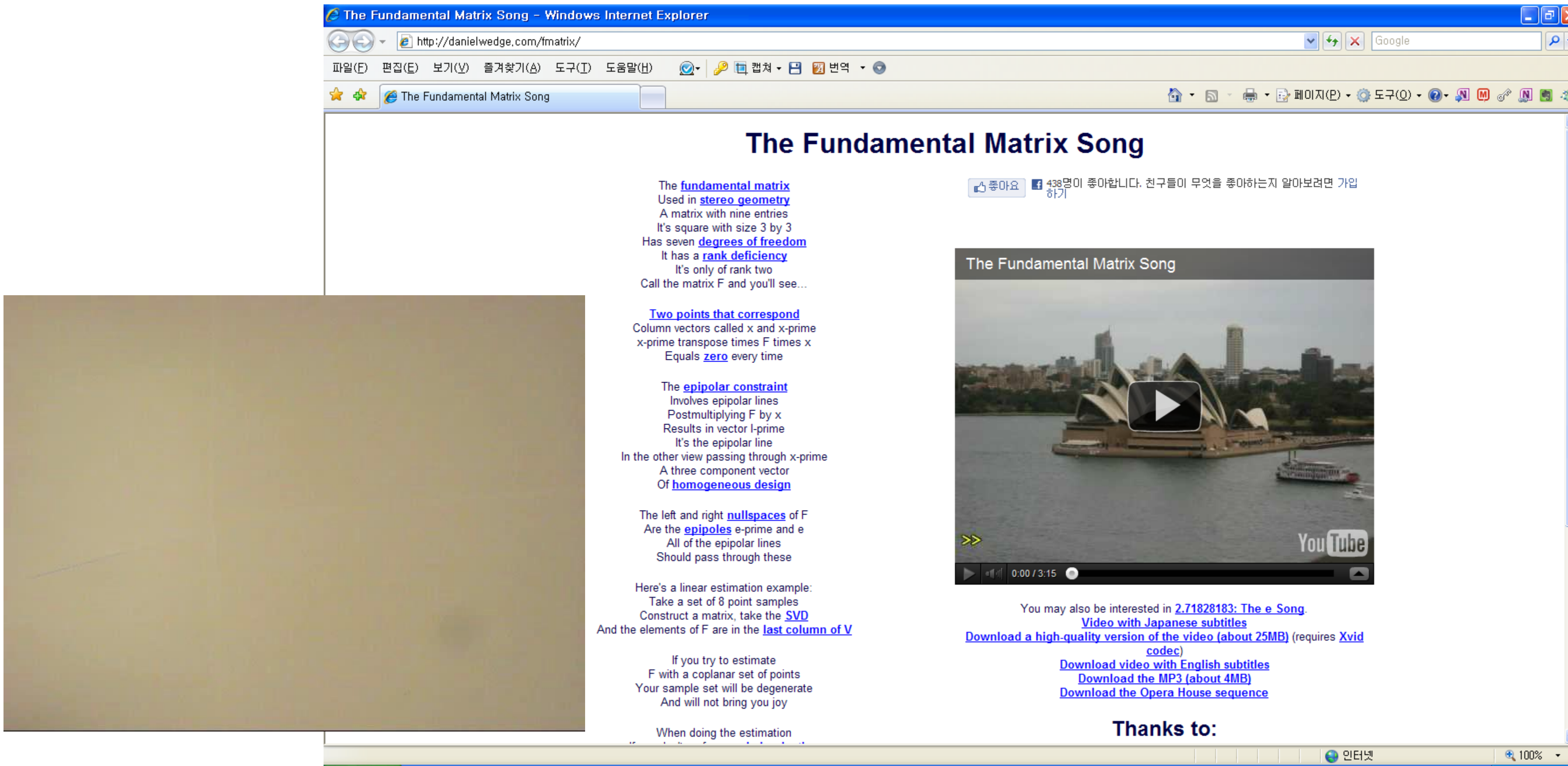
Correspondence Condition

The fundamental matrix satisfies the condition that for any pair of corresponding points $x \leftrightarrow x'$ in the two images

$$x'^{T} F x = 0$$

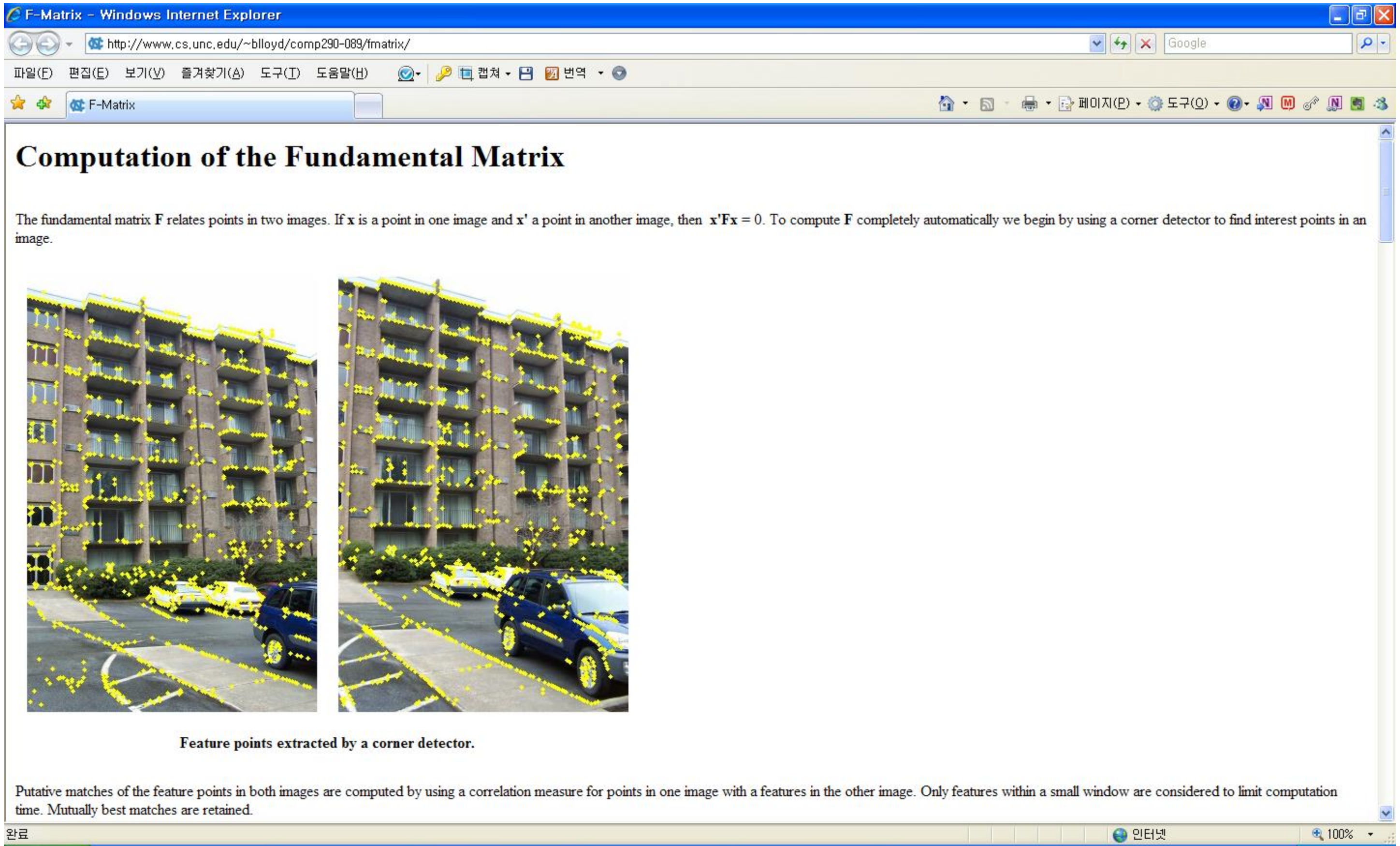
The Fundamental Matrix Song

- http://danielwedge.com/fmatrix/



The Fundamental Matrix

- http://www.cs.unc.edu/~blloyd/comp290-089/fmatrix/



Epipolar geometry: basic equation

$$\mathrm{x'^T F x} = 0$$

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

separate known from unknown

$$[x'x, x'y, x', y'x, y'y, y', x, y, 1][f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}]^T = 0$$

(data) (unknowns)

$$\begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{bmatrix} \mathrm{f} = 0$$

$$\mathrm{Af} = 0$$

the NOT normalized 8-point algorithm

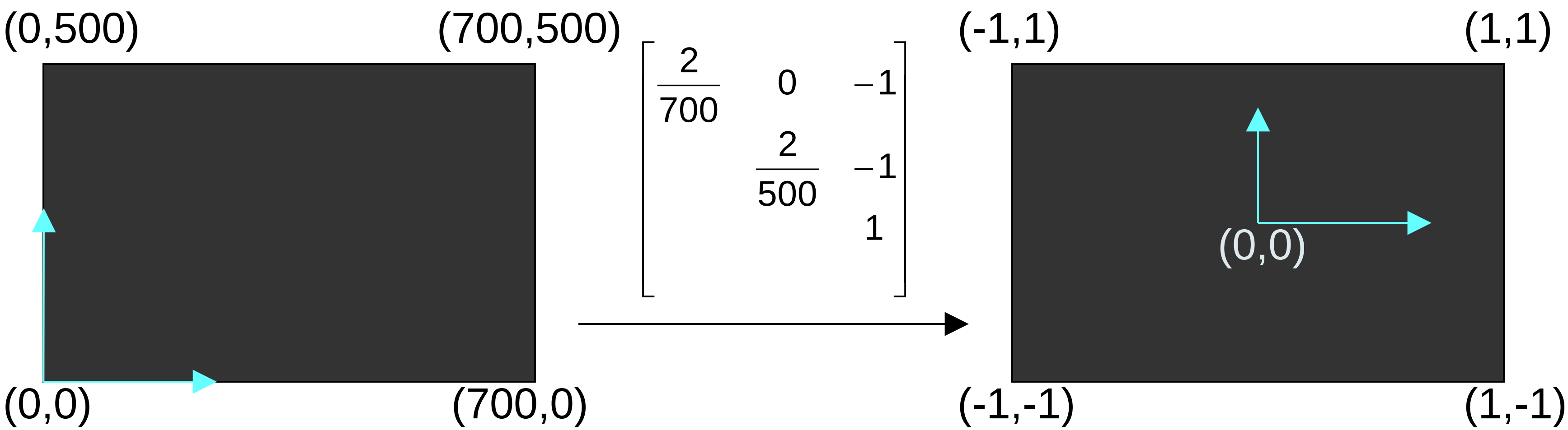
$$
\begin{bmatrix}
x_1x_1' & y_1x_1' & x_1' & x_1y_1' & y_1y_1' & y_1' & x_1 & y_1 & 1 \\
x_2x_2' & y_2x_2' & x_2' & x_2y_2' & y_2y_2' & y_2' & x_2 & y_2 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_nx_n' & y_nx_n' & x_n' & x_ny_n' & y_ny_n' & y_n' & x_n & y_n & 1
\end{bmatrix}
\begin{bmatrix}
f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33}
\end{bmatrix} = 0
$$

~10000 ~10000 ~100 ~10000 ~10000 ~100 ~100 ~100 1

Orders of magnitude difference
Between column of data matrix
$\rightarrow$ least-squares yields poor results

the normalized 8-point algorithm

Transform image to ~[-1,1]x[-1,1]



$$\begin{bmatrix} \frac{2}{700} & 0 & -1 \\ & \frac{2}{500} & -1 \\ & & 1 \end{bmatrix}$$

Least squares yields good results (Hartley, PAMI´97)

lbg@dongseo.ac.kr

The fundamental matrix F

F is the unique 3x3 rank 2 matrix that satisfies $x'^T F x = 0$ for all $x \leftrightarrow x'$

(i) **Transpose:** if F is fundamental matrix for (P,P'), then F^T is fundamental matrix for (P',P)

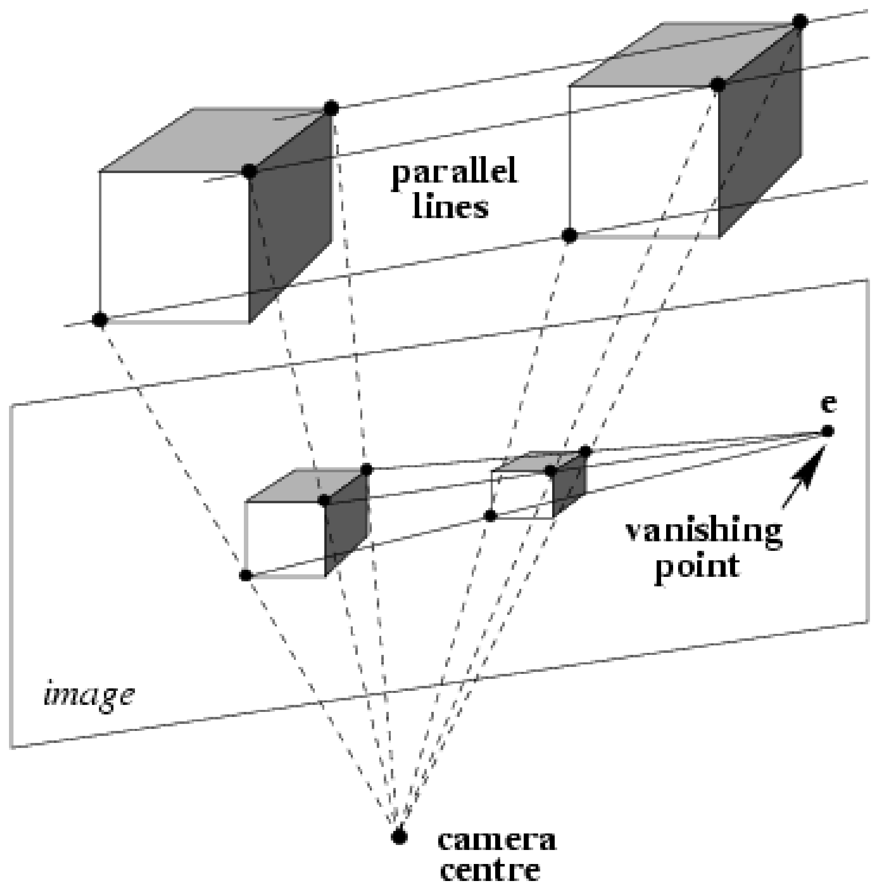
(ii) **Epipolar lines:** $l' = Fx$ & $l = F^T x'$

(iii) **Epipoles:** on all epipolar lines, thus $e'^T F x = 0, \forall x \Rightarrow e'^T F = 0$, similarly $Fe = 0$

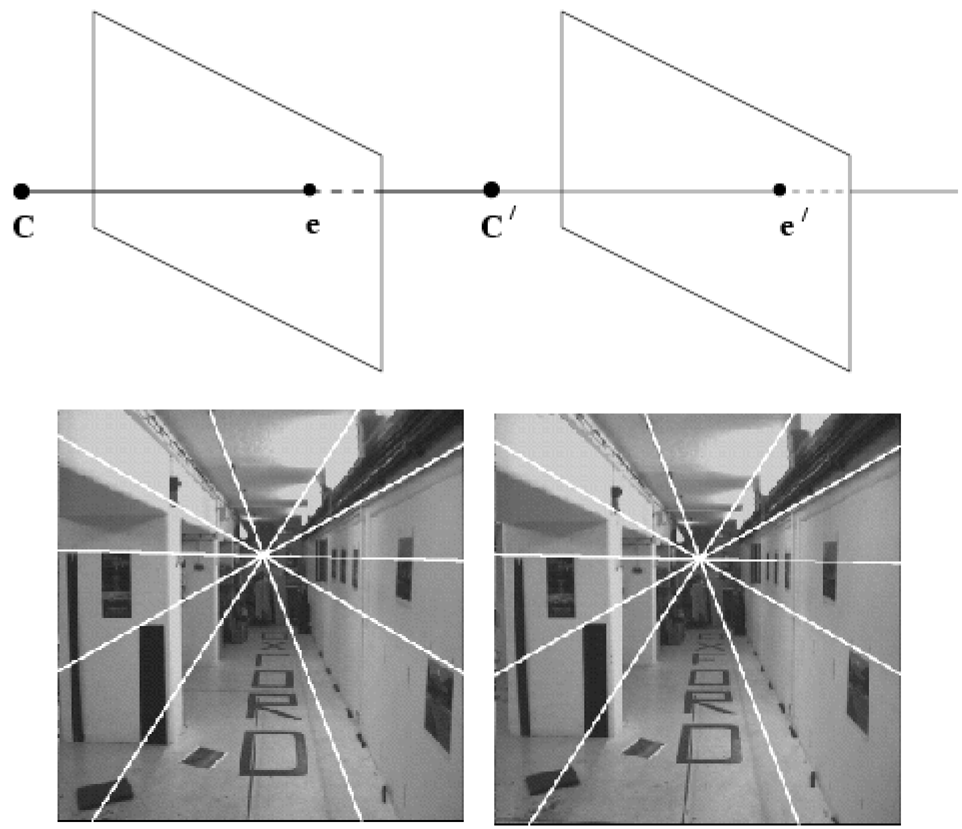
(iv) **F** has 7 d.o.f. , i.e. 3x3-1(homogeneous)-1(rank2)

(v) **F** is a correlation, projective mapping from a point x to a line $l' = Fx$ (not a proper correlation, i.e. not invertible)

Fundamental matrix for pure translation



Fundamental matrix for pure translation



Fundamental matrix for pure translation

$$F = [e']_\times H_\infty = [e']_\times \quad (H_\infty = K^{-1}RK)$$

Example:

$$e' = (1,0,0)^T \quad F = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}_\times$$

$$x'^T F x = 0 \Leftrightarrow y = y'$$

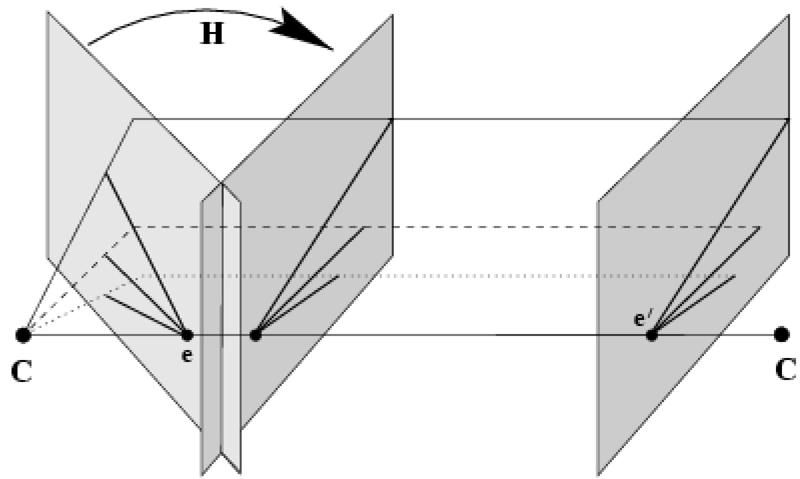
$$x = PX = K[I \mid 0]X \qquad (X,Y,Z)^T = K^{-1}x/Z$$

$$x' = P'X = K[I \mid t]\begin{bmatrix} K^{-1}x \\ Z \end{bmatrix} \qquad x' = x + Kt/Z$$

motion starts at x and moves towards e, faster depending on *Z*

pure translation: F only 2 d.o.f., $x^T[e]_x x=0 \Rightarrow$ auto-epipolar

General Motion



$$x'^{T}[e']_{\times}Hx = 0$$

$$x'^{T}[e']_{\times}\hat{x} = 0$$

$$x' = K'RK^{-1}x + K't/Z$$

Projective transformation and invariance

Derivation based purely on projective concepts

$$\hat{x} = Hx,\ \hat{x}' = H'x' \Rightarrow \hat{F} = H'^{-T} F H^{-1}$$

F invariant to transformations of projective 3-space

$$x = PX = (PH)(H^{-1}X) = \hat{P}\hat{X}$$

$$x' = P'X = (P'H)(H^{-1}X) = \hat{P}'\hat{X}$$

$(P, P') \mapsto F$ unique

$F \mapsto (P, P')$ not unique

Canonical form

$$P = [I \mid 0] \qquad F = [m]_\times M \qquad F = [e']_\times P'P^+$$

$$P' = [M \mid m]$$

Projective ambiguity of cameras given F

previous slide: at least projective ambiguity this slide: not more!

Show that if F is same for (P,P') and ($\tilde{P}$,$\tilde{P}$'), there exists a projective transformation H so that $\tilde{P}$=HP and $\tilde{P}$'=HP'

$$\mathrm{P}=[\mathrm{I}\,|\,0]\ \mathrm{P'}=[\mathrm{A}\,|\,\mathrm{a}]\ \ \tilde{\mathrm{P}}=[\mathrm{I}\,|\,0]\ \tilde{\mathrm{P}}'=[\tilde{\mathrm{A}}\,|\,\tilde{\mathrm{a}}]$$

$$\mathrm{F}=[\mathrm{a}]_\times \mathrm{A}=[\tilde{\mathrm{a}}]_\times \tilde{\mathrm{A}}$$

<u>lemma:</u>

$$\tilde{\mathrm{a}}=k\mathrm{a}\ \ \tilde{\mathrm{A}}=k^{-1}\left(\mathrm{A}+\mathrm{av}^\mathrm{T}\right)\qquad \mathrm{aF}=\mathrm{a}[\mathrm{a}]_\times \mathrm{A}=0=\tilde{\mathrm{a}}\mathrm{F}\overset{\text{rank 2}}{\Longrightarrow}\tilde{\mathrm{a}}=k\mathrm{a}$$

$$[\mathrm{a}]_\times \mathrm{A}=[\tilde{\mathrm{a}}]_\times \tilde{\mathrm{A}}\Rightarrow[\mathrm{a}]_\times\left(k\tilde{\mathrm{A}}-\mathrm{A}\right)=0\Rightarrow\left(k\tilde{\mathrm{A}}-\mathrm{A}\right)=\mathrm{av}^\mathrm{T}$$

$$H=\begin{bmatrix} k^{-1}I & 0 \\ k^{-1}\mathrm{v}^\mathrm{T} & k \end{bmatrix}$$

$$\mathrm{P'H}=[\mathrm{A}\,|\,\mathrm{a}]\begin{bmatrix} k^{-1}I & 0 \\ k^{-1}\mathrm{v}^\mathrm{T} & k \end{bmatrix}=[k^{-1}\left(\mathrm{A}-\mathrm{av}^\mathrm{T}\right)|\,k\mathrm{a}]=\tilde{\mathrm{P}}'$$ (22-15=7, ok)

Canonical cameras given F

F matrix corresponds to P,P' iff $P'^T F P$ is skew-symmetric

$$\left(X^T P'^T F P X = 0, \forall X\right)$$

F matrix, S skew-symmetric matrix

$$P = [I \mid 0] \quad P' = [SF \mid e'] \quad \text{(fund.matrix=F)}$$

$$\left([SF \mid e']^T F [I \mid 0] = \begin{bmatrix} F^T S^T F & 0 \\ e'^T F & 0 \end{bmatrix} = \begin{bmatrix} F^T S^T F & 0 \\ 0 & 0 \end{bmatrix} \right)$$

Possible choice:

$$P = [I \mid 0] \quad P' = [[e']_\times F \mid e']$$

Canonical representation:

$$P = [I \mid 0] \quad P' = [[e']_\times F + e' v^T \mid \lambda e']$$

The Essential matrix

~ Fundamental matrix for calibrated cameras (remove K)

$$E = [t]_{\times} R = R[R^{T} t]_{\times}$$

$$\hat{x}'^{T} E \hat{x} = 0 \qquad \left(\hat{x} = K^{-1} x;\ \hat{x}' = K^{-1} x'\right)$$

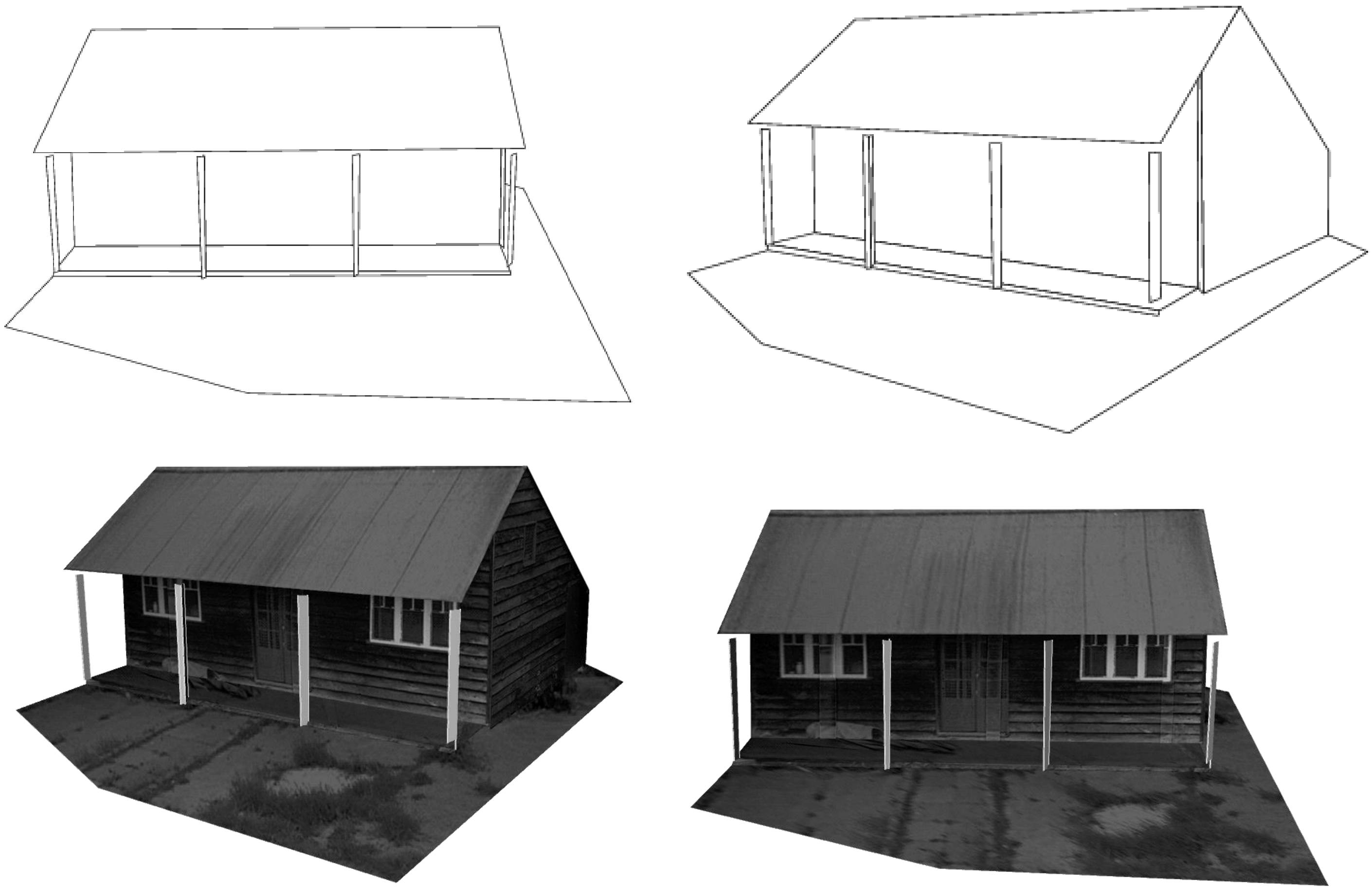
$$E = K'^{T} F K$$

5 d.o.f. (3 for R; 2 for t up to scale)

E is essential matrix if and only if two singularvalues are equal (and third=0)

$$E = U \mathrm{diag}(1,1,0) V^{T}$$

3D Reconstruction



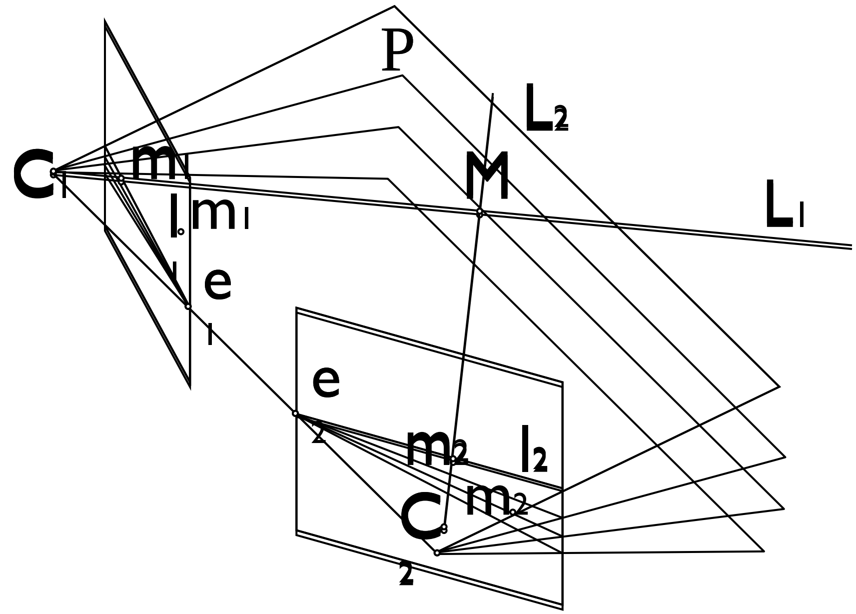
Epipolar Geometry

Underlying structure in set of matches for rigid scenes

$l^T_1 \quad l_2$

$$m_2^T \mathbf{F} \, m_1 = 0$$

Fundamental matrix
(3x3 rank 2 matrix)



1. Computable from corresponding points
2. Simplifies matching
3. Allows to detect wrong matches
4. Related to calibration

Canonical representation:

$$P = [I \mid 0] \quad P' = [[e']_\times F + e' v^T \mid \lambda e']$$

3D reconstruction of cameras and structure

Reconstruction Problem:

given $x_i \leftrightarrow x'_i$, compute P,P' and X_i

$$\mathrm{x}_i = \mathrm{P}\mathrm{X}_i \quad \mathrm{x}'_i = \mathrm{P}\mathrm{X}'_i \quad \text{for all } i$$

without additional informastion possible up to projective ambiguity

Outline of 3D Reconstruction

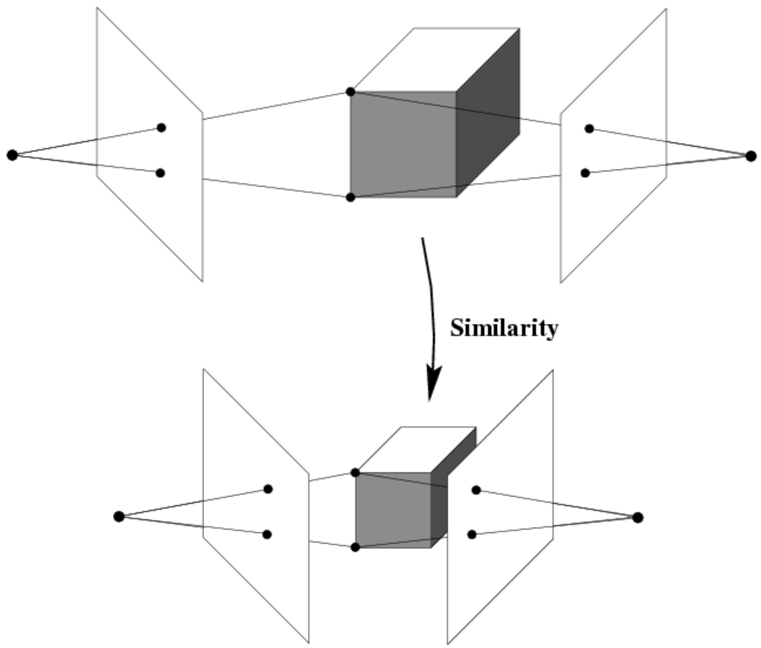
(i) Compute F from correspondences
(ii) Compute camera matrices P,P‘ from F
(iii) Compute 3D point for each pair of corresponding points

computation of F
use $x'_i F x_i = 0$ equations, linear in coeff. F
8 points (linear), 7 points (non-linear), 8+ (least-squares)

computation of camera matrices
use $P = [I \mid 0] \quad P' = [[e']_\times F + e' v^T \mid \lambda e']$

triangulation
compute intersection of two backprojected rays

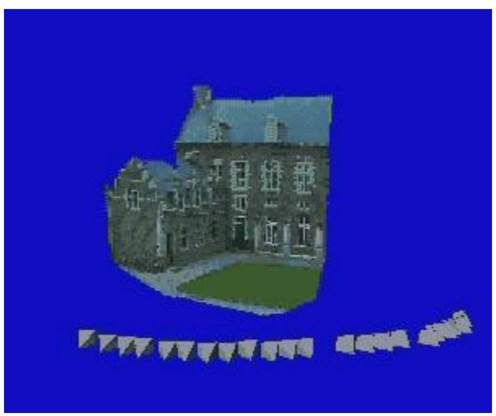
Reconstruction ambiguity: similarity

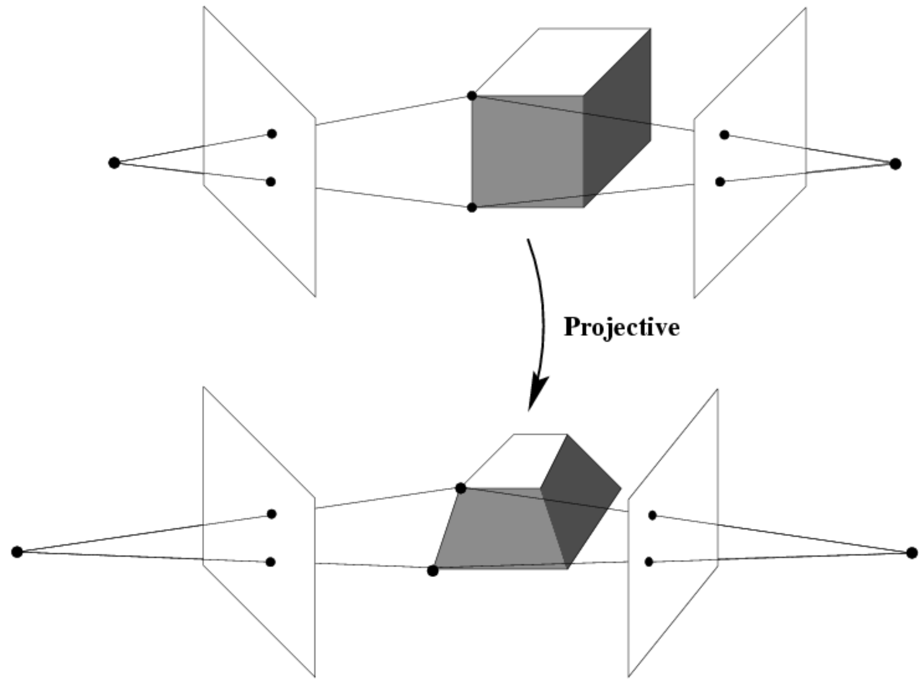


$$x_i = PX_i = \left(PH_S^{-1}\right)\left(H_S X_i\right)$$

$$PH_S^{-1} = K[R\,|\,t]\begin{bmatrix} R'^T & -R'^T t' \\ 0 & \lambda \end{bmatrix} = K[RR'^T\,|\,-RR'^T t' + \lambda t]$$

Reconstruction ambiguity: projective





$$x_i = PX_i = \left(PH_P^{-1}\right)\left(H_P X_i\right)$$

Terminology

$x_i \leftrightarrow x'_i$

Original scene X_i

Projective, affine, similarity reconstruction
= reconstruction that is identical to original up to projective, affine, similarity transformation

Literature: Metric and Euclidean reconstruction
= similarity reconstruction

The projective reconstruction theorem

If a set of point correspondences in two views determine the fundamental matrix uniquely, then the scene and cameras may be reconstructed from these correspondences alone, and any two such reconstructions from these correspondences are projectively equivalent

$$x_i \leftrightarrow x'_i \quad (P_1, P_1', \{X_{1i}\}) \quad (P_2, P_2', \{X_{2i}\})$$

$$P_2 = P_1 H^{-1} \quad P'_2 = P'_1 H^{-1} \quad X_2 = HX_1 \quad (\text{except}: Fx_i = x'_i F = 0)$$

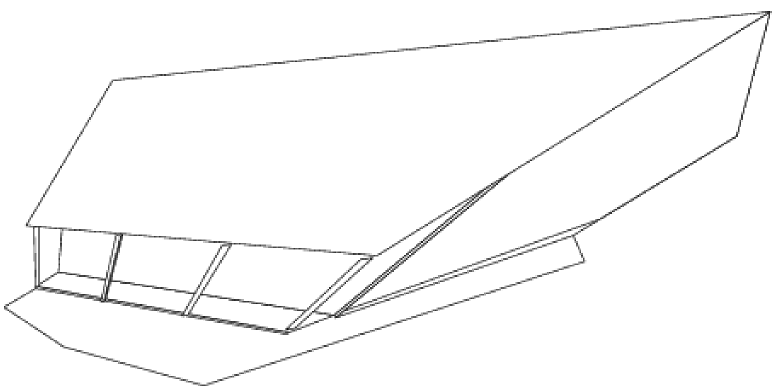
theorem from last class

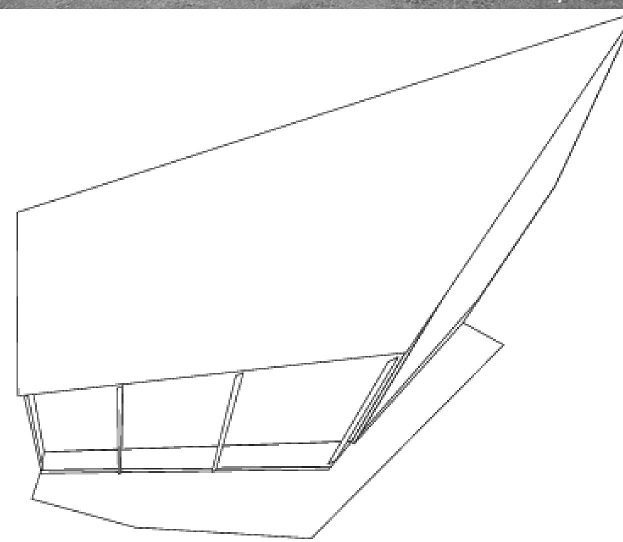
$$P_2(HX_{1i}) = P_1 H^{-1} HX_{1i} = P_1 X_{1i} = x_i = P_2 X_{2i}$$

$\Rightarrow$ along same ray of P_2, idem for P'_2

two possibilities: $X_{2i}=HX_{1i}$, or points along baseline

key result : allows reconstruction from pair of uncalibrated images



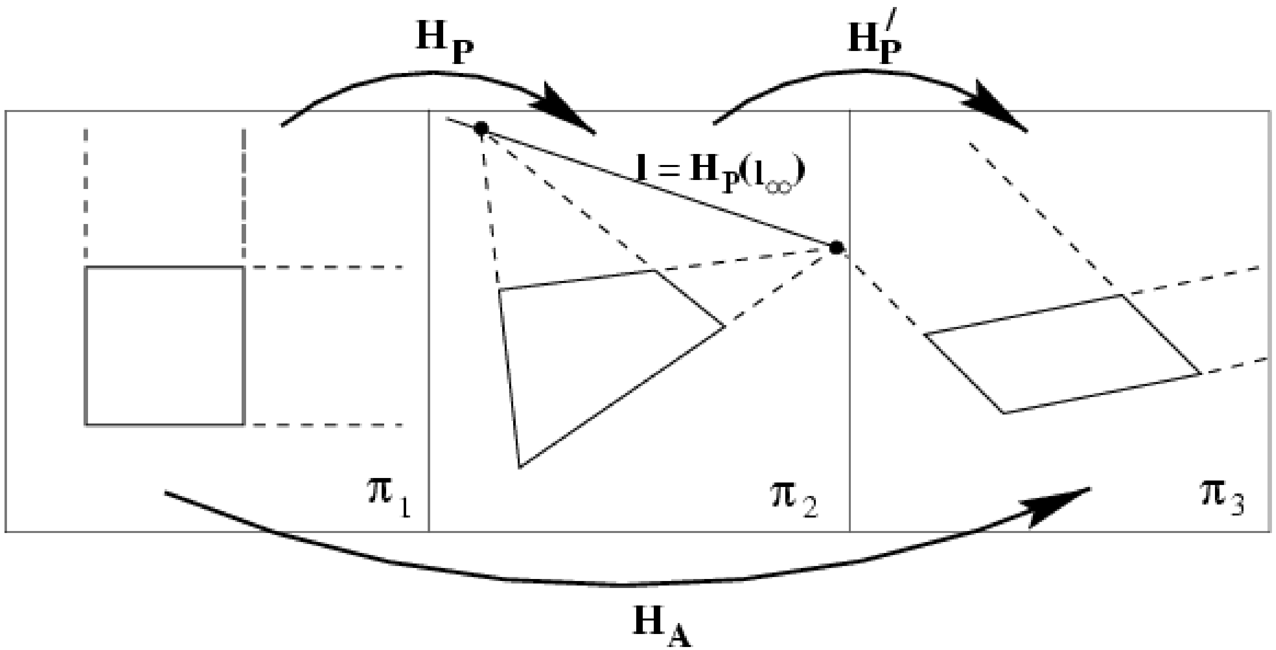


Stratified reconstruction

(i) Projective reconstruction
(ii) Affine reconstruction
(iii) Metric reconstruction

Projective to affine

Remember 2-D case



Projective to affine

$$\left(\mathrm{P}, \mathrm{P}', \{\mathrm{X}_i\}\right)$$

$$\pi_\infty = (A, B, C, D)^\mathsf{T} \mapsto (0,0,0,1)^\mathsf{T}$$

$$\mathrm{H}^{-\mathsf{T}} \pi_\infty = (0,0,0,1)^\mathsf{T}$$

$$\mathrm{H} = \begin{bmatrix} \mathrm{I} \mid 0 \\ \pi_\infty \end{bmatrix}$$ (if $D \neq 0$)

theorem says up to projective transformation,
but projective with fixed p_∞ is affine transformation

can be sufficient depending on application,
e.g. mid-point, centroid, parallellism

Translational motion

points at infinity are fixed for a pure translation

$\Rightarrow$ reconstruction of $x_i \leftrightarrow x_i$ is on p_∞

$$F = [e]_\times = [e']_\times \qquad P = [I \mid 0]$$

$$P = [I \mid e']$$

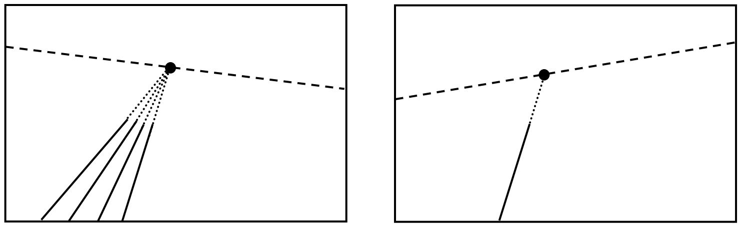
Scene constraints

Parallel lines
parallel lines intersect at infinity
reconstruction of corresponding vanishing point yields
point on plane at infinity

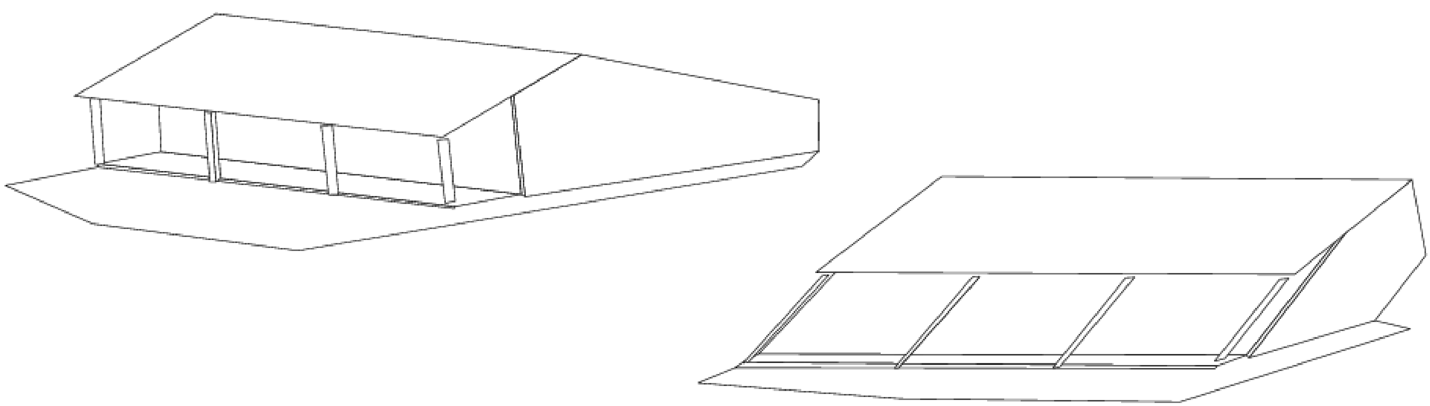
3 sets of parallel lines allow to uniquely determine p_∞

remark: in presence of noise determining the intersection of parallel lines is a delicate problem

remark: obtaining vanishing point in one image can be sufficient



Scene constraints



Scene constraints

Distance ratios on a line

known distance ratio along a line allow to determine point at infinity (same as 2D case)

The infinity homography

$$P = [M \mid m] \quad P' = [M' \mid m']$$

$$H_\infty = M' M^{-1}$$

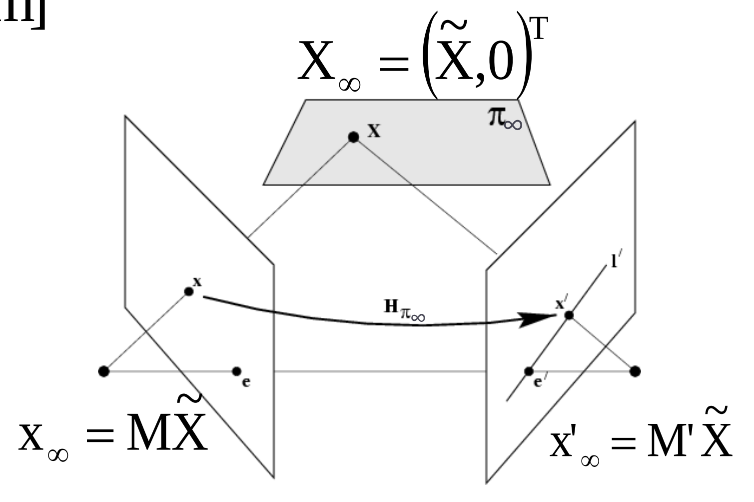
unchanged under affine transformations

$$P = [M \mid m]\begin{bmatrix} A & a \\ 0 & 1 \end{bmatrix} = [MA \mid Ma + m]$$

$$H_\infty = M' A A^{-1} M^{-1}$$

affine reconstruction

$$P = [I \mid 0] \quad P = [H_\infty \mid e]$$



One of the cameras is affine

According to the definition,
the principal plane of an affine camera is at infinity

to obtain affine recontruction,
compute H that maps third row of P to $(0,0,0,1)^T$
and apply to cameras and reconstruction

e.g. if P=[I|0], swap 3rd and 4th row, i.e.

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Affine to metric

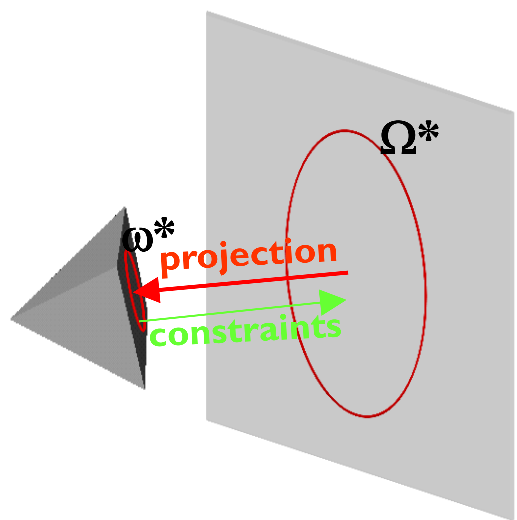
identify absolute conic

transform so that $\Omega_\infty : X^2 + Y^2 + Z^2 = 0, \text{on } \pi_\infty$

then projective transformation relating original and reconstruction is a <u>similarity transformation</u>

in practice, find image of W_∞
image w_∞ back-projects to cone that intersects p_∞ in W_∞

note that image is independent of particular reconstruction



Affine to metric

given $P = [M \mid m] \quad \omega$

possible transformation from affine to metric is

$$H = \begin{bmatrix} A^{-1} & 0 \\ 0 & 1 \end{bmatrix} \qquad AA^T = \left(M^T \omega M\right)^{-1}$$

(cholesky factorisation)

<u>proof:</u>

$$P_M = PH^{-1} = [MA \mid m]$$

$$\omega^* = M_M M_M{}^T = MAA^T M^T \quad \left(\Omega^* = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}\right)$$

$$M^{-1} \omega^{-1} M^{-T} = AA^T$$

Orthogonality

vanishing points corresponding to orthogonal directions

$$v_1^T \omega v_2 = 0$$

vanishing line and vanishing point corresponding to plane and normal direction

$$l = \omega v$$

lbg@dongseo.ac.kr

Correspondence and RANSAC Algorithm.

lbg@dongseo.ac.kr

Correspondence Search

Feature Matching

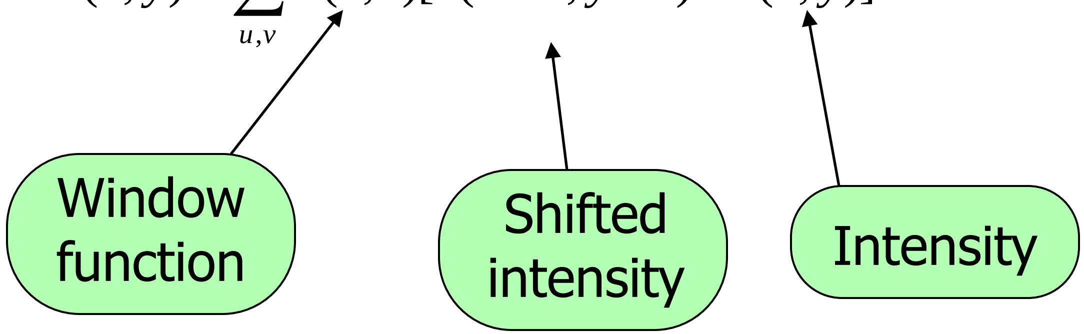
To match the points in one image to the points in the other image by exhaustive search(to match one point in one image to all the points in the other image) is a difficult and long process so some constraints are applied. As geometric constraint to minimize the search area for correspondence.

The geometric constraints is provided by the epipolar geometry

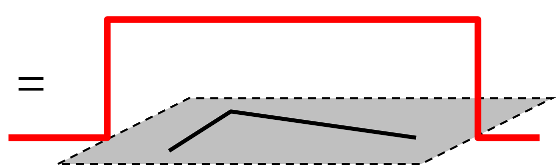
Harris Detector: Mathematics

Change of intensity for the shift [*x*,*y*]:

$$E(x,y) = \sum_{u,v} w(u,v)[I(x+u, y+v) - I(x,y)]^2$$

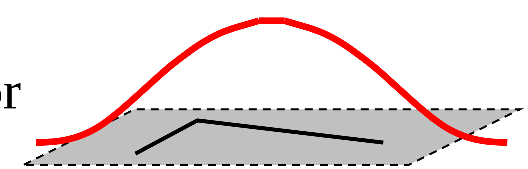


Window function $w(u,v)$ =



1 in window, 0 outside

or



Gaussian

$$E(x,y)=\sum_{w}[I(x,y)-I(x+\Delta x,y+\Delta y)]^2 \qquad \mathrm{u}=\Delta \mathrm{x}, \mathrm{v}=\Delta \mathrm{y}$$

$$=\sum_{w}\left(I(x,y)-I(x,y)-[I_x(x,y)I_y(x,y)]\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}\right)^2$$

$$=\sum_{w}\left([I_x(x,y)I_y(x,y)]\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}\right)^2$$

$$=\begin{bmatrix}\Delta x & \Delta y\end{bmatrix}\begin{bmatrix}\sum_{w}(I_x(x,y))^2 & \sum_{w}I_x(x,y)I_y(x,y)\\ \sum_{w}I_x(x,y)I_y(x,y) & \sum_{w}(I_y(x,y))^2\end{bmatrix}\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}$$

$$=\begin{bmatrix}\Delta x & \Delta y\end{bmatrix}C(x,y)\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix}$$

Harris Detector: Mathematics

For small shifts [u,v] we have a *bilinear* approximation:

$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a 2×2 matrix computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Harris Detector: Mathematics

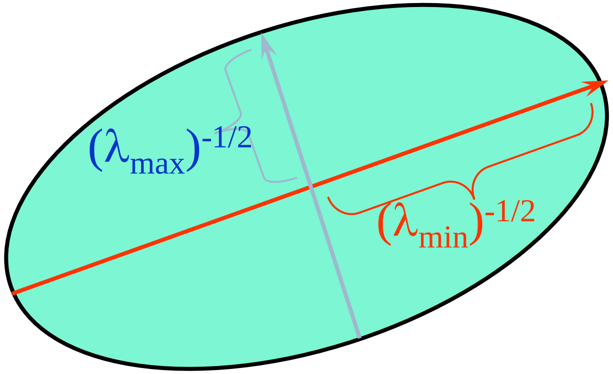
Intensity change in shifting window: eigenvalue analysis

$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

λ_1, λ_2 – eigenvalues of M

If we try every possible orientation **n**, the max. change in intensity is λ_2

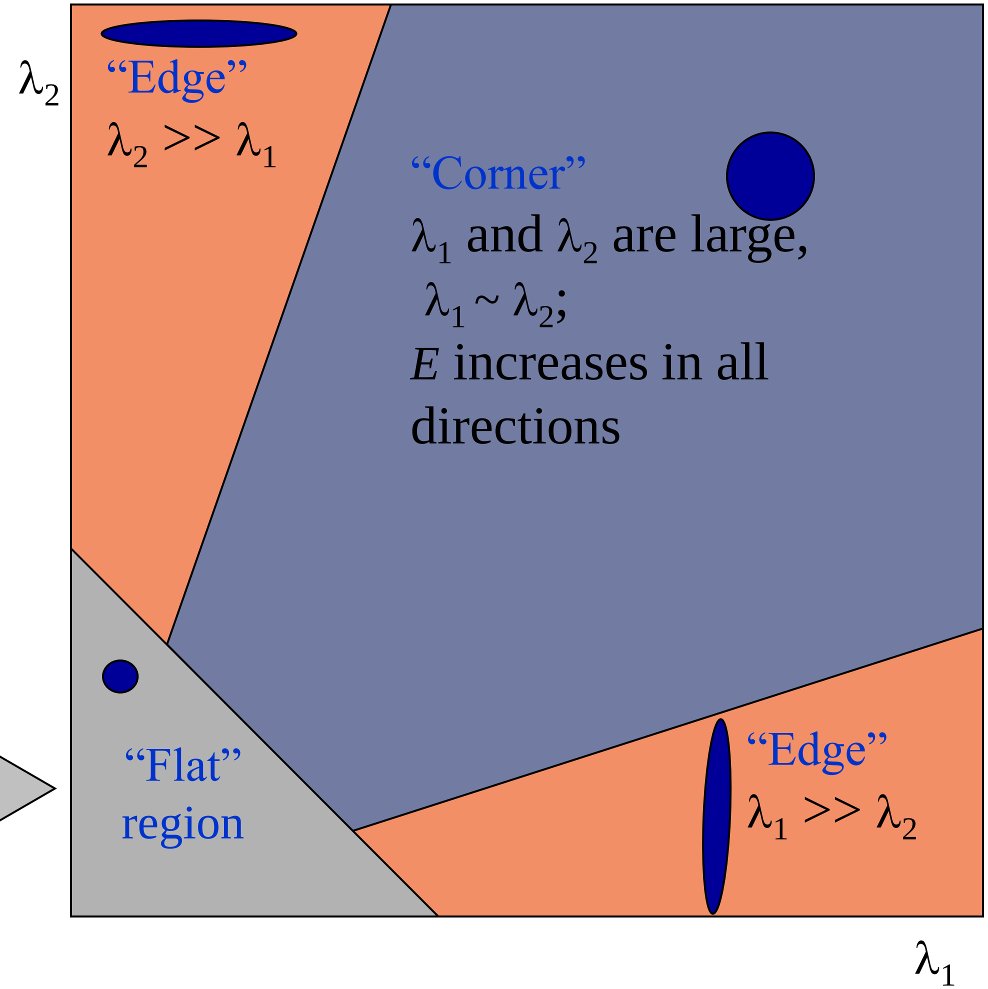
Ellipse $E(u,v)$ = const



Harris Detector: Mathematics

Classification of image points using eigenvalues of M:

λ_1 and λ_2 are small;
E is almost constant in all directions



Harris Detector: Mathematics

Measure of corner response:

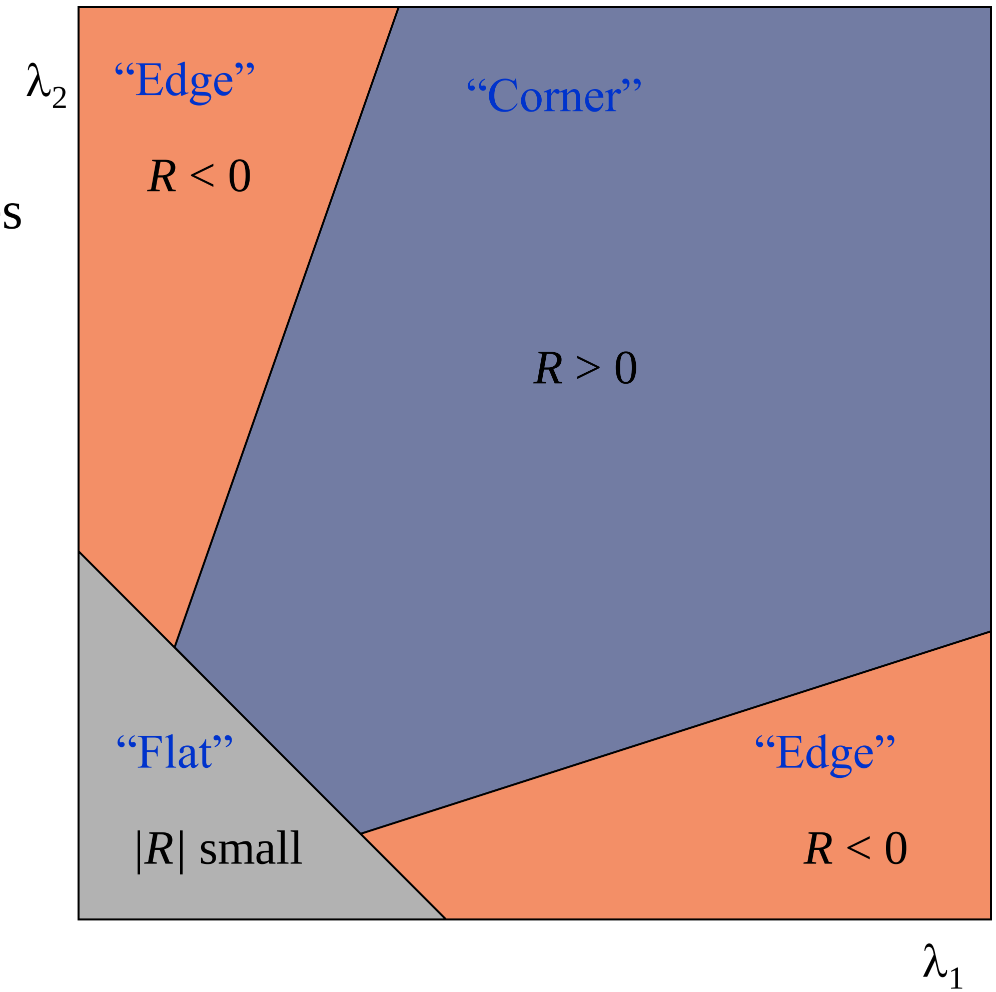
$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

(k – empirical constant, k = 0.04-0.06)

Harris Detector: Mathematics

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



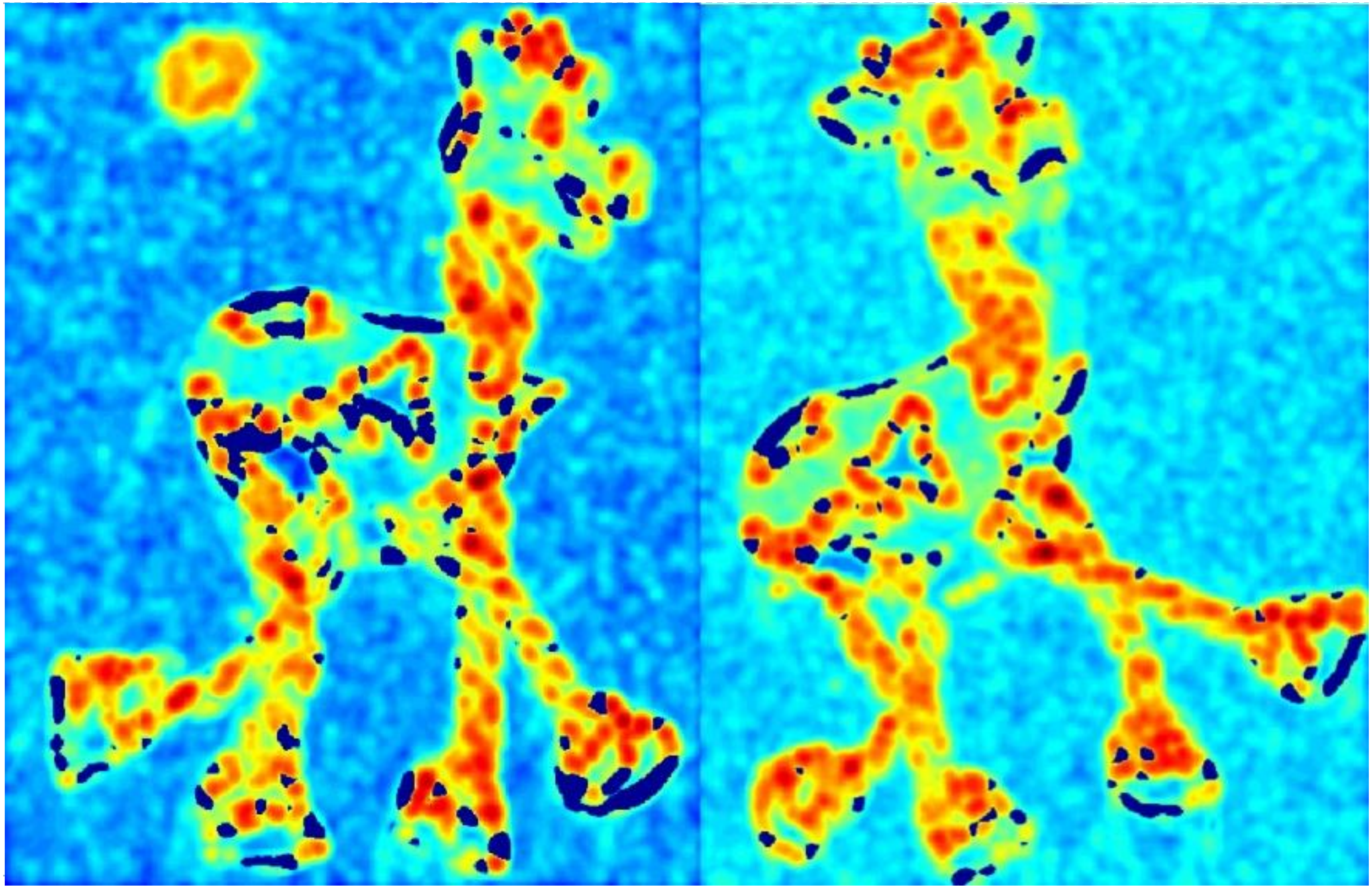
Harris Detector

- The Algorithm:
  - Find points with large corner response function R
  - Take the points of local maxima of R

Harris Detector : Workflow

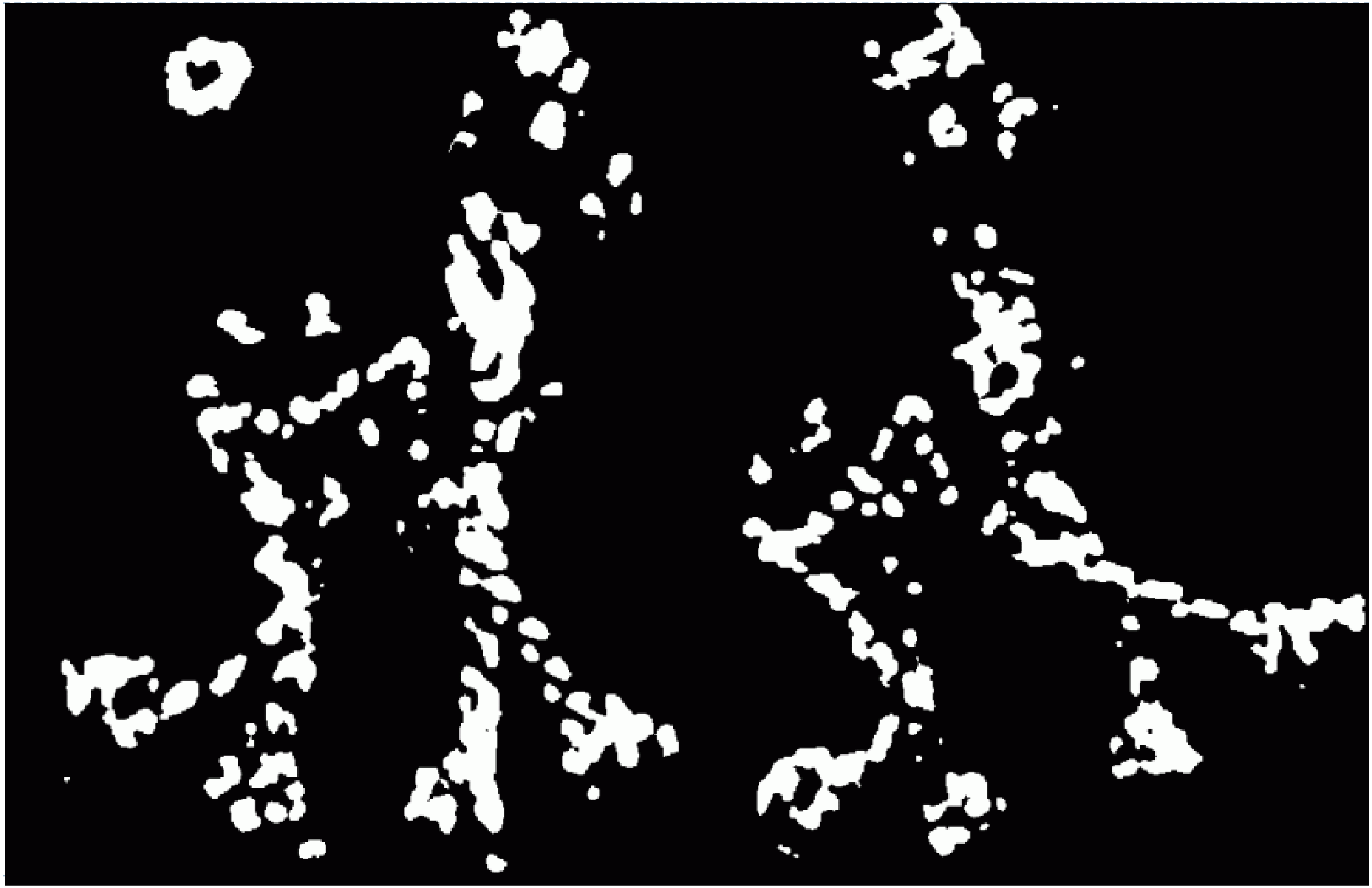
Compute corner response R

Harris Detector : Workflow

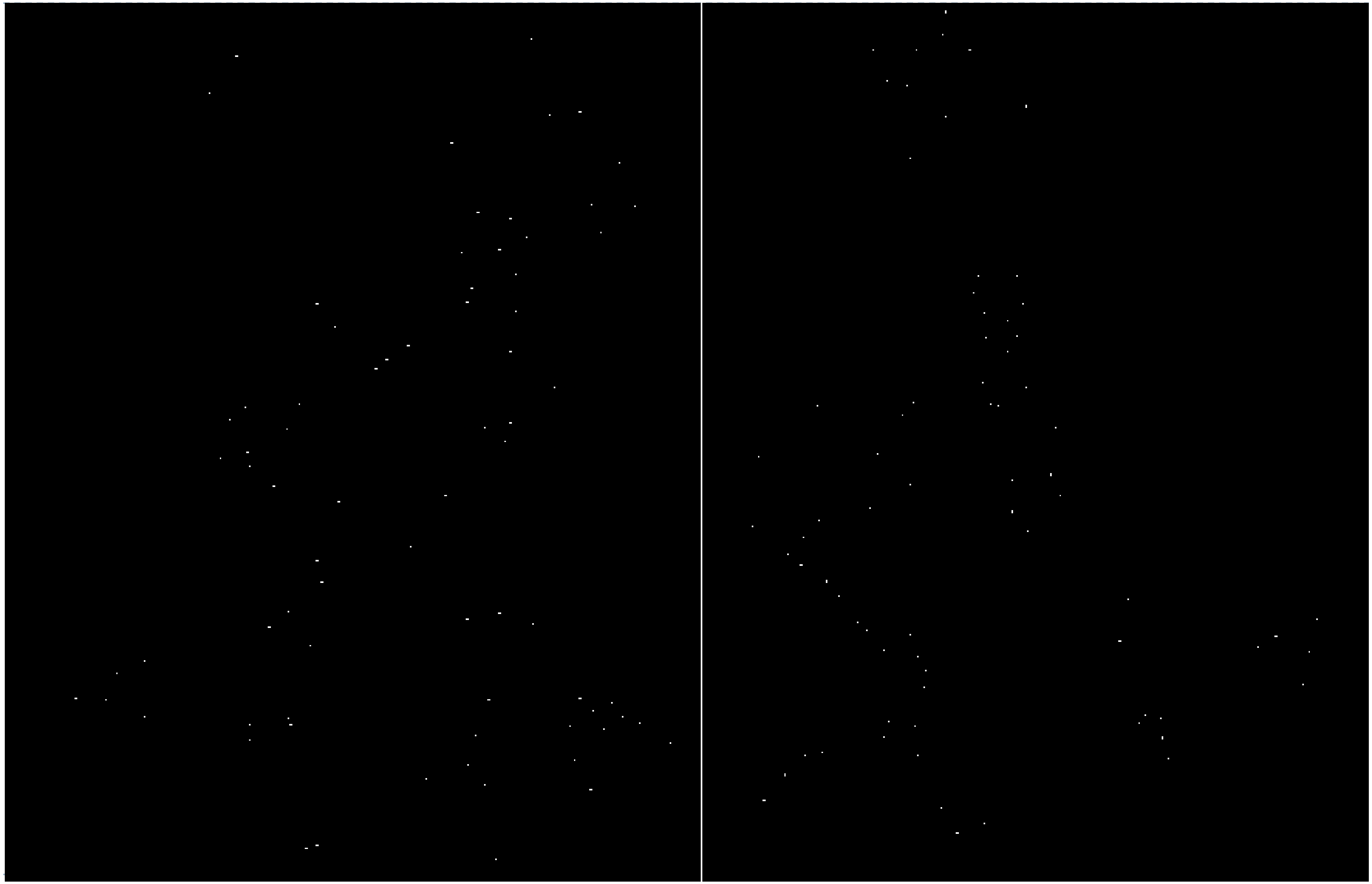


Find points with large corner response: R>threshold

Harris Detector : Workflow

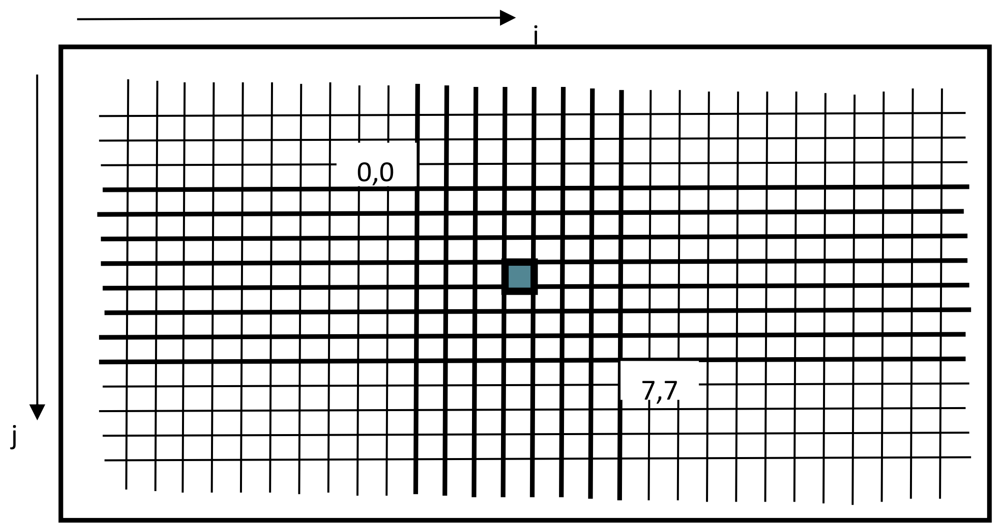


Take only the points of local maxima of R
Harris Detector : Workflow



Harris Detector : Workflow

Correlation for Correspondence Search



Left image: 1. From the left feature point image we select one feature point.

2. Draw the window(N*N) around, with feature point in the center.

3. Calculate the normalized window using the formula

$$s_1 = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} w_1(i,j) * w_1(i,j)} \qquad \text{N } \textit{is the size of window}$$

$$w_{1(nor)}(i,j) = \frac{w_1(i,j)}{s_1}$$

Correlation Algorithm

- Select one feature point from the first image.
- Draw the window across it(7*7).
- Normalize the window using the given formula.

$$s_1 = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} w_1(i,j) * w_1(i,j)} \quad \text{N } \textit{is the size of window}$$

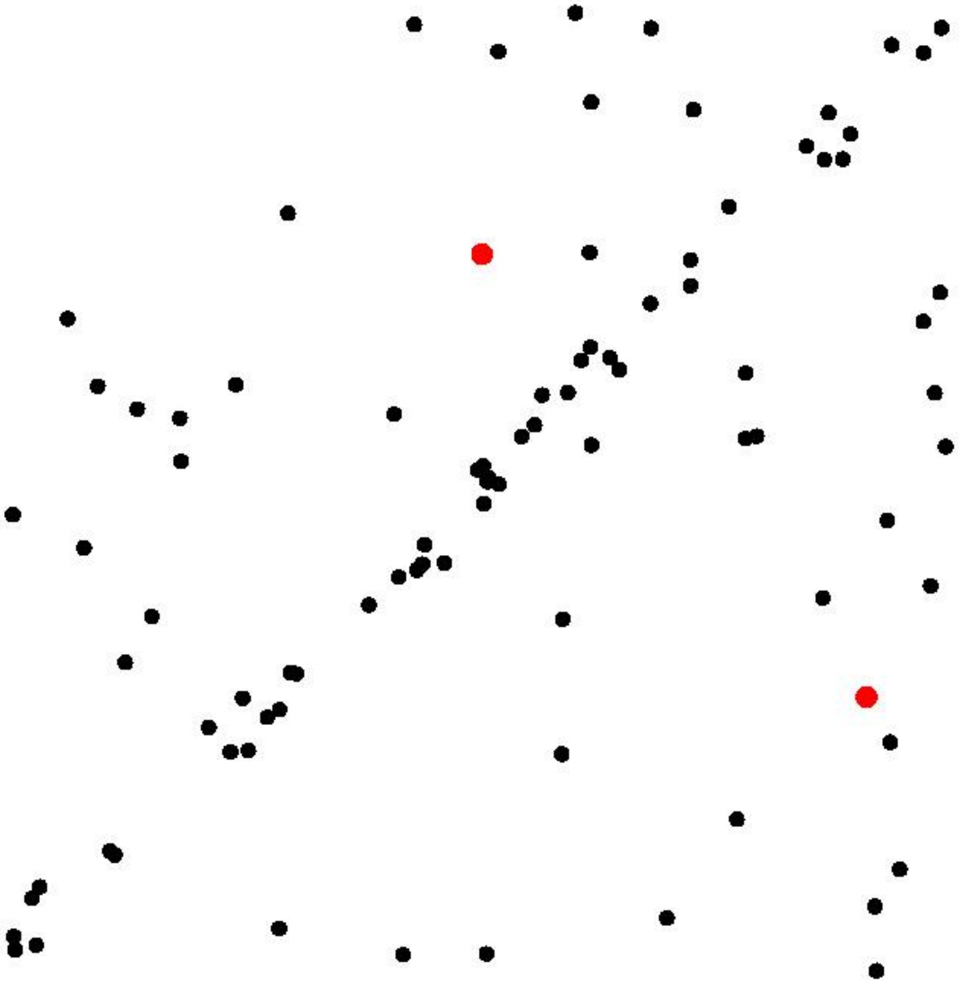
$$w_{1(nor)}(i,j) = \frac{w_1(i,j)}{s_1}$$

- Find the feature in the right image, that are to be considered in the first image,(this should be done by some distance thresholding)
- After finding the feature point the window of same dimension should be selected in the second image.
- The normalized correlstion measure should be calculated using the formula

$$s_2 = \sum_{i=1}^{N}\sum_{j=1}^{N} w_1(i,j) * w_2(i,j) \qquad cormat = \frac{s_2}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} w_2(i,j) * w_2(i,j)}}$$

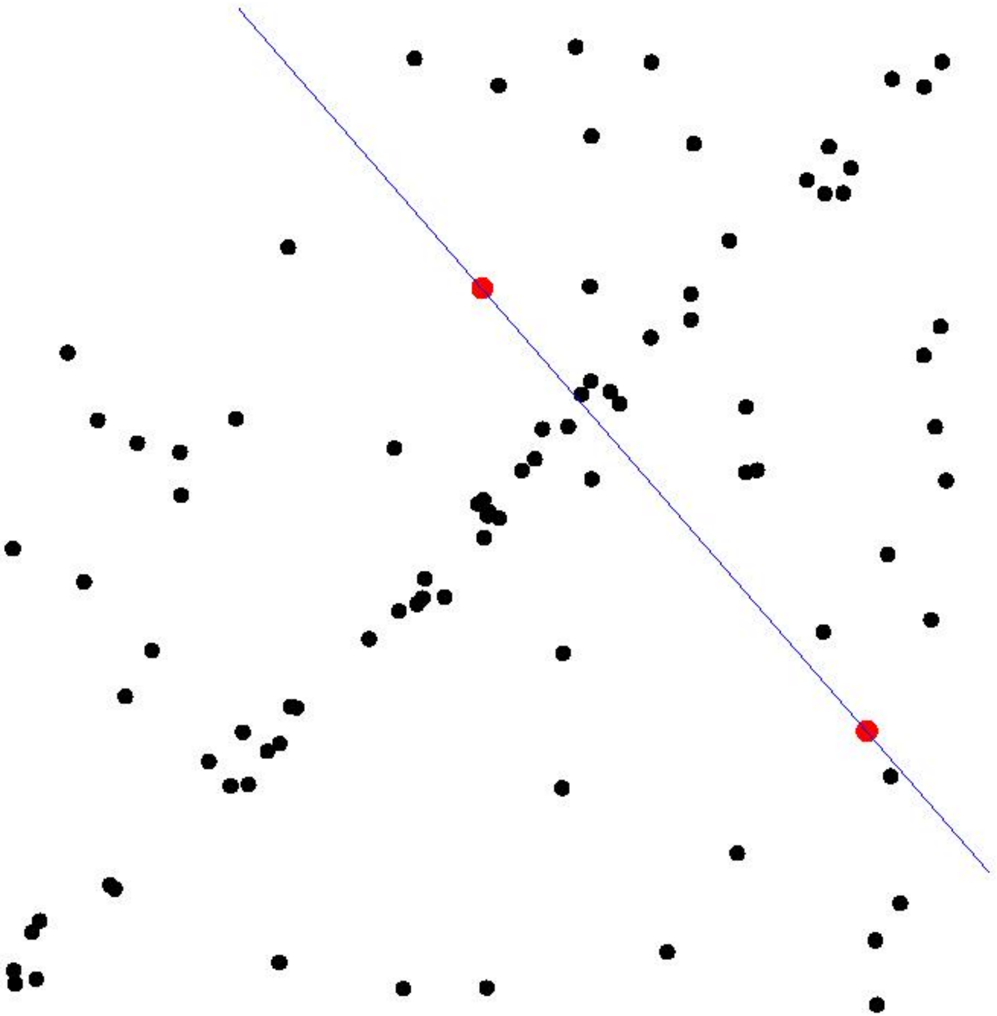
RANSAC

RANSAC



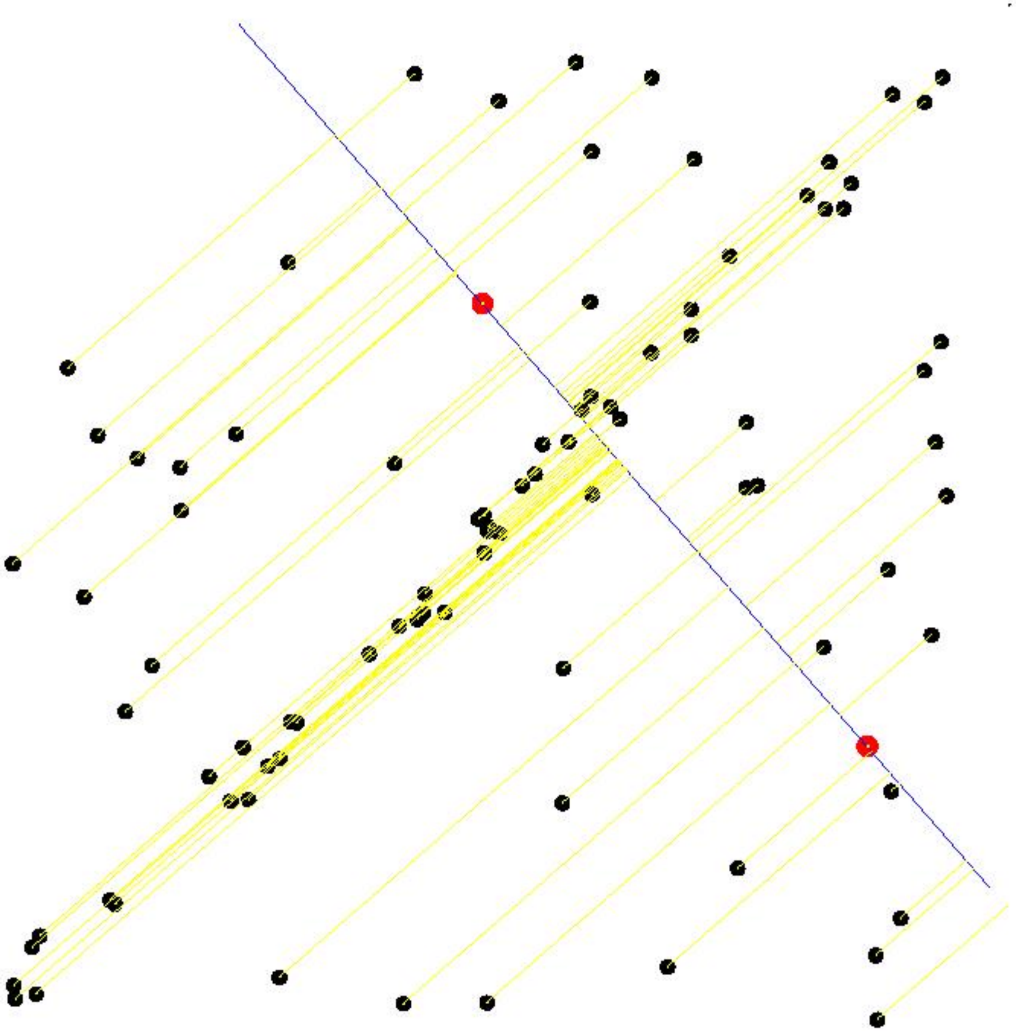
- **Select sample of m points at random**

RANSAC



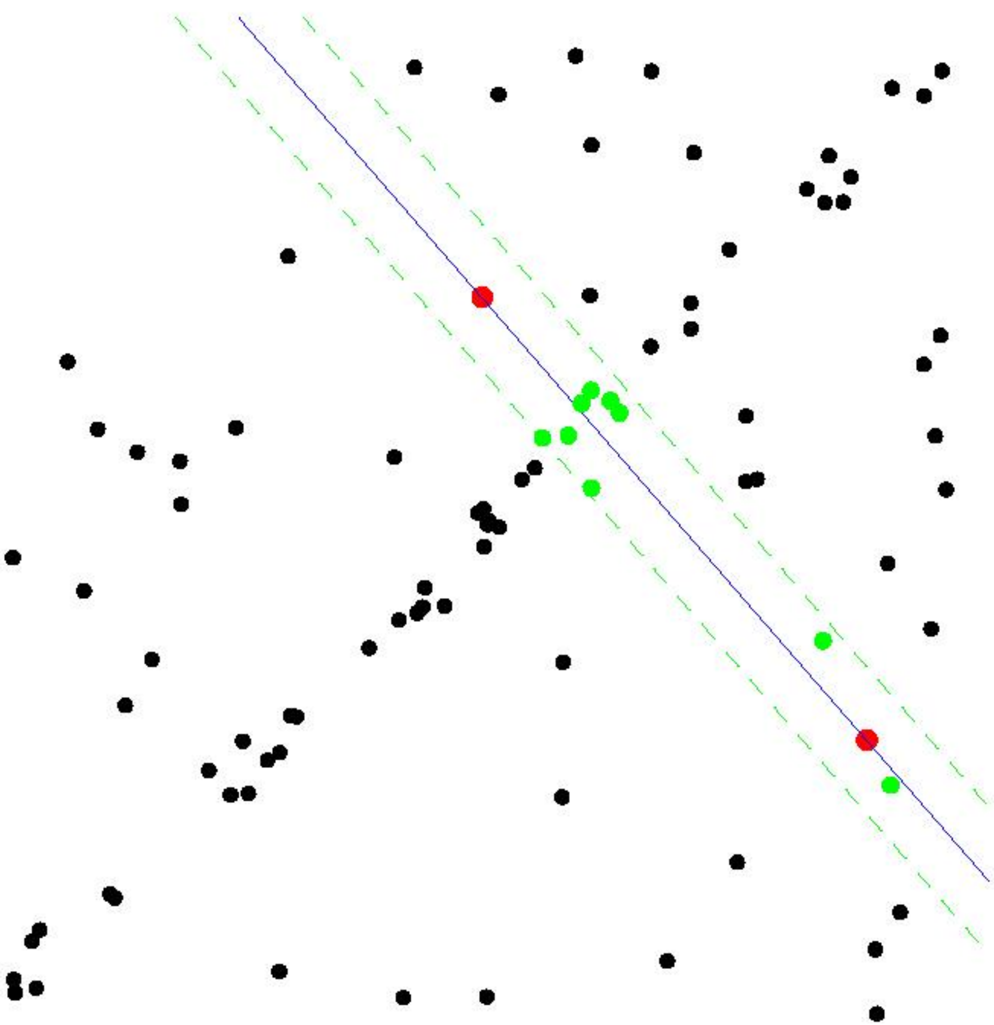
- Select sample of m points at random
- **Calculate model parameters that fit the data in the sample**

RANSAC



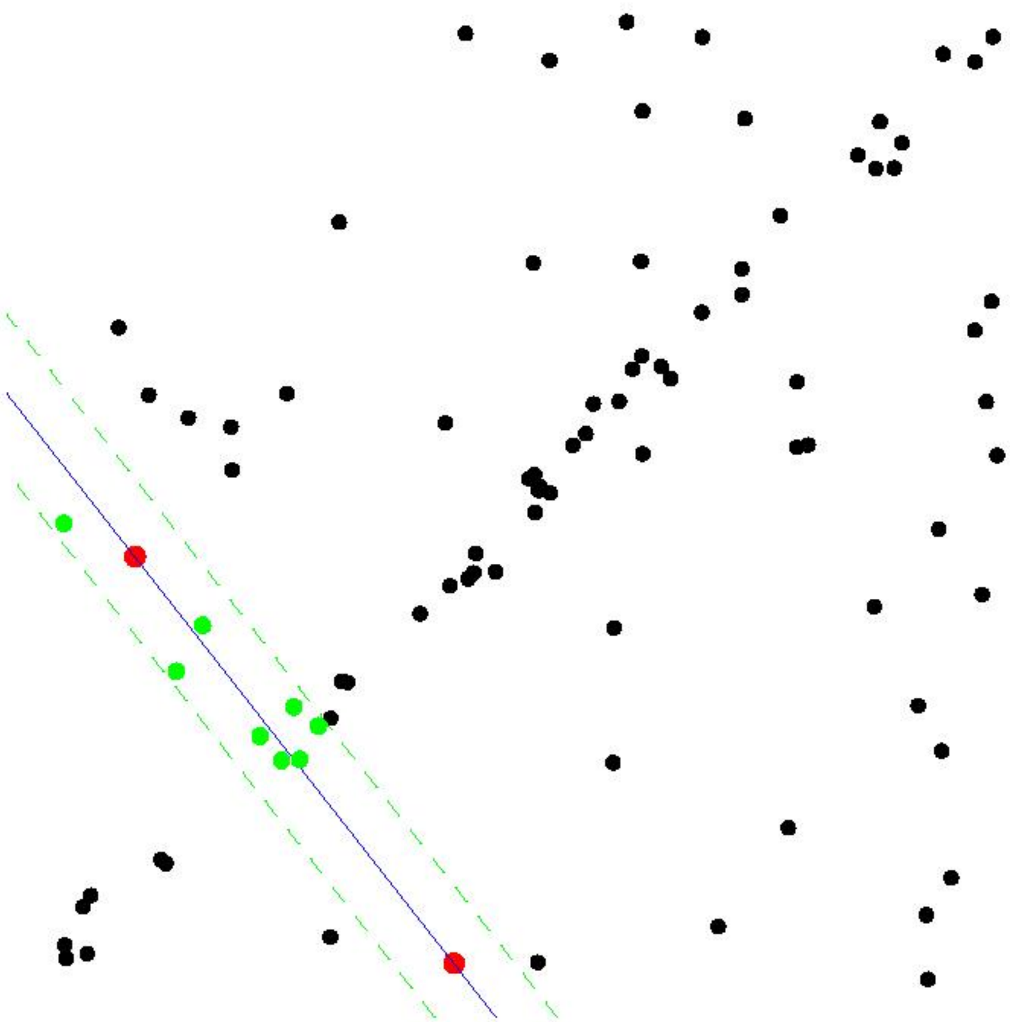
- Select sample of m points at random
- Calculate model parameters that fit the data in the sample
- **Calculate error function for each data point**

RANSAC



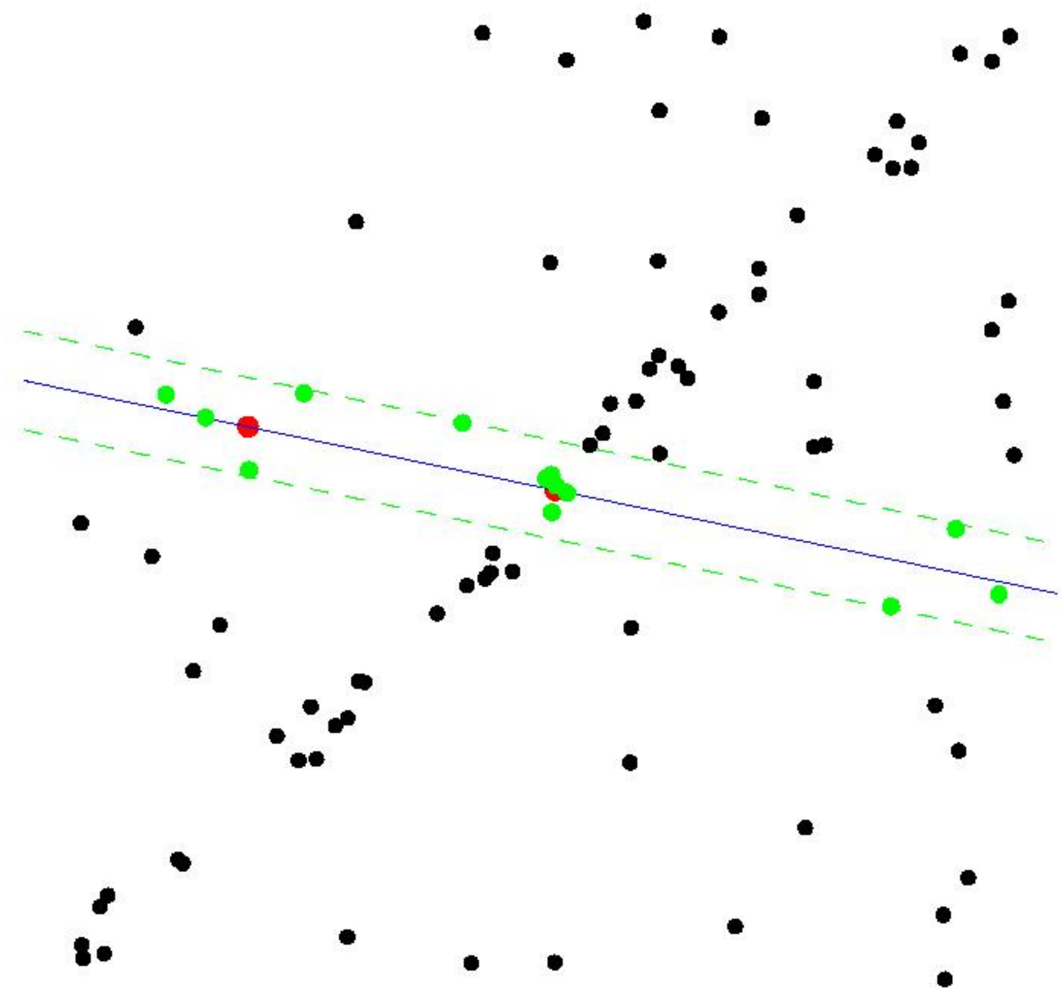
- Select sample of m points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- **Select data that support current hypothesis**

RANSAC



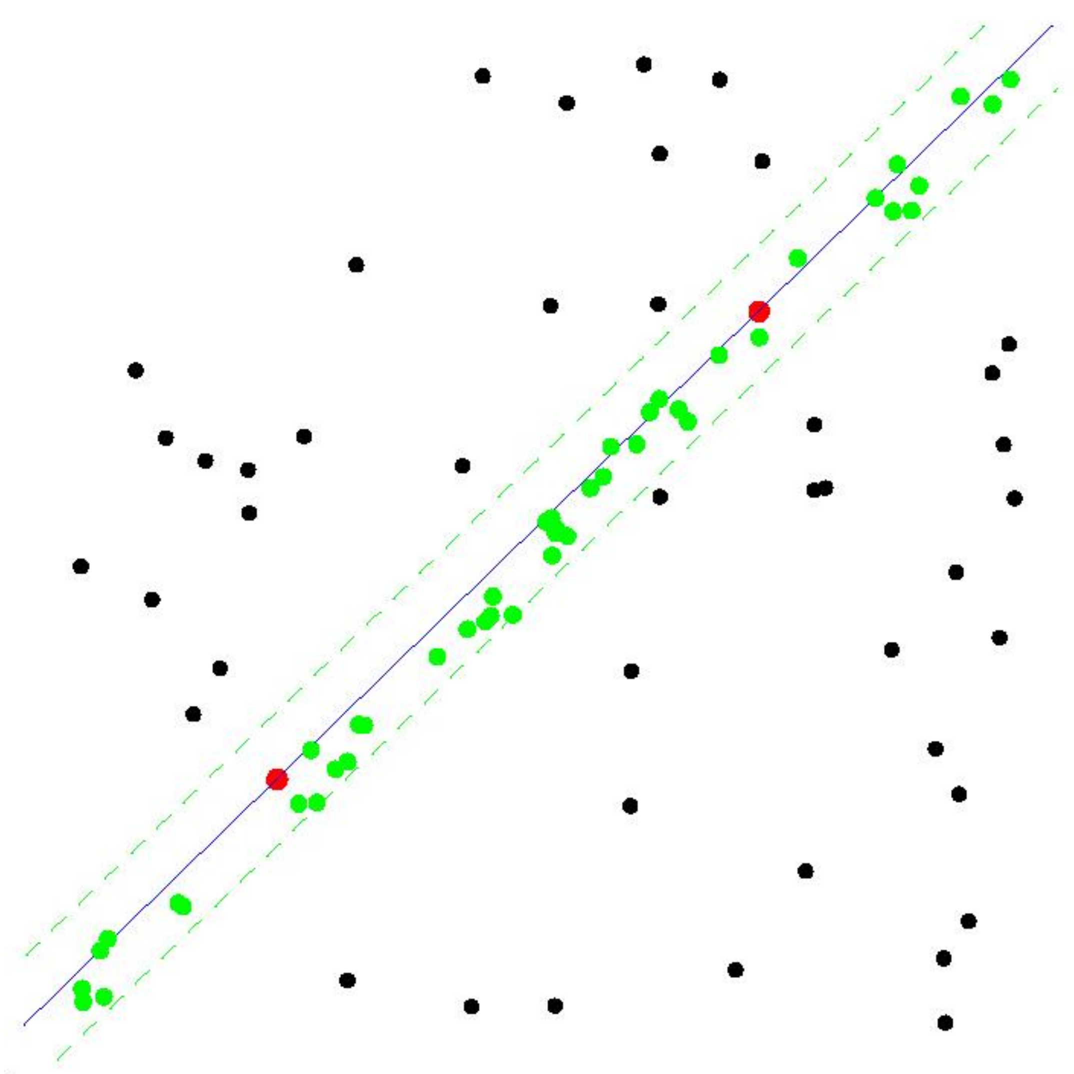
- Select sample of m points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- **Repeat sampling**

RANSAC



- Select sample of m points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that support current hypothesis
- **Repeat sampling**

RANSAC



ALL-INLIER SAMPLE

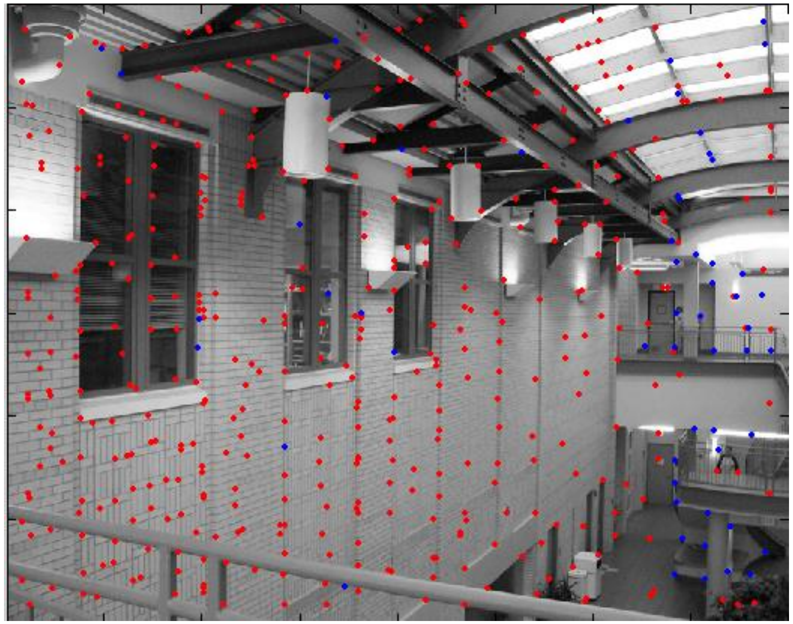
RANSAC time complexity

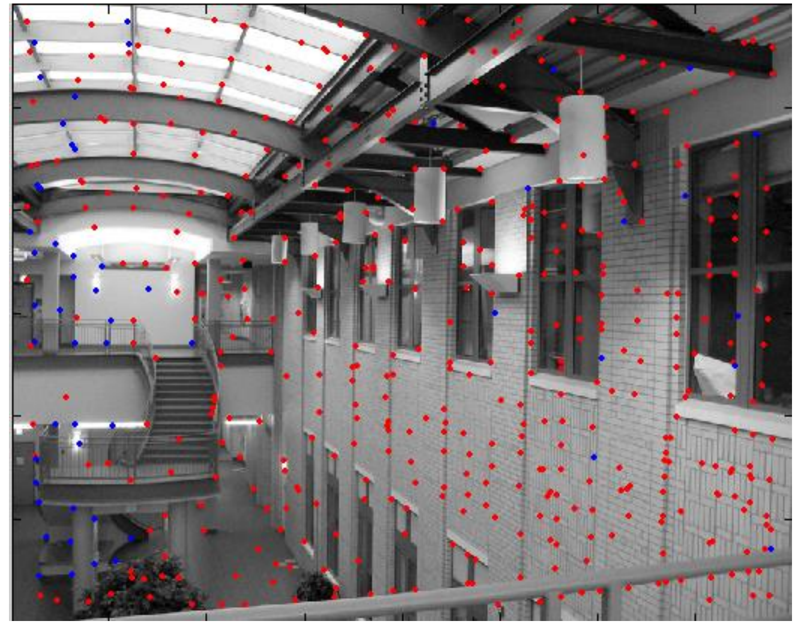
$$t = k(t_M + \overline{m}_s N)$$

k … number of samples drawn
N … number of data points
t_M … time to compute a single model
m_S … average number of models per sample

lbg@dongseo.ac.kr

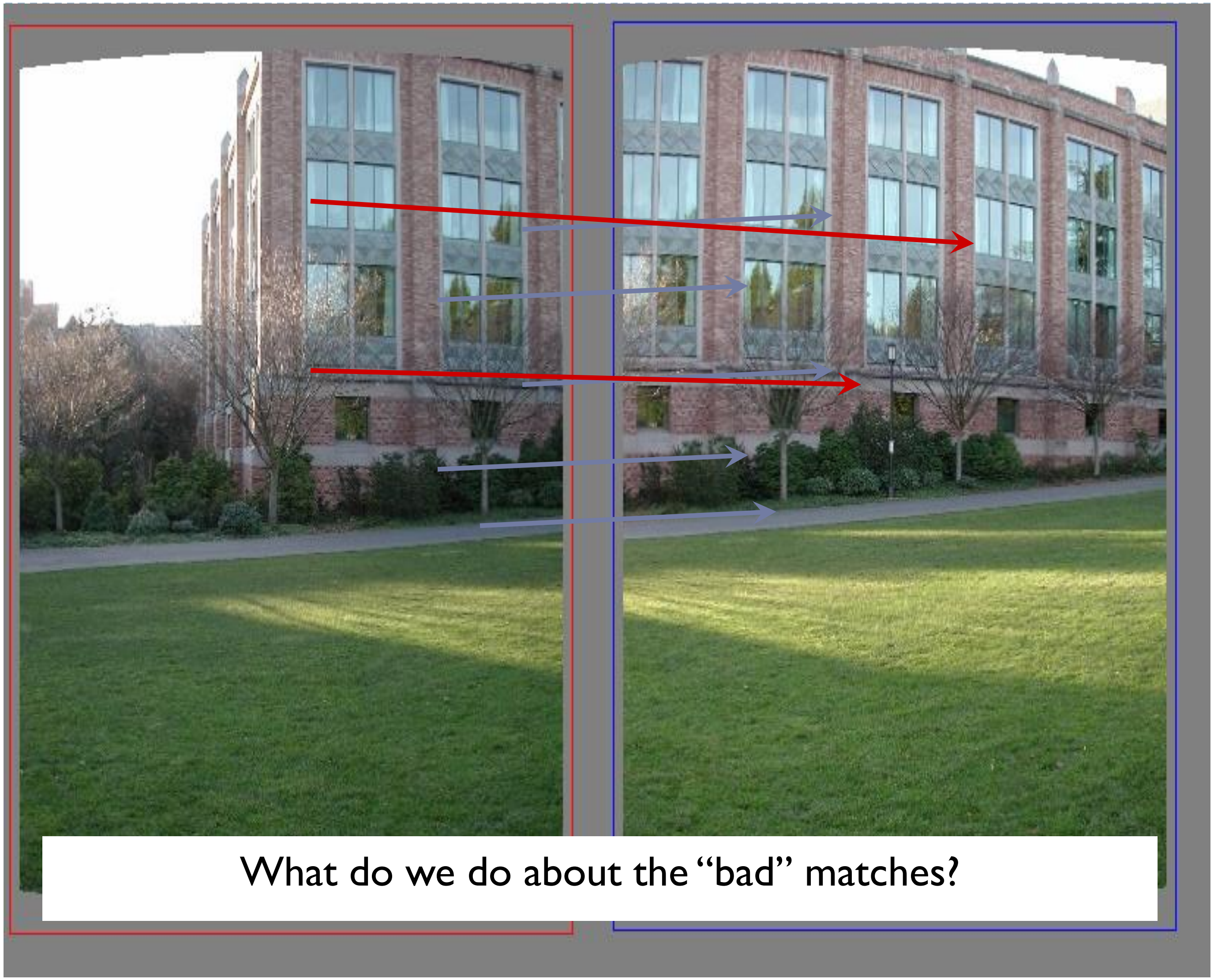
Feature-space outliner rejection



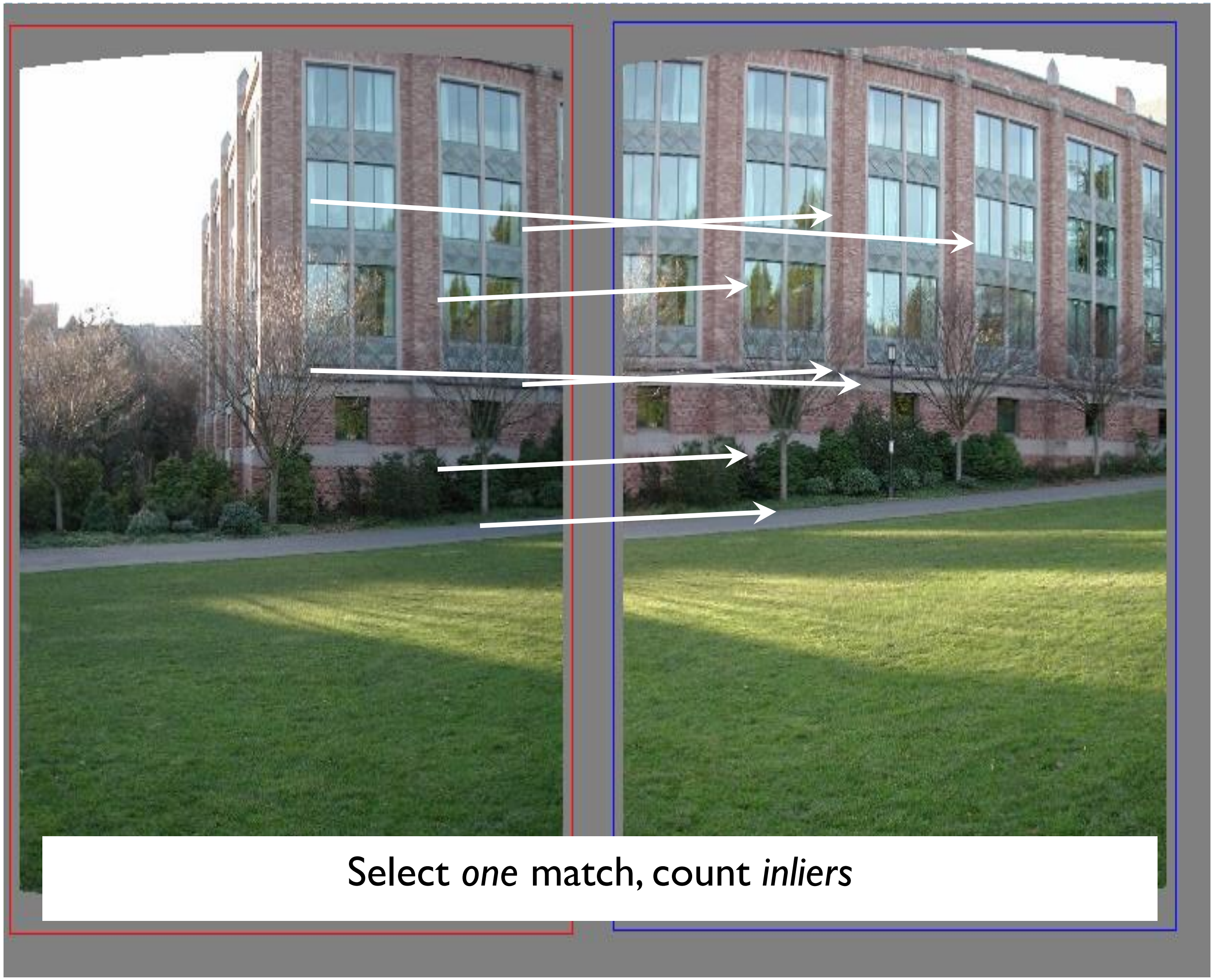


- Can we now compute H from the blue points?
  - No! Still too many outliers…
  - What can we do?

Matching features

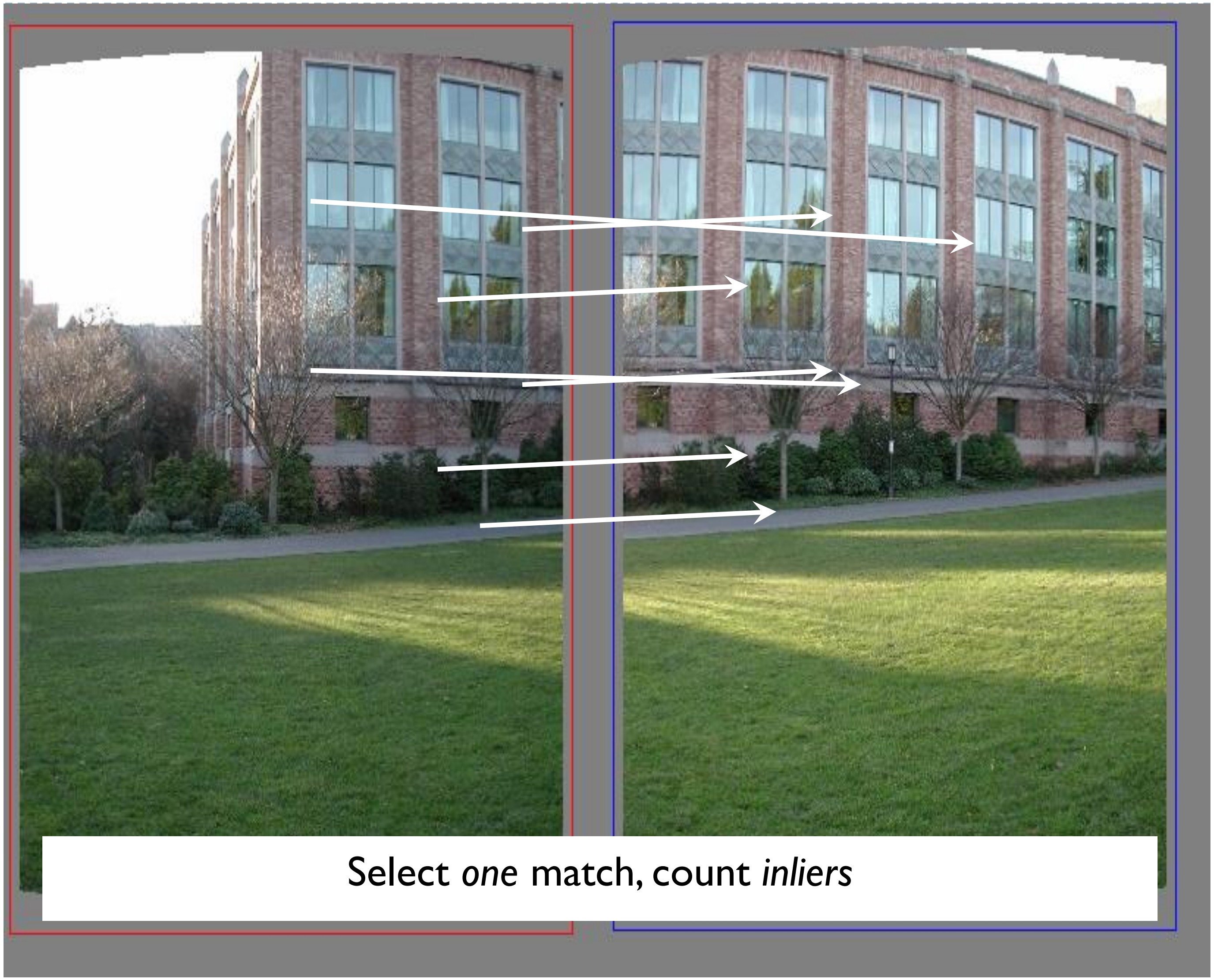


RAndom SAmple Consensus



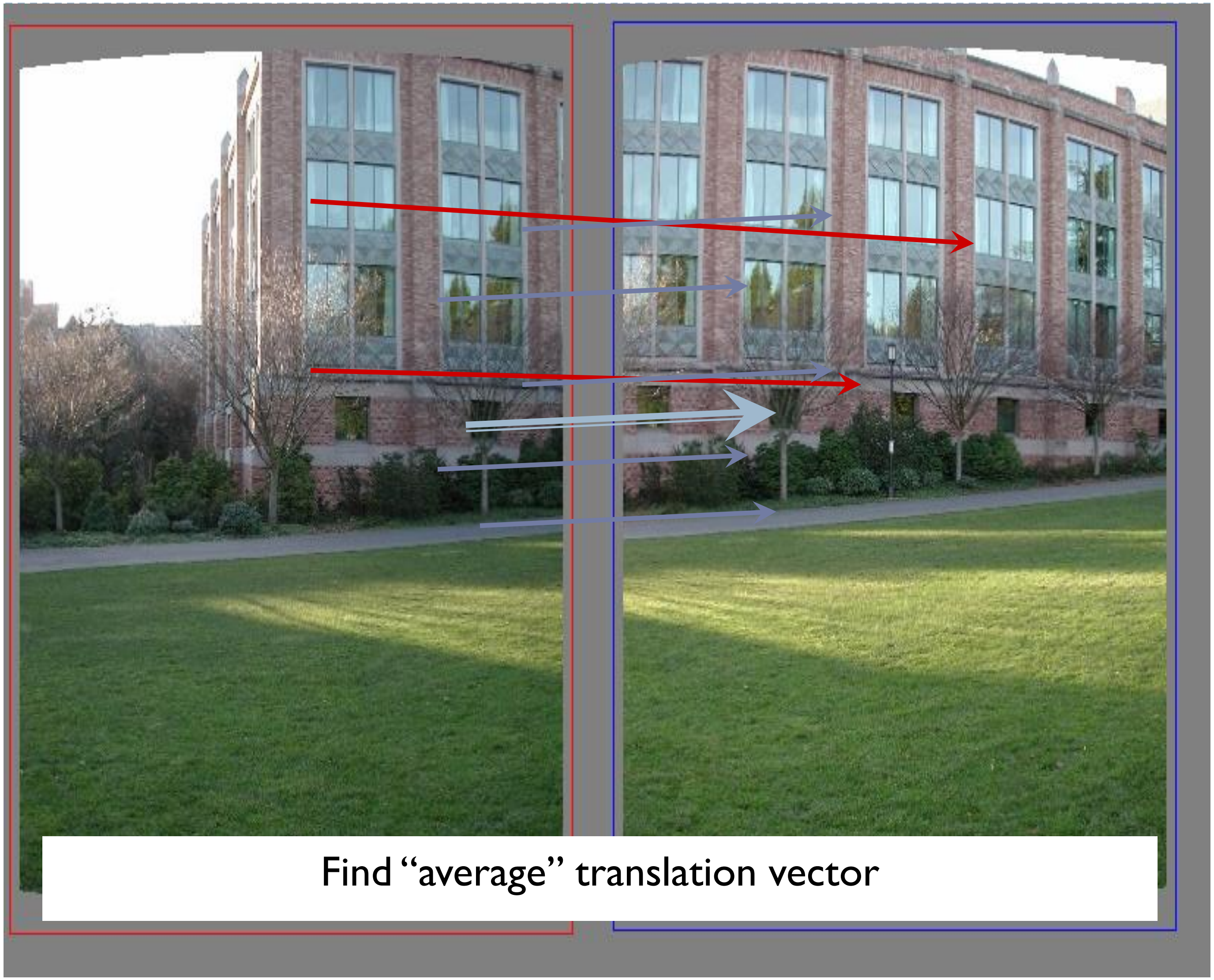
Select *one* match, count *inliers*

RAndom SAmple Consensus



Select *one* match, count *inliers*

Least squares fit



Find "average" translation vector

RANSAC for estimating homography

- RANSAC loop:

1. Select four feature pairs (at random)
2. Compute homography H (exact)
3. Compute *inliers* where $SSD(p_i', \boldsymbol{H}\, p_i) < \varepsilon$
4. Keep largest set of inliers
5. Re-compute least-squares H estimate on all of the inliers

RANSAC for Fundamental Matrix

Step 1. Extract features

Step 2. Compute a set of potential matches

Step 3. do

- Step 3.1 select minimal sample (i.e. 7 matches)
- Step 3.2 compute solution(s) for F

} (generate hypothesis)

- Step 3.3 determine inliers (verify hypothesis)

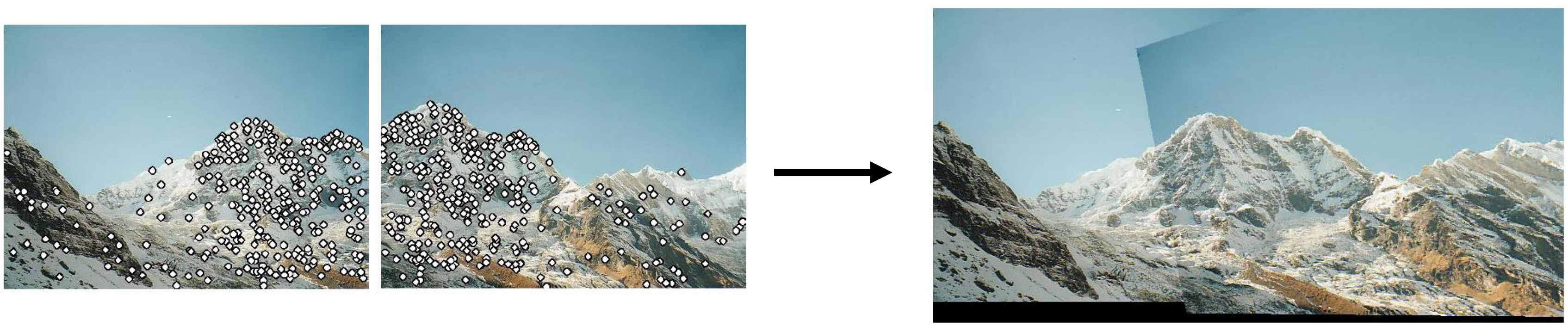
until Γ(#*inliers*,#*samples*)<95%

Step 4. Compute F based on all inliers

Step 5. Look for additional matches

Step 6. Refine F based on all correct matches

RANSAC



RANSAC

Example: Mosaicking with homographies

www.cs.cmu.edu/~dellaert/mosaicking

Recognizing Panoramas

Recognizing Panoramas

THANK YOU!!

	9/28	10/5	10/12	10/19		
Liu Chang	photosynth	create account Take pictures Construct photosynth		Search others photosynth works Select one point in DSU, take pictures		
Markus	Photomodeller	Select one small object Make 3d model data		Projector Price?		
Yang Yun	PhotoCity	Select one small object Make 3d model data				
Wang Ping	PhotoCity	Presentation file How to install & use Cygwin, bunder, … Matlab CC	Matlab CC	Matlab CC Lab Image Data SIFT Fundamental Matrix		
Liu Pengxin	Remove Perspective distortion	Load image Choose feature points SIFT & ASIFT OpenCV CC	OpenCV CC	Remove PT OpenCV CC PT		
Zhu Zi Jian		PhotoTourism ARToolkit Camera Calibration		PhotoTourism ARToolkit Camera Calibration Fundamental Matrix		
Wang Yuo		PhotoTourism		M C C – led lamp		

Topics in Image Processing

lbg@dongseo.ac.kr

Bayesian Modeling of Dynamic Scenes for Object Detection

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 27, NO. 11, NOVEMBER 2005
Yaser Sheikh, and Mubarak Shah
by lbg@dongseo.ac.kr 2009.02.25

Background $\varphi_b = \{y_1, y_2, \cdots, y_n\}, y = (r, g, b, x, y) \in \mathbb{R}^5$

$$P(x \mid \psi_b) = \frac{1}{n}\sum_{i=1}^{n} \varphi_H(x - y_i)$$

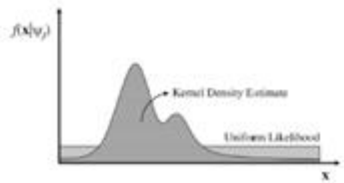
d-variate Gaussian density

$$\varphi_H^{(N)}(x) = |H|^{-1/2} (2\pi)^{-d/2} \exp(-\frac{1}{2} x^T H^{-1} x)$$

Foreground $\varphi_f = \{z_1, z_2, \cdots, z_m\}$

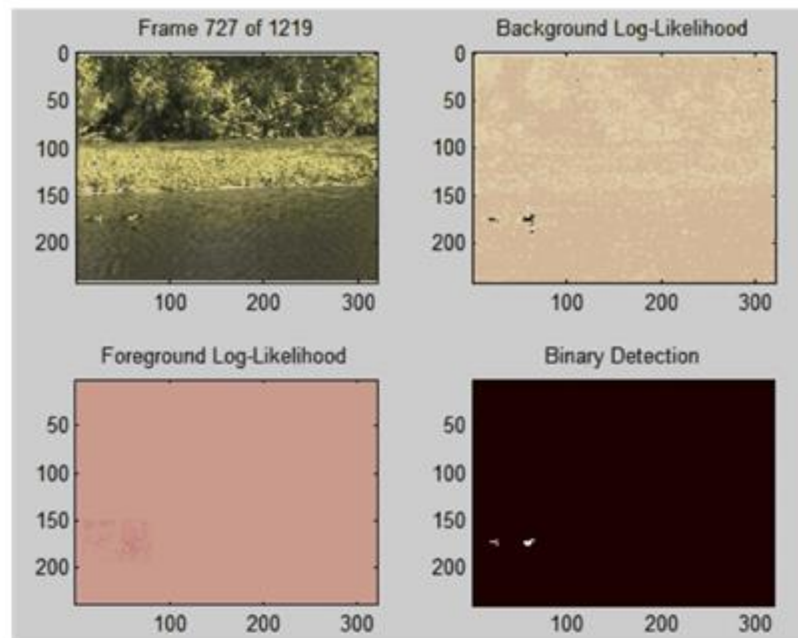
$$P(x \mid \psi_f) = \alpha\gamma + (1-\alpha) m^{-1} \sum_{i=1}^{m} \varphi_H(x - z_i)$$

Likelihood ratio classifier $\tau = -\ln \frac{P(x \mid \psi_b)}{P(x \mid \psi_f)}$



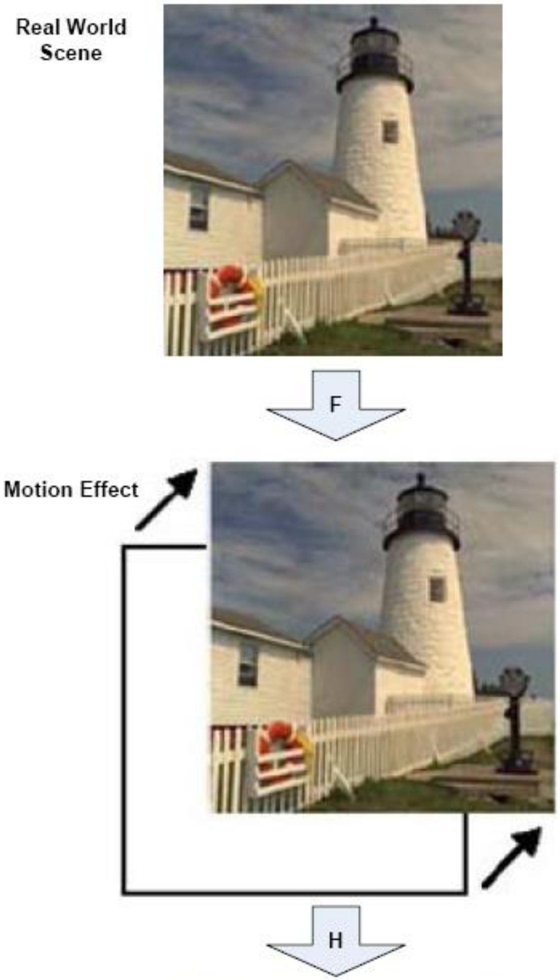
Algorithm
Initialize ψ_b using 1^{st} frame, $\psi_f = \emptyset$. At frame t, for each pixel,
Detection Step
1) Find $P(\mathbf{x}_i|\psi_f)$ (Eq. 7) and $P(\mathbf{x}_i|\psi_b)$ (Eq. 1) and compute the Likelihood Ratio τ (Eq. 8).
2) Construct the graph to minimize Equation 13.

Model Update Step
1) Append all pixels detected as foreground to the foreground model ψ_f.
2) Remove all pixels in ψ_f from ρ_f frames ago.
3) Append all pixels of the image to the background model ψ_b.
4) Remove all pixels in ψ_b from ρ_b frames ago.



Fast and Robust Multiframe Super Resolution

IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 13, NO. 10, OCTOBER 2004
Sina Farsiu, M. Dirk Robinson, Michael Elad, and Peyman Milanfar
by lbg@dongseo.ac.kr 2009.02.23



$$\underline{Y}_k = D_k H_k F_k \underline{X} + \underline{V}_k$$

$$\underline{\widehat{X}} = \underset{\underline{X}}{ArgMin}\left[\sum_{k=1}^{N}\left\|D_k H_k F_k \underline{X} - \underline{Y}_k\right\|_p^p\right] \qquad \underline{\widehat{X}} = \underset{\underline{X}}{ArgMin}\left[\sum_{k=1}^{N}\left\|D_k H_k F_k \underline{X} - \underline{Y}_k\right\|_p^p + \lambda\Upsilon(\underline{X})\right]$$

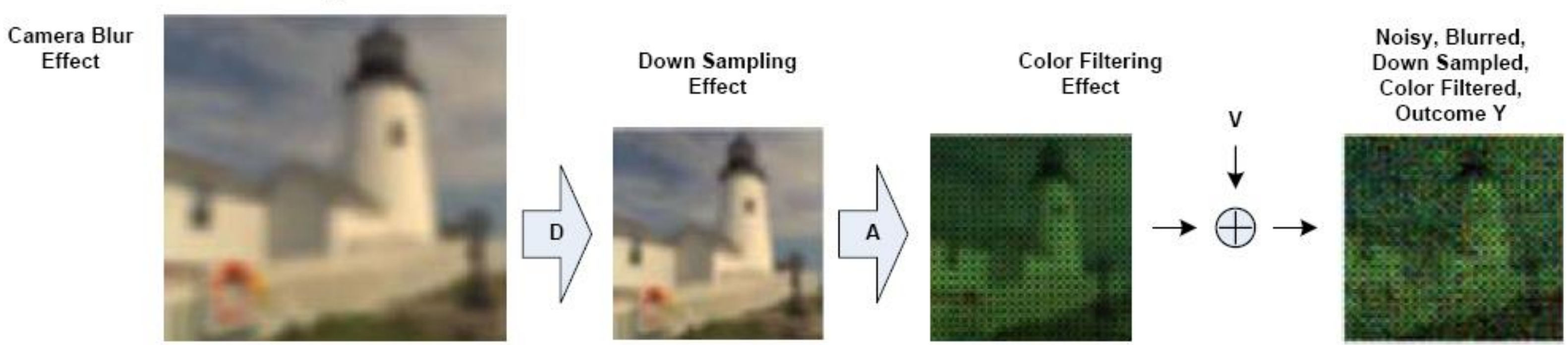
$$\Upsilon_T(\underline{X}) = \left\|\Gamma \underline{X}\right\|_2^2 \qquad \Upsilon_{TV}(\underline{X}) = \left\|\nabla \underline{X}\right\|_1 \qquad \Upsilon_{BTV}(\underline{X}) = \underbrace{\sum_{l=-p}^{p}\sum_{m=0}^{p}}_{l+m\geq 0} \alpha^{m+|l|}\left\|\underline{X} - S_x^l S_y^m \underline{X}\right\|_1$$

Robust Method

$$\underline{\widehat{X}}_{n+1} = \underline{\widehat{X}}_n - \beta\{\sum_{k=1}^{N} F_k^T H_k^T D_k^T sign(D_k H_k F_k \underline{\widehat{X}}_n - \underline{Y}_k) + \lambda \underbrace{\sum_{l=-P}^{P}\sum_{m=0}^{P}}_{l+m\geq 0} \alpha^{m+|l|}[I - S_y^{-m} S_x^{-l}]sign(\underline{\widehat{X}}_n - S_x^l S_y^m \underline{\widehat{X}}_n)\}$$

Fast Robust Method

$$\underline{\widehat{X}}_{n+1} = \underline{\widehat{X}}_n - \beta\{H^T A^T sign(AH\underline{\widehat{X}}_n - A\underline{\widehat{Z}}) + \lambda \underbrace{\sum_{l=-P}^{P}\sum_{m=0}^{P}}_{l+m\geq 0} \alpha^{m+|l|}[I - S_y^{-m} S_x^{-l}]sign(\underline{\widehat{X}}_n - S_x^l S_y^m \underline{\widehat{X}}_n)\}$$



Fast Motion Deblurring

Sunghyun Cho
POSTECH

Seungyong Lee
POSTECH

Input blurred image Deblurring result Magnified views

Figure 1: *Fast single image deblurring. Our method produces a deblurring result from a single image very quickly. Image size:* 713×549. *Motion blur kernel size:* 27×27. *Processing time:* 1.078 *seconds.*

Abstract

This paper presents a fast deblurring method that produces a deblurring result from a single image of moderate size in a few seconds. We accelerate both latent image estimation and kernel estimation in an iterative deblurring process by introducing a novel prediction step and working with image derivatives rather than pixel values. In the prediction step, we use simple image processing techniques to predict strong edges from an estimated latent image, which will be solely used for kernel estimation. With this approach, a computationally efficient Gaussian prior becomes sufficient for deconvolution to estimate the latent image, as small deconvolution artifacts can be suppressed in the prediction. For kernel estimation, we formulate the optimization function using image derivatives, and accelerate the numerical process by reducing the number of Fourier transforms needed for a conjugate gradient method. We also show that the formulation results in a smaller condition number of the numerical system than the use of pixel values, which gives faster convergence. Experimental results demonstrate that our method runs

nature of imaging sensors that accumulate incoming lights for an amount of time to produce an image. During exposure, if the camera sensor moves, a motion blurred image will be obtained.

If a motion blur is shift-invariant, it can be modeled as the convolution of a latent image with a motion blur kernel, where the kernel describes the trace of a sensor. Then, removing a motion blur from an image becomes a deconvolution operation. In *non-blind* deconvolution, the motion blur kernel is given and the problem is to recover the latent image from a blurry version using the kernel. In *blind* deconvolution, the kernel is unknown and the recovery of the latent image becomes more challenging. In this paper, we solve the blind deconvolution problem of a single image, where both blur kernel and latent image are estimated from an input blurred image.

Single-image blind deconvolution is an ill-posed problem because the number of unknowns exceeds the number of observed data. Early approaches imposed constraints on motion blur kernels and used parameterized forms for the kernels [Chen et al. 1996; Chan and Wong 1998; Yitzhaky et al. 1998; Rav-Acha and Peleg

YOUTECH
40-Years Experienced

"Defense of East Sea for Coast & Coastline" in Korea
(120-Guard soldiers take off by one Robot PTZ system: 40x3times=120)

"Builtin special software program for intruder moving image detection/tracking & recording/alarm each preset zones by motion image analyzer automatically"

- **20 Preset zone x3 Group (Total 60 Preset Zones)**
- **Operation by 8-Army group for "Defense of East Sea"**

IntelliVIX-FarSight I/II

YOUTECH
40-Years Experienced

(Intelligent Robot PTZ Camera System)

- 360˚ Endless high speed heavy duty Pan/Tilt driver with Day-20km,15km,6km,3km,2km
- Motorized Zoom true color/Night-10M~10KM(30,000Ft) image pick-up by 56-strong "IR"illuminator with collimator & Laser "IR" illuminator at night (IP66, 20kg) (Patent Pending)
- Integrated with advanced video analytics algorithm (Patent Pending)
- Intelligent use of PTZ Cameras: Preset Touring, Auto PTZ Tracking (Patent Pending)

Robot camera move to intruder detecting area with recording/alarm to use Defense of Borderline, Coast & Coastline, airport, DAM, Weapon area, Oil pumping area, Military zone with intruders detecting sensor (N.C or N.O)

PATENT PENDING

- **Built-in 1/2" sony EX-View or 1/2" EM-CCD extreme sensitivizzty color camera with D/N filter & Sens-up to take 0.00001 or 0.000005 lux ultra sensitivity at night**
- **Built-in Menu display & adjust of privacy mask, WDR, motion detector, flickerless, sens-up, to take 0.00001 lux ultra sensitivity at night (0.000005 lux by 4000WAMTFH Model)**

- **It is long range a Robot PTZ Camera system**
  1) **Super long range IVS Robot system; Day-20km true color/Night-10km human detection**
     - YPZ-5000WS-4000WAMTFH-101000/2020000 (+) YIO-200-IP (+) (+) 2-YIL-5000FH-28SS (+) 2-YLI-4KM (+) MDT-5000WS / **Intellivix–FarSight I software**
  2) **Long Range IVS Robot system; Day-15km true color/Night-8km human detection**
     - YPZ-5000WS-4000WAMTFH-12750/251500 (+) YIO-200-IP (+) 2-YIL-5000FH-28SS (+) 2-YLI-4KM (+) MDT- 5000WS / **Intellivix – FarSight I software**
  3) **Medium range IVS Robot system to use defense of borderline, military zone, etc; Day-3km true color/night 3km human detection**
     - YPZ-5000WS-3000WAMTFH-10330 (+) YIO-200-IP (+) 2-YIL-5000FH-28SS (+) YLI-4KM (+) MDT-5000WS / **Intellivix – FarSight I software**
- **We have 2 special software program to make IVS, Robot PTZ camera system**

1) IntelliVIX-FarSight I: Software program to use one Robot PTZ Camera System
   - More than 20-Intruders motion detection/Tracking/Recording /Alarm each preset zones by intruder motion video analyzing within one sec. each preset zones or stop of 360deg. PT driver positions.
   - MAX. 20-Preset zones day time still image display on the bottom of the monitor screen after fixed preset zones & If Robot PTZ move to any preset zone, A matching image pop-up by RED color marking around still image to easy check of all intruders at Day time or Night time

IVS: Intelligent Video Surveillance

IntelliVIX-FarSight III

YOUTECH
40-Years Experienced

(Intelligent Panoramic Video Surveillance System)

PVX-180T-50/100DN minimize a blind spot of a camera to observe the surrounding wide area by one shot acquired with 180˚ video pictures. **IntelliVIX-FarSight III** makes a seamless live panorama video pictures from the video pictures of **PVX-180T-50/100DN/1000DN** and obtains enlarged pictures of the objects tracking with close-up image to identifications of the face, license plate manually/automatically by an integrated PTZ camera that auto-sense the motions of people or cars, etc.

※ PVX-180T-1000DN: Human detecting distance 1km(3,000ft) (Day-Night)

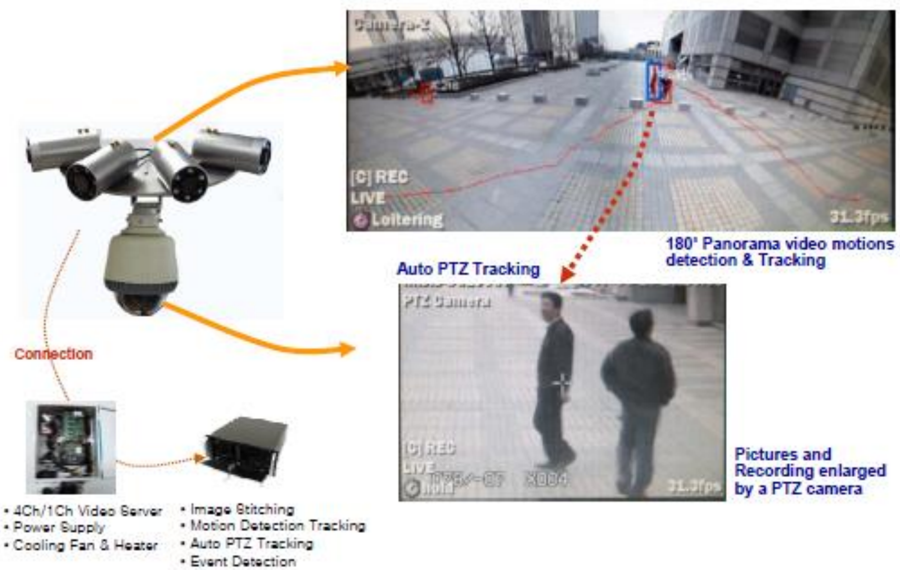
PATENT PENDING

PVX-180T-50DN

PVX-180T-100DN

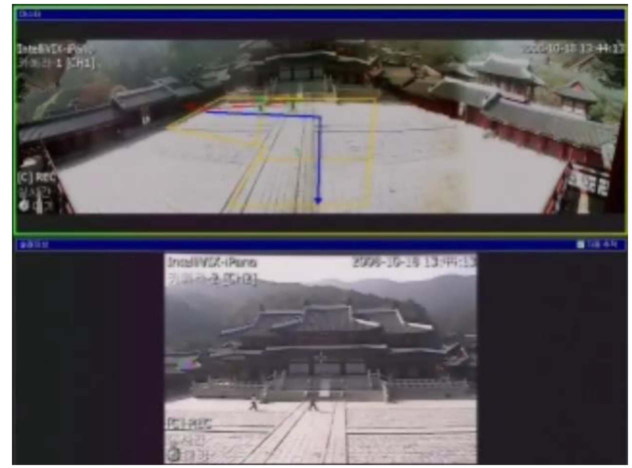
PVX-180T-50DN
- **1/3" Sony Ex-View Color CCD, Motorized Day & Night Changing Filter, 4-Day-Night Camera with IR illuminator**
- 1/4" Sony Super-HAD CCD Auto Zoom Camera, 1.0 Lux(Day), 0.00 Lux(Night) by 49-Hybrid IR Modules of Panorama Cameras, F 1.6 3.5~91mm(26x), Auto/Day/Night(IR Cut Filter)
- **Human detecting distance: 50m(165 Ft)(Day/Night)**

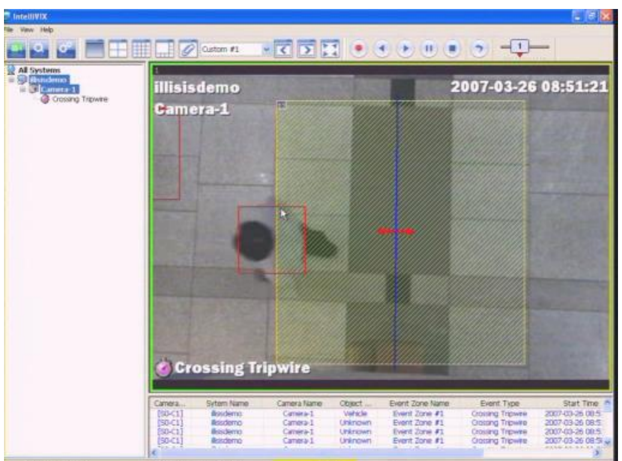
PVX-180T-100DN
- **1/3" Sony Ex-View Color CCD, Motorized Day & Night Changing Filter, 4-Day-Night Camera with IR illuminator(+)3-Strong IR illuminator**
- 1/4" Sony Super-HAD CCD Auto Zoom Camera, 1.0 Lux(Day), 0.00 Lux(Night) by 49-Hybrid IR Modules of Panorama Cameras, F 1.6 3.5~91mm(26x), Auto/Day/Night(IR Cut Filter)
- **Human detecting distance: 100m(328 Ft)(Day/Night)**

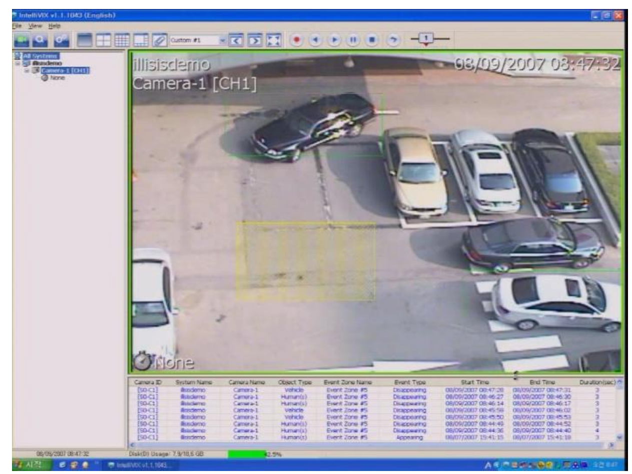


illisis


IntelliVIX: Powerful Video Analytics Solution
An Introduction to
IntelliVIX-SDK
for Windows
illisis Inc.
(www.illisis.com)
Copyright © 2007 illisis Inc. All rights reserved. Proprietary information for official use only. Do not distribute.




illisisdemo
Camera-1
2007-03-26 08:51:21
Crossing Tripwire


illisisdemo
Camera-1 [CH1]
08/09/2007 08:47:32
None

LYYN


LYYN
we give you a clearer vision
Want to see
what we see?
Home
Showroom
Live lyynified camera
Applications
Fog, Mist and Haze
Smoke
Rain
Snow
Dust
Lowlight
Underwater
Industries
SubSea
Commercial diving
Security & Surveillance
Aviation
Maritime & Harbor
Security
Traffic management
User stories
LYYN Real-time Video Enhancement
Want to see
what we see?
Click to view
Newsletter


(C) LYYN AB 2008
(C) LYYN AB 2005
(C) LYYN AB 2005

LYYN is an R&D company working with image enhancement for improving visibility in different industries. Behind the company's technology lies many years of research in the human vision system and image technologies. LYYN offers products and solutions based on a technical platform, V.E.T., Visibility Enhancement Technology. The platform works with digital still images and video from common color cameras, in real time, but also in post processing of stored material. V.E.T. improves visibility in for instance fog, haze, snow, rain, dust, darkness, etc. as well as in sub sea and medical applications. For examples please visit www.lyyn.com.

LYYN
We Give You A Clearer Vision

LYYN T38™

REAL-TIME IMAGE ENHANCER

FOR ANALOG VIDEO

LYYN T38™ is the solution to your visibility problems in fog, haze, smoke, dust, rain, low-light and many more situations.

The true power of LYYN T38™ is that the visibility enhancement is done in real-time in a live video stream. And you can also use it on stored material. Use the LYYN T38™ to enhance the feed from a surveillance camera in a video security system. Or to enhance a video tape in a camera brought back to office after some police fieldwork. Or wherever you need to see more with video.

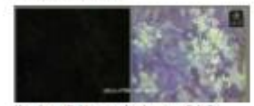
The LYYN T38™ is easy to use. Connect it in-between the camera and a monitor, tv set or vcr and you are up and running. With controls for enhancement level and size and position of a rectangular selection of the scene you can always make sure to catch the object on film, and see it.

If you need visibility aid in the field you connect the LYYN T38™ in-between your camera and an external screen. The screen will show enhanced visibility real-time, helping you zooming and focusing on the right object.

The pictures speak for themselves.
In any situation where you need a clear view, you need LYYN T38™.

Security enhancement

A misty morning on the road
- not so misty anymore



How beautiful is a garden in moon light?

The murky waters of the Baltic Sea are suddenly more agreeable

Find the New Yorkers in the blizzard...

...or some soldiers in a sand storm in Iraq

In large and in small, even as small as in a microscope, what ever your need is, LYYN V.E.T. provides a better view

CONTACT INFORMATION

LYYN AB
Ideon Science Park
SE-223 70 Lund, Sweden
Phone: +46 46 286 57 90
info@lyyn.com
www.lyyn.com

LYYN
We Give You A Clearer Vision

LYYN T38™

REAL-TIME IMAGE ENHANCER
FOR ANALOG VIDEO

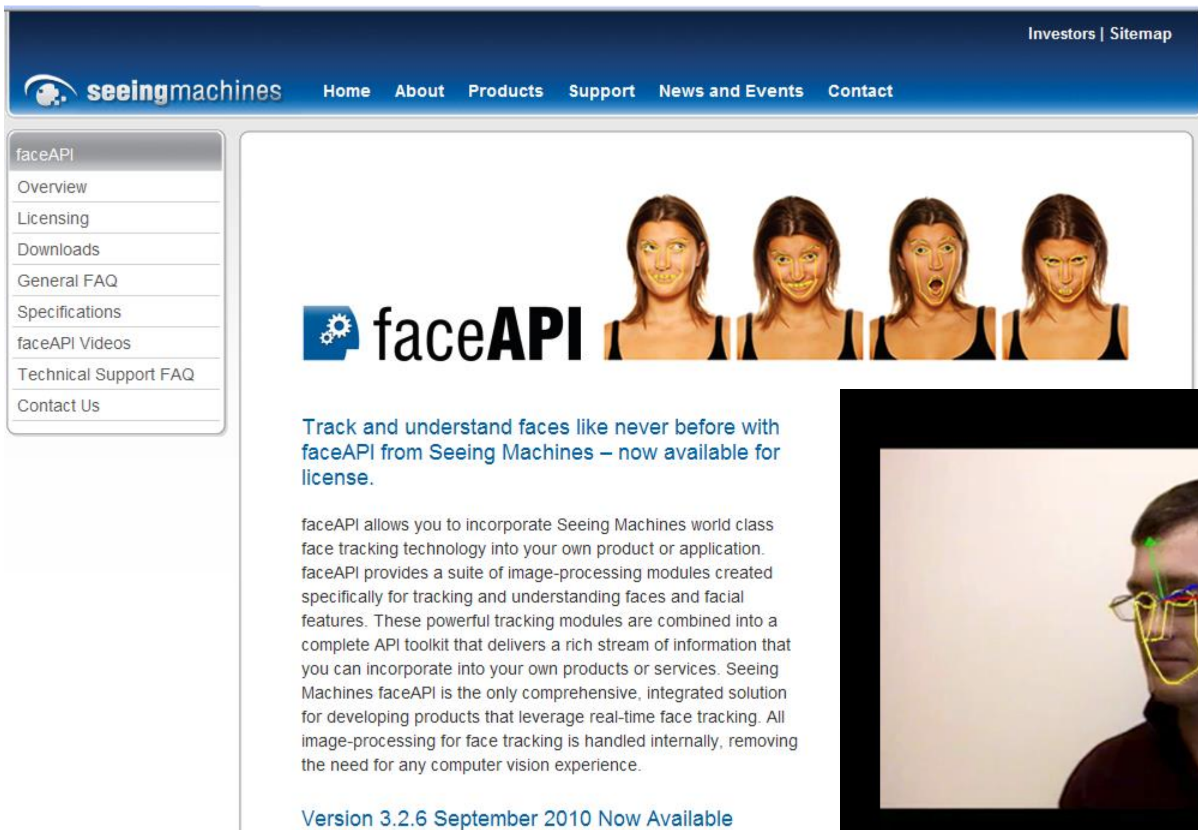
Easy to use and connect

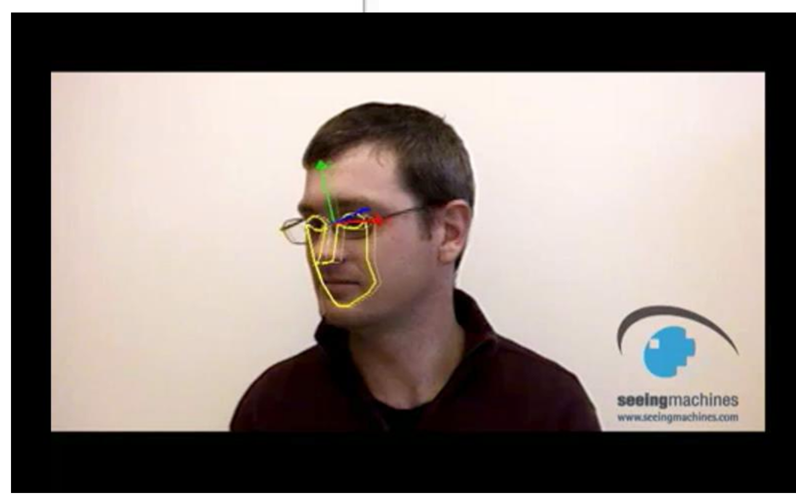
Easy to use real-time enhancement processing

Model	LYYN T38™
Supported video formats	Analog PAL and NTSC (Time Base Correct, Internal Sync)
Supported audio formats	n/a
Frame rate	PAL: 25 fps, NTSC: 30 fps
Sources	Real-time live video feed Stored material, e.g. from VHS player, video camera, etc
Buffering	1 frame
Delay	PAL: 1/25 second, NTSC: 1/30 second
Image settings	Lyynification level Rectangular selection, size and position
Connectors	BNC analog video, NTSC/PAL auto sensing
Casing	Metal casing. Standalone.
Power	12V DC, 0.3A
Operating conditions	5-50°C (41-122 °F) Humidity 20-80% RH (non-condensing)
Included Accessories	Power supply 12V DC, 12V car plug 2 Video cables, BNC connectors
Approvals	CE
Dimensions	54x172x184 (HxWxD in mm)
Weight	925g, excl. power supply

faceAPI

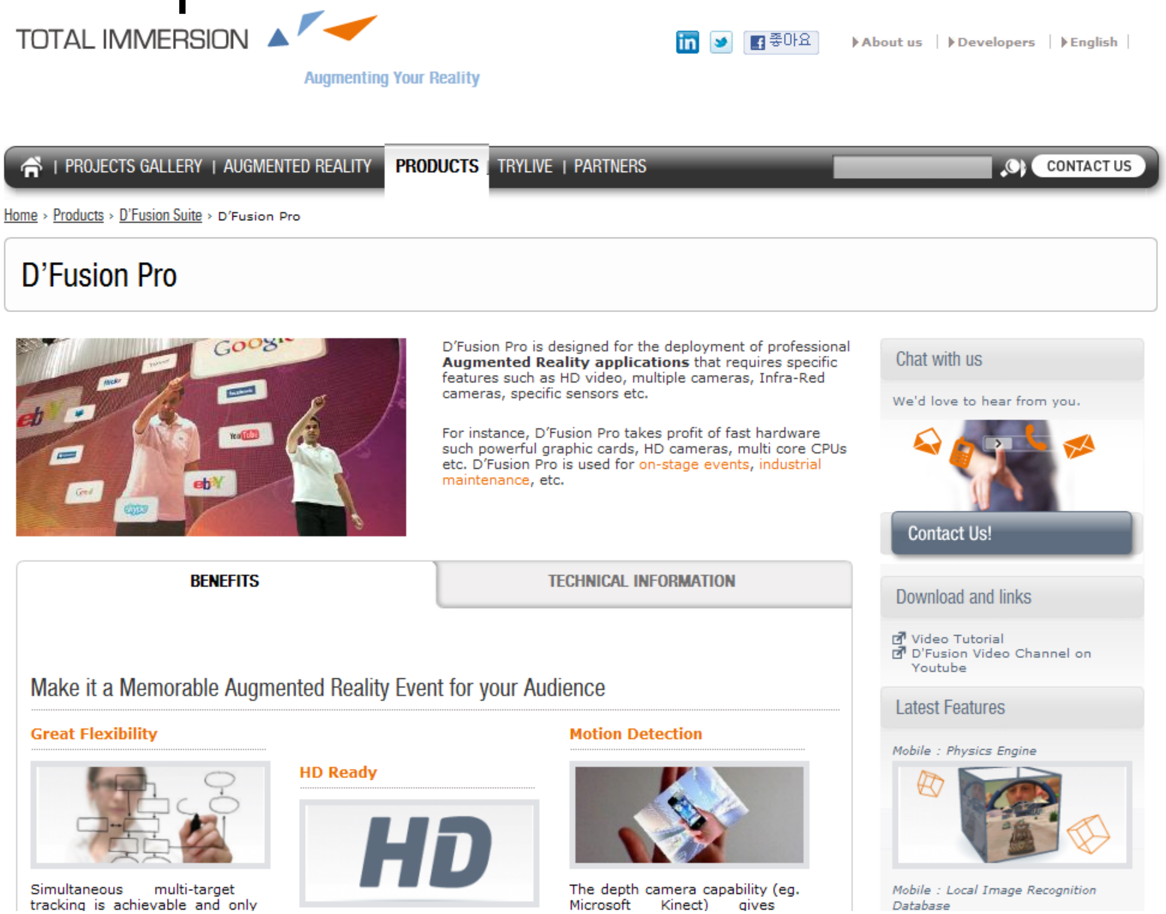
- http://www.seeingmachines.com/product/faceapi/



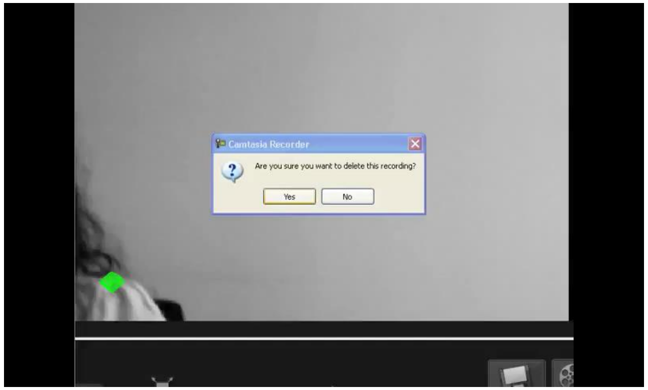


Total Immersion - D'Fusion Pro

- http://www.t-immersion.com/products/dfusion-suite/dfusion-pro



D'Fusion Pro - Markless Tracking

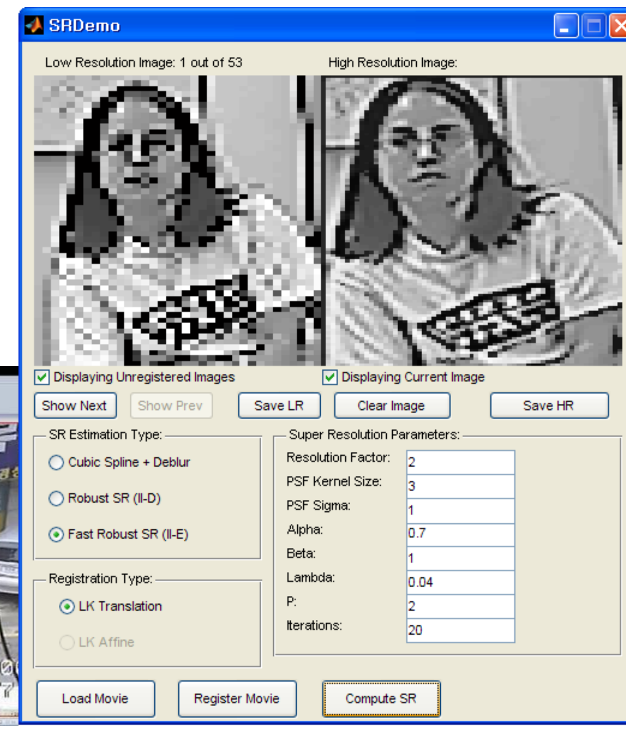


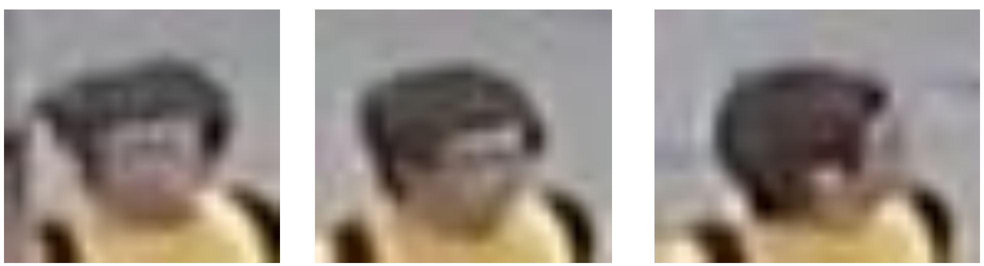

11/07/07
17:26:19
http://www.busunggroup.co.kr
TEL:628-3477 FAX:628-3538
1980-01-07
02:06:49.194
CAM ID:001
144.6 17.7

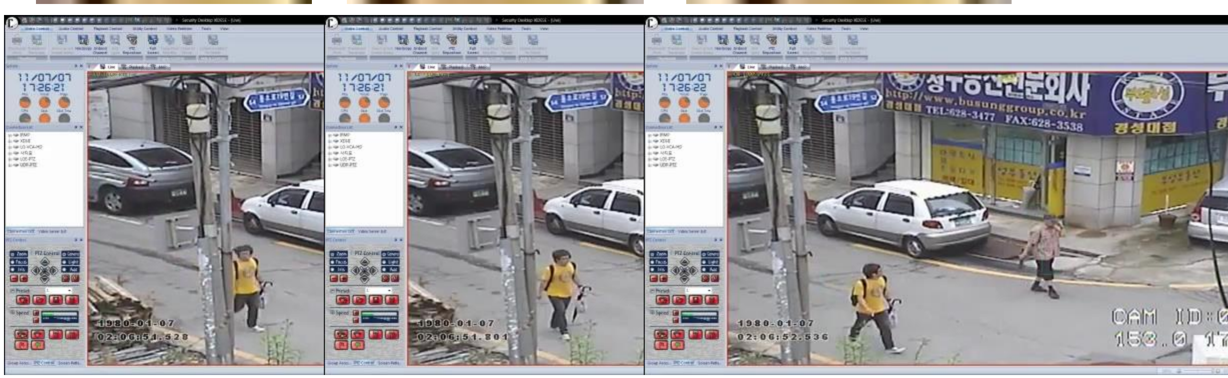
Monitoring System

Scenarios

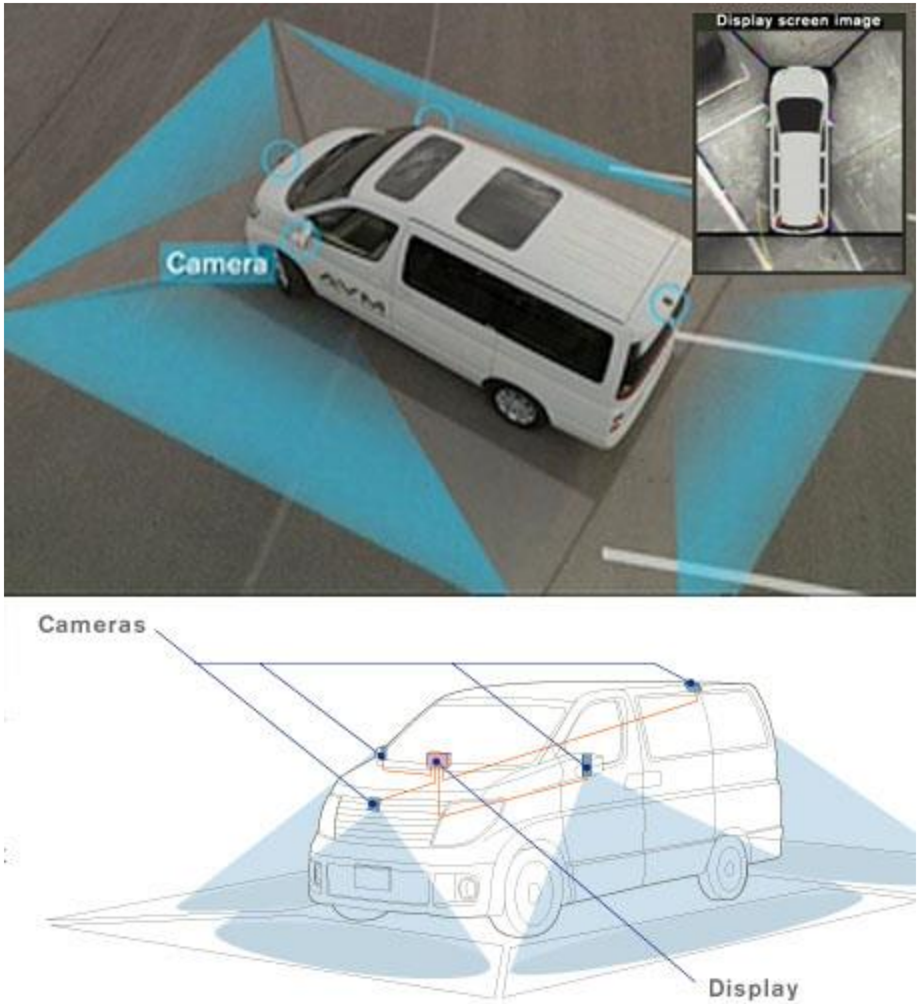
- Clear Vision - Denoising
- Motion Deblurring
- SuperResolution
- Panoramic View
- Background Modeling







Around View Monitor

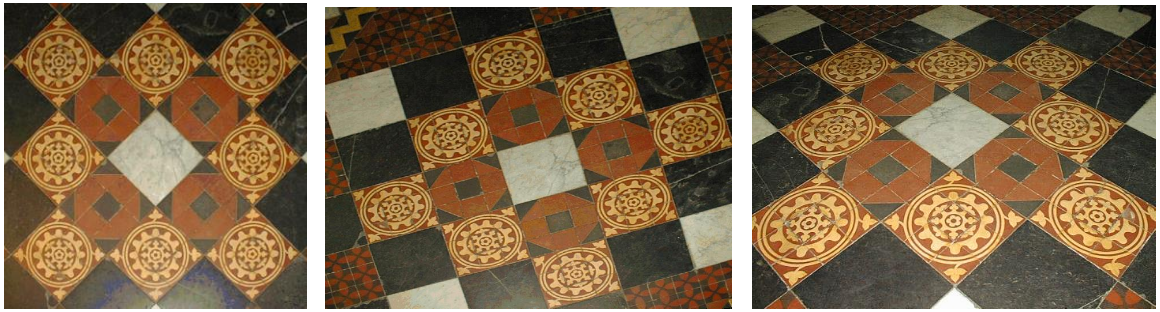


Parking Guide Function

Projective Transformations

lbg@dongseo.ac.kr

Projective 2D Geometry



Projective transformations

Definition:

A *projectivity* is an invertible mapping h from P^2 to itself such that three points x_1, x_2, x_3 lie on the same line if and only if $h(x_1), h(x_2), h(x_3)$ do.

Theorem:

A mapping $h: P^2 \rightarrow P^2$ is a projectivity if and only if there exist a non-singular 3x3 matrix **H** such that for any point in P^2 reprented by a vector **x** it is true that $h(\mathbf{x}) = \mathbf{H}\mathbf{x}$

Definition: Projective transformation

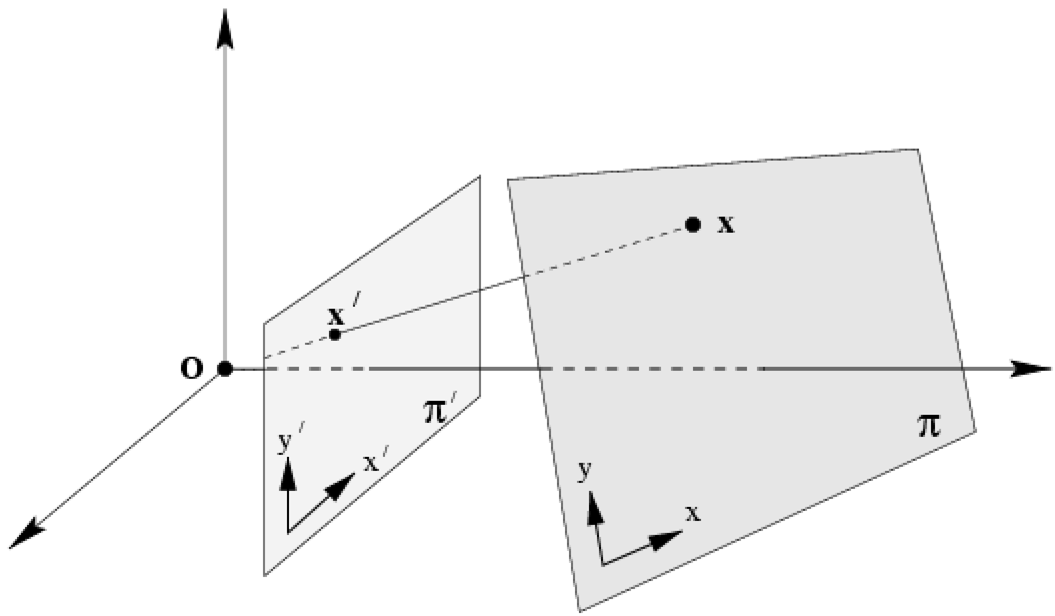
$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

or $\mathbf{x}' = \mathbf{H}\,\mathbf{x}$

8DOF

projectivity=collineation=projective transformation=homography

Mapping between planes



central projection may be expressed by x'=Hx

(application of theorem)

Removing projective distortion


select four points in a plane with know coordinates

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

$$x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13}$$
$$y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23}$$

(linear in h_{ij})

(2 constraints/point, 8DOF $\Rightarrow$ 4 points needed)