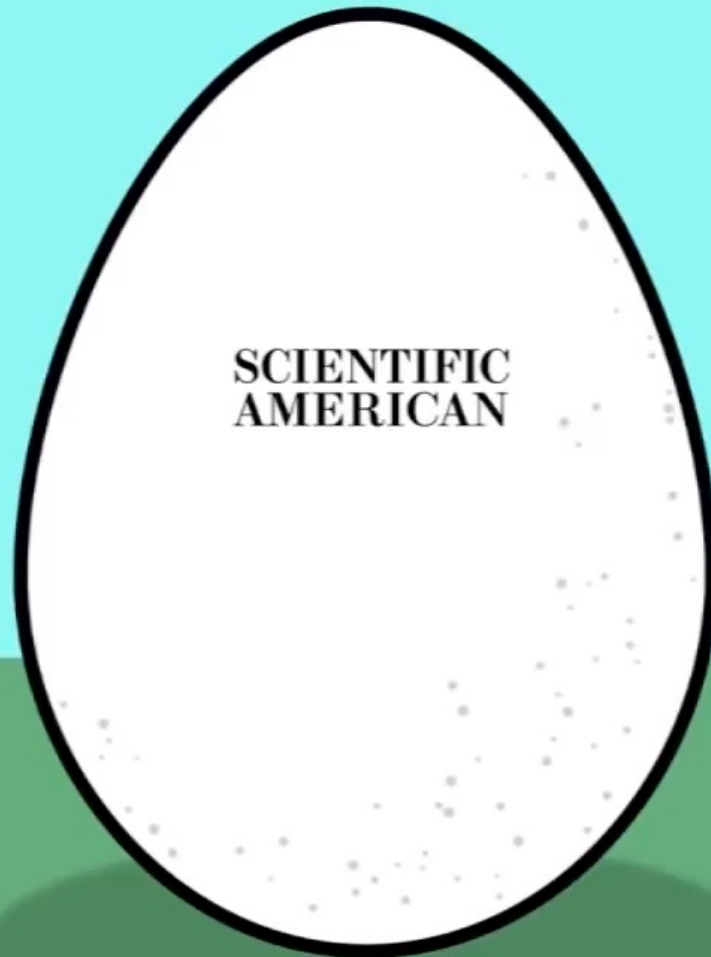
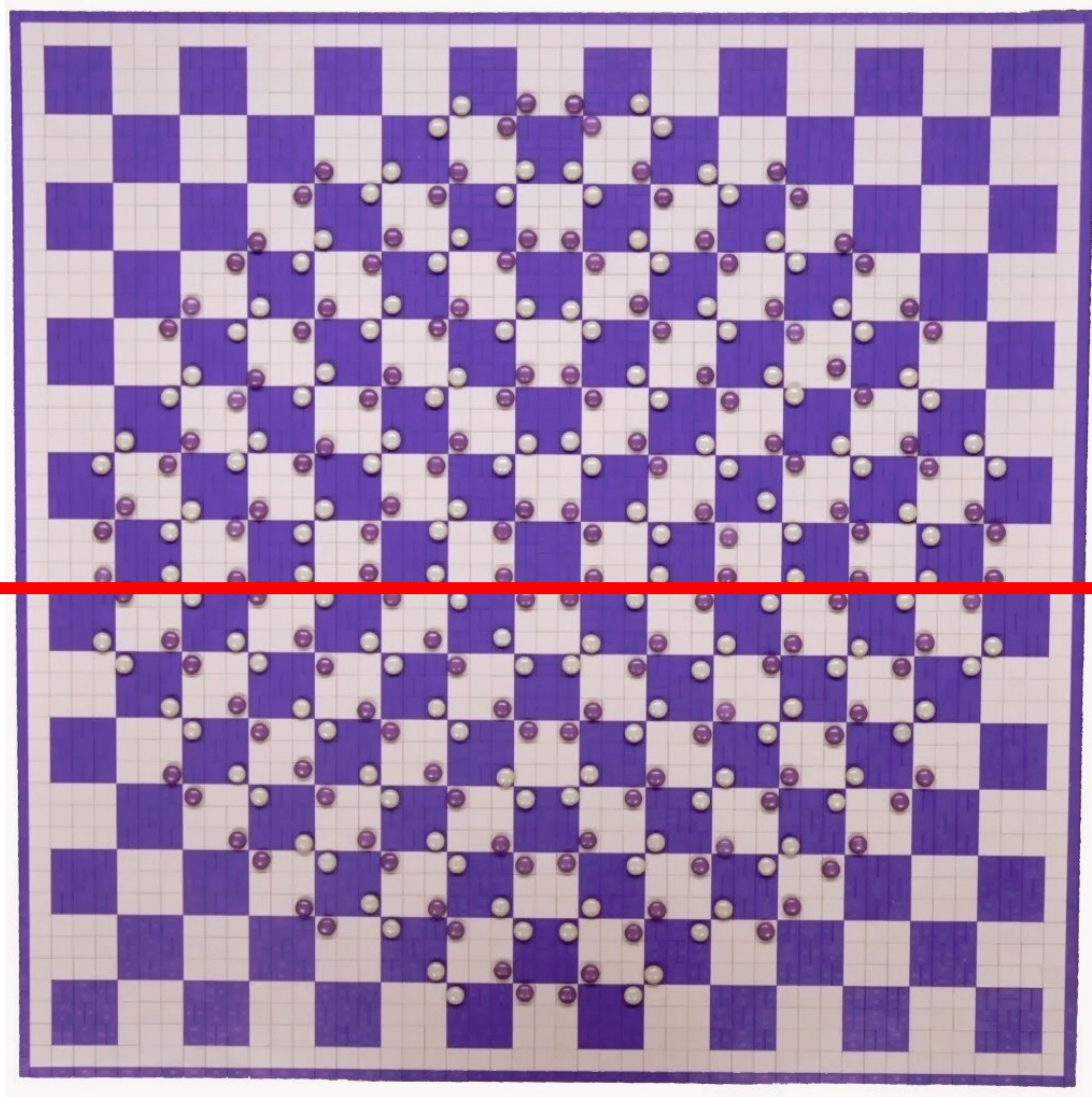


# Ames illusion





From Akiyoshi Kitaoka

# Проективные преобразования в $P^3$

$$w \begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = H \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = H \Lambda \quad \Lambda \in P^3 \quad \exists H^{-1}$$

Уравнение плоскости:  $\pi^T \Lambda = 0$


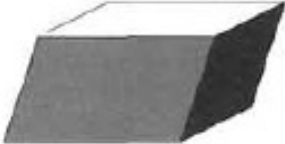


# Плоскость на бесконечности

$$\pi_{\infty} = (0,0,0,1)^T$$

$$\pi_{\infty}^T \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = 0$$

Проективное преобразование, которое переводит плоскость на бесконечности в плоскость на бесконечности, является аффинным преобразованием. Верно и обратное.

# Hierarchy of $P^3$ homography transformations

Group	Matrix	Distortion	Invariant properties
Projective 15 dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		Intersection and tangency of surfaces in contact. Sign of Gaussian curvature.
Affine 12 dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Parallelism of planes, volume ratios, centroids. The plane at infinity, $\pi_\infty$ , (see section 3.5).
Similarity 7 dof	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$		The absolute conic, $\Omega_\infty$ , (see section 3.6).
Euclidean 6 dof	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Volume.

# Канонический вид фундаментальной матрицы

- Любую пару матриц проекции камер можно привести к виду

$$P = [I \mid 0] \quad P' = [M \mid m]$$

- Фундаментальная матрица в этом случае будет иметь вид

$$F = [m]_{\times} M$$

# Проективная неоднозначность камер

- Паре матриц проекции камер  $(P, P')$  канонического вида соответствует целый класс матриц проекции  $(PH, P'H)$ , где  $H$  – любая обратимая матрица  $4 \times 4$ , с такой же фундаментальной матрицей
- Существует ли пара матриц проекции с такой же фундаментальной матрицей, которые неприводимы к этому каноническому виду?

# Проективная неоднозначность камер

- Теорема: если  $F$  – фундаментальная матрица, соответствующая парам матриц камер  $(P_1, I_1)$  и  $(P_2, I_2)$ , то существует  $H$  такое, что

$$P_2 = P_1 H, P'_2 = P'_1 H$$

- Фундаментальная матрица определяет матрицы проекции с точностью до проективного преобразования  $P^3$



# Доказательство

- Пусть существует две пары матриц проекции камер  $(P_1, P'_1)$  и  $(P_2, P'_2)$ , которые соответствуют одна фундаментальная матрица  $F$
- Без ограничения общности положим

- Тогда  $P_1 = [I | 0]$ ,  $P'_1 = [A | a]$ ,  $P_2 = [\tilde{A} | \tilde{a}]$

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

# Лемма

- Пусть матрица  $F$  ранга 2 может быть представлена двумя разными способами:

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

- Тогда существует такое число  $k$  и вектор  $v$ , что

$$\tilde{a} = ka, \tilde{A} = k^{-1} (A + av^T)$$

$$P_1' = [A \mid a], P_2' = [k^{-1}(A + av^T) \mid ka]$$

- Рассмотрим проективное преобразование

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

- Легко убедиться, что

$$P_1 H = k^{-1} P_2, P_1' H = P_2'$$

# Теорема о проективной реконструкции

- Пусть  $(q_i, q'_i)$  – пары соответствующих точек на двух изображениях, и существует единственная  $F$  такая, что  $q'^T_i F q_i = 0$  для всех  $i$ . Пусть  $(P_1, P'_1, \{Q_{1i}\})$  и  $(P_2, P'_2, \{Q_{2i}\})$  – две реконструкции, соответствующие  $(q_i, q'_i)$ . Тогда существует несингулярная матрица  $H$  такая, что  $P_2 = P_1 H^{-1}$ ,  $P'_2 = P'_1 H^{-1}$  и  $Q_{2i} = H Q_{1i}$ , кроме точек, для которых верно  $F q_i = q'^T_i F = 0$

# Доказательство

- Из предыдущего результата следует, что

$$\exists H \mid P_2 = P_1 H^{-1}, P'_2 = P'_1 H^{-1}$$

- Тогда:

$$P_2 H Q_{1i} = P_1 Q_{1i} = q_i = P_2 Q_{2i}, P'_2 H Q_{1i} = P'_2 Q_{2i}$$

- Это означает, что точки  $HQ_{1i}$  и  $Q_{2i}$  либо равны, либо проектируются в эиполи, чтд.

# PR2 складывает полотенце

A PR2 robot arm is shown in a dimly lit room, reaching towards a green surface. The arm is white and black, with a gripper at the end. The background is dark, and the lighting is focused on the robot and the surface it is interacting with.

Cloth Grasp Point Detection  
based on Multiple-View Geometric Cues  
with Application to Robotic Towel Folding

Jeremy Maitin-Shepard  
Marco Cusumano-Towner  
Jinna Lei  
Pieter Abbeel

Department of Electrical Engineering and Computer Science  
University of California, Berkeley

International Conference on Robotics and Automation, 2010

# Трехмерная реконструкция

Проективная реконструкция

(находим фундаментальную матрицу)



Афинная реконструкция

(находим плоскость на бесконечности)



Метрическая реконструкция

(взаимное расположение камер)

# Преобразование плоскости на бесконечности

- Теорема: проективное преобразование  $H:P^3 \rightarrow P^3$  отображает плоскость на бесконечности в плоскость на бесконечности тогда и только тогда, когда  $H$  – аффинное.

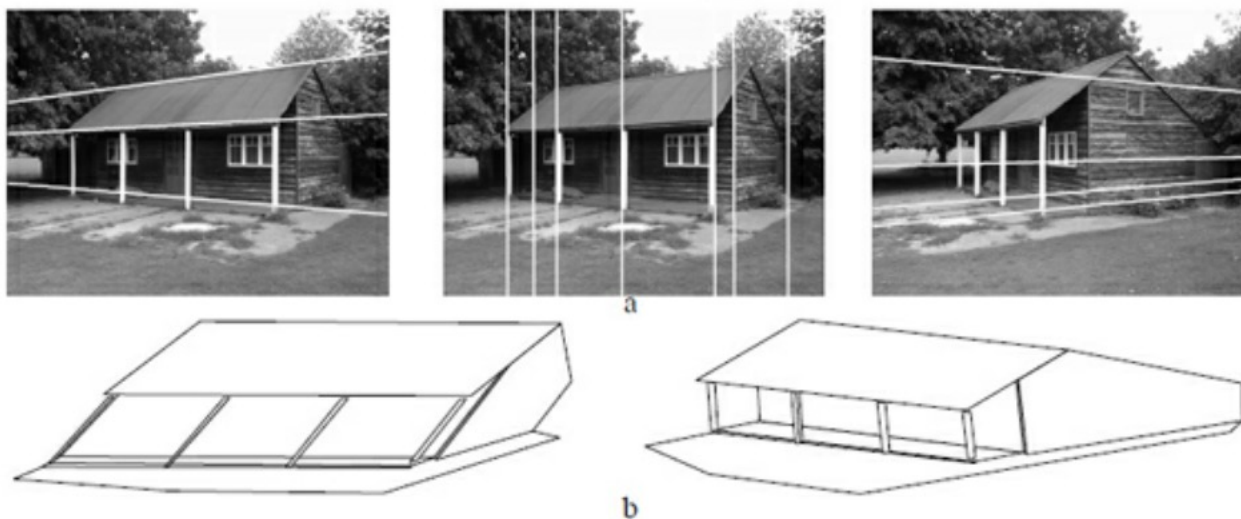


# Афинная реконструкция

- Пример преобразования, преобразующего плоскость  $\pi$  в плоскость на бесконечности (если последняя компонента вектора  $\pi$  не равна 0):

$$H = \left[ \begin{array}{c|c} E & 0 \\ \hline \pi^T & \end{array} \right]$$

# Нахождение плоскости на бесконечности



Нахождение матриц камеры при известных внутренних параметрах

$F = [T]_{\times} R$ : для единичной матрицы внутренних параметров фундаментальная матрица называется необходимой (essential) матрицей

Лемма: матрица  $3 \times 3$  является необходимой матрицей тогда и только тогда, когда два ее сингулярных числа равны друг другу, а третье равно 0.

# Восстановление позы по необходимой матрице

- Обозначим  $D = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- Пусть сингулярное разложение необходимой матрицы  $E = [T]_{\times} R$  (с точностью до константы) равно  
 $E \propto U \text{diag}(1, 1, 0) V^T$ , где  $\det(U) > 0$  и  $\det(V) > 0$ .
- Тогда  $T \propto u_3$ , а  $R$  принимает одно из двух значений:  
 $R_1 = U D V^T$  и  $R_2 = U D^T V^T$

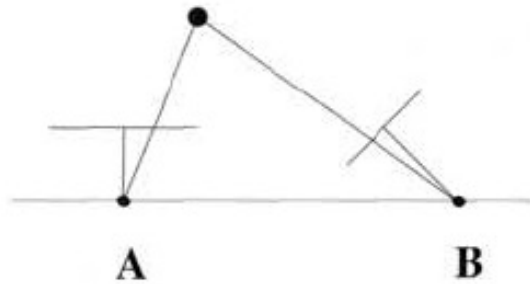
# Неоднозначность в восстановлении позы

- Существует 4 решения, которые соответствуют необходимой матрице:

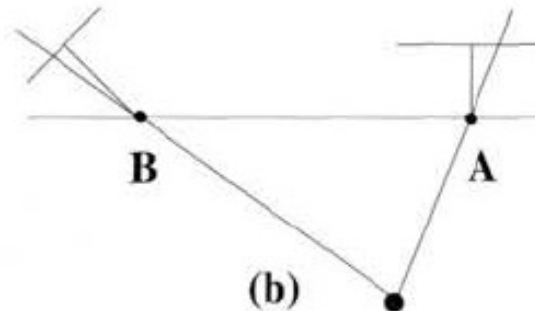
$$[R_1|T], [R_1| - T], [R_2|T], [R_2| - T]$$

- Но только одно из них соответствует положениям трехмерных точек перед камерами!

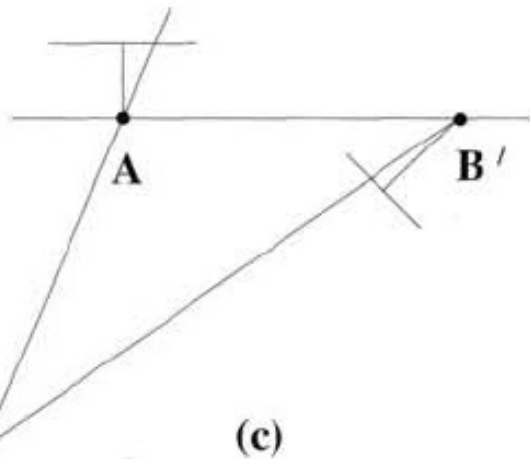
# 4 решения для матрицы проекции второй камеры



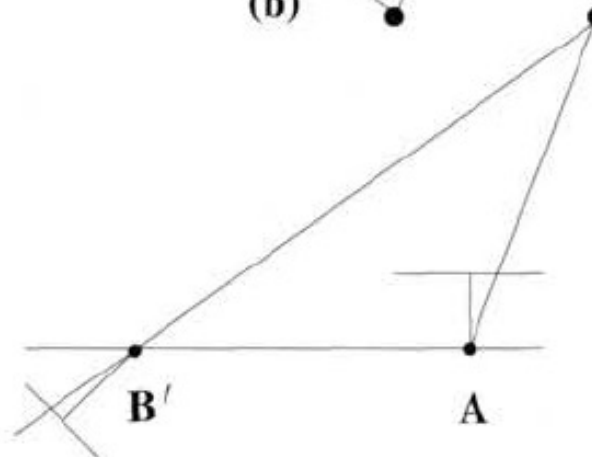
(a)



(b)



(c)



(d)

# Нахождение матриц камеры при известных внутренних параметрах

- Первое решение: Higgins, H. A computer algorithm for reconstructing a scene from two projections. *Nature***293**, 133–135 (1981).
- Устойчивый алгоритм D.Nister, An Efficient Solution to the Five-Point Relative Pose Problem, IEEE Trans. PAMI, June 2004

# Нахождение позы по гомографии: Алгоритм Faugeras-Lustman

$$Q_2 = RQ_1 + t \quad nQ_1 = d$$



$$q_2 = Hq_1 \quad H = dR + tn^T$$

- В общем случае это уравнение имеет 8 различных решений. Решений 4 ттт, когда  $H$  имеет сингулярное значение кратности 2. Решений несчётное множество ттт, когда  $H$  имеет сингулярное значение кратности 3.



# Алгоритм Faugeras-Lustman

- Используя условие, что все точки должны быть в области видимости, из 8 решений остается только два. При этом, если не все точки находятся ближе к одной камере, чем к другой, то решение будет единственно\*.

\* Hugh Christopher Longuet-Higgins, The reconstruction of a plane surface from two perspective projections, *Proc. R. Soc. Lond. B*,  
<http://doi.org/10.1098/rspb.1986.0030>

# Другие методы декомпозиции гомографии

- Also based on SVD
  - Zhang, Zhongfei & Hanson, Allen. (1996). D Reconstruction Based on Homography Mapping. ARPA Image Understanding Workshop.
- Closed form solution (implemented in OpenCV)
  - Ezio Malis, Manuel Vargas, and others. Deeper understanding of the homography decomposition for vision-based control. Inria report, 2007.

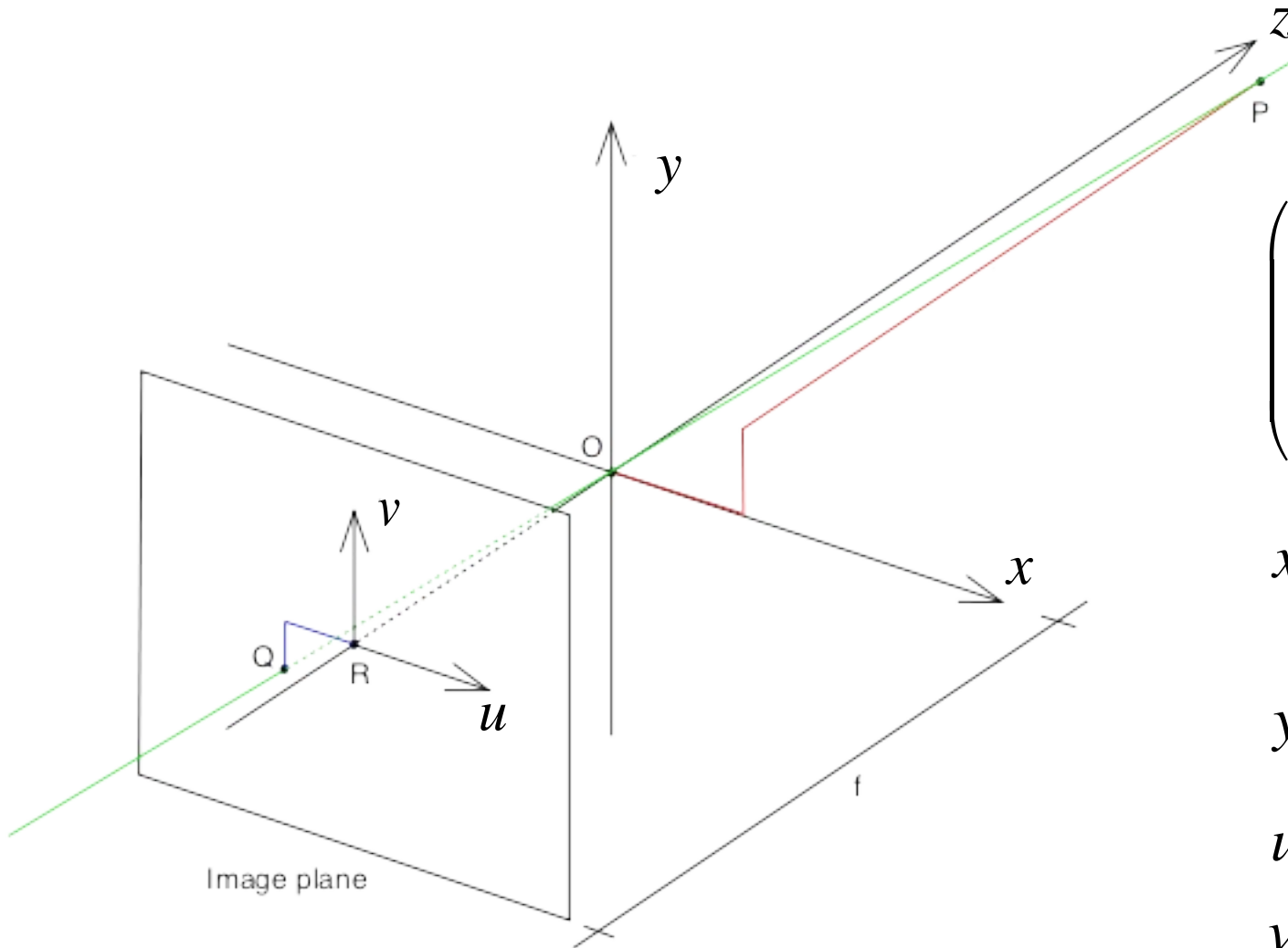
# Полезные функции OpenCV

- **findFundamentaMat**: нахождение фундаментальной матрицы по координатам пикселей
- **findEssentialMat**: нахождение необходимой матрицы по координатам пикселей
- **decomposeEssentialMat**: нахождение позы по необходимой матрице (возвращает 4 решения)
- **decomposeHomographyMat**: нахождение позы по матрице гомографии

# SLAM applications for robotics



# Pinhole camera model



$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} + t$$

$$x' = \frac{X}{Z}$$

$$y' = \frac{Y}{Z}$$

$$u = f_x x' + c_x$$

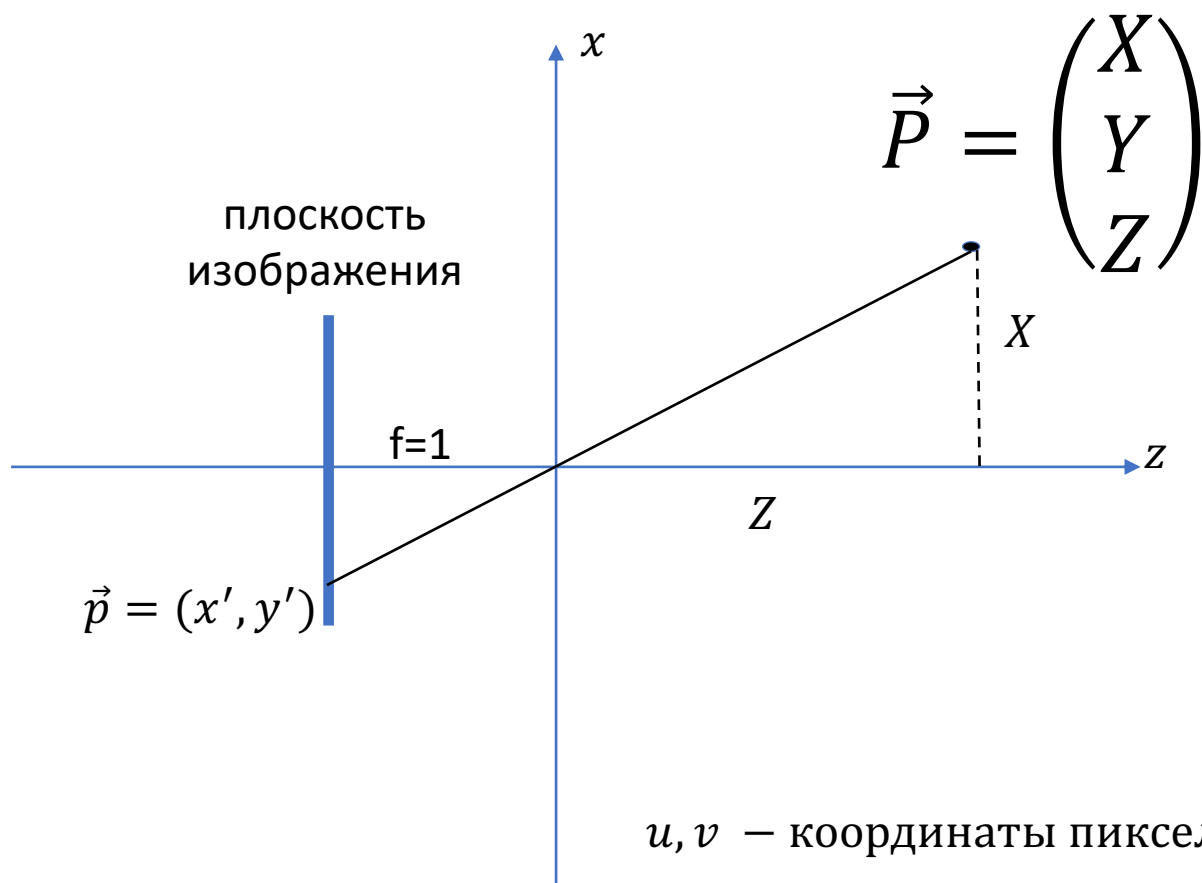
$$v = f_y y' + c_y$$

# Large-Scale SLAM: Dubrovnik



Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz and Richard Szeliski, *Building Rome in a Day*. ICCV 2009

# Модель камеры-обскуры



$$x' = \frac{X}{Z}$$

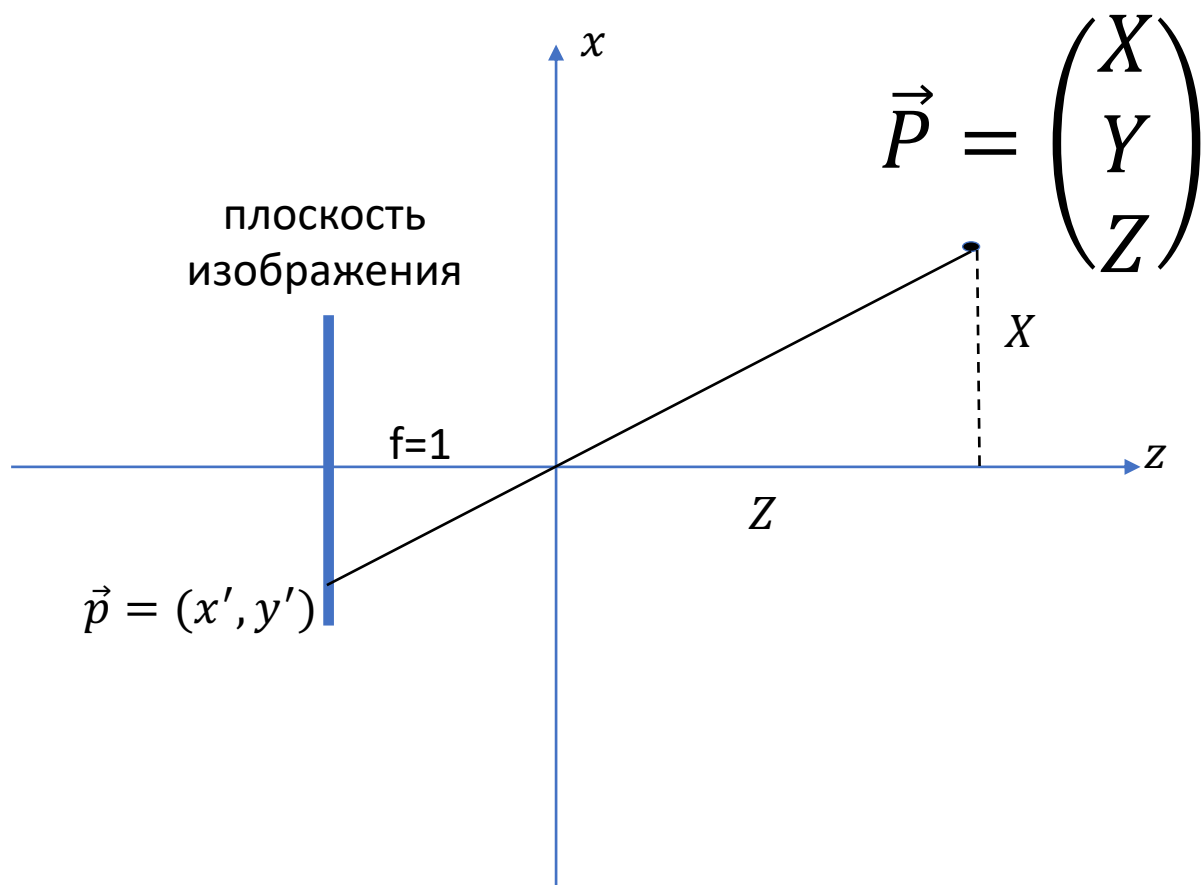
$$y' = \frac{Y}{Z}$$

$$u = f_x x' + c_x$$

$$v = f_y y' + c_y$$

$u, v$  — координаты пикселей на изображении

# Получение внутренних параметров по фокусному расстоянию и dpi



$$x' = \frac{X}{Z}f$$

$$y' = \frac{Y}{Z}f$$

$$u = \frac{x'}{\text{pixel size } x} + c_x$$

$$v = \frac{y'}{\text{pixel size } y} + c_y$$

$$\text{pixel size } x = \frac{1}{\text{dpi } x}$$

$$\text{pixel size } y = \frac{1}{\text{dpi } y}$$

$u, v$  — координаты пикселей на изображении



# Получение внутренних параметров по фокусному расстоянию 35мм

Сенсор 35мм имеет размер 36x24mm, то есть нам достаточно знать разрешение изображения:

$$x' = \frac{X}{Z} f_{35}$$

$$y' = \frac{Y}{Z} f_{35}$$

$$u = \frac{x'}{\text{pixel size } x} + c_x$$

$$v = \frac{y'}{\text{pixel size } y} + c_y$$

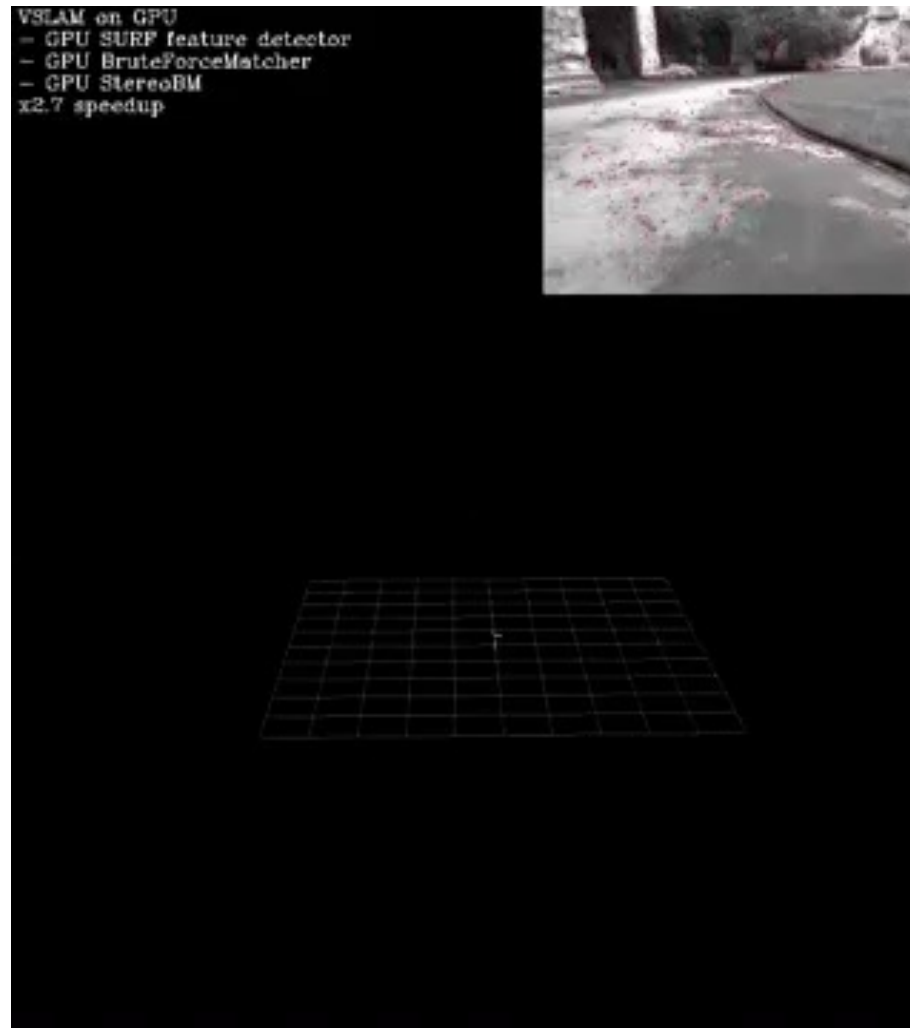
$$\text{pixel size } x = \frac{36mm}{width}$$

$$\text{pixel size } y = \frac{24mm}{height}$$

# SLAM: Simultaneous Localization and Mapping



# SLAM: Simultaneous Localization and Mapping



# Оптимизация положения камер и точек

$$\min_{P_j, Q_i} \sum_{i,j} w_{ij} d(q_{ij}, P_j Q_i)^2$$

$P_j$  -- матрица проекции  $j$ -й камеры

$Q_i \in P^3$  --  $i$ -я точка в трехмерном пространстве

$q_{ij} \in P^2$  -- ее проекция на  $j$ -е изображение

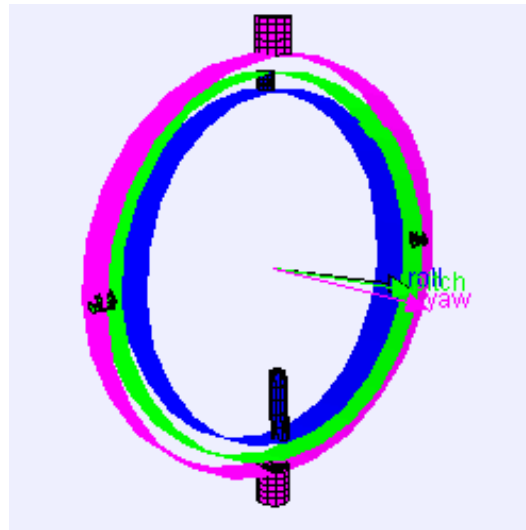
$w_{ij}$  равно 1, если  $Q_i$  присутствует на изображении  $j$ , иначе 0

# Как представлять повороты?

- Углы Эйлера
- Представление «ось-угол»
- Кватернионы

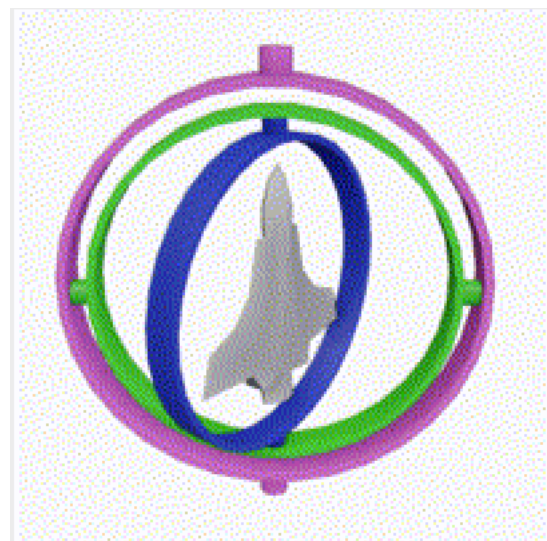
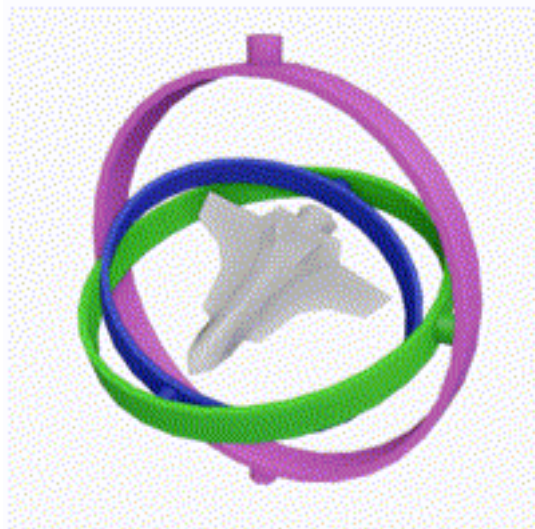
# Углы Эйлера

$$R = R_x R_y R_z =$$
$$= \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & -\sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{pmatrix}$$



# Складывание рамок (gimbal lock)

Если две рамки близки друг к другу, возникает сингулярность: требуются большие углы поворота для того, чтобы повернуть вектор на небольшой угол



Инцидент с складыванием рамок на Аполло <https://www.hq.nasa.gov/alsj/gimbals.html>

# Представление «ось-угол»

- Теорема вращения Эйлера: любое движение твердого тела в  $R^3$ , имеющее неподвижную точку, является вращением тела вокруг некоторой оси
- $\vec{\alpha} = (\alpha_x, \alpha_y, \alpha_z)$  представляет поворот на угол  $|\alpha|$  вдоль оси  $\frac{\vec{\alpha}}{|\alpha|}$
- Сложная связь с матрицами поворота, затрудняющая вычисление производных
- Проблема на границе: большое расстояние между векторами поворота не означает большой относительный поворот:  $(0,0, 179^\circ)$  и  $(0,0, -179^\circ)$  отличаются всего на два градуса



# Кватернионы

- $q = a + bi + cj + dk$
- $ii = jj = kk = -1$
- $ij = k, ji = -k$
- $jk = i, kj = -i$
- $ki = j, ik = -j$
- $q^* = a - bi - cj - dk$
- $(q_1 q_2)^* = q_2^* q_1^*$
- Произведение некоммутативно!

# Broom Bridge, Dublin, Ireland



# Поворот при помощи кватернионов

- $q = \cos \frac{\theta}{2} + u \sin \frac{\theta}{2}, u = u_x \mathbf{i} + u_y \mathbf{j} + u_z \mathbf{k}$
- $p = p_x \mathbf{i} + p_y \mathbf{j} + p_z \mathbf{k}$
- $p' = qpq^*$  -- вектор  $p$ , повернутый относительно вектора  $u$  на угол  $\theta$
- $p' = q_1 q_2 p (q_1 q_2)^* = q_1 q_2 p q_2^* q_1^*$  -- соответствует повороту  $q_2$ , за которым следует поворот  $q_1$ .

# Преимущества представления поворотов кватернионами

- Нет сингулярности, соответствующей эффекту складывания рамок
- Более компактное представление, чем матрицей поворота
- Простая связь с вектором поворота, легко считать производные
- Легче получить угол поворота, чем для матрицы, легче нормализовать (для матрицы надо использовать SVD)