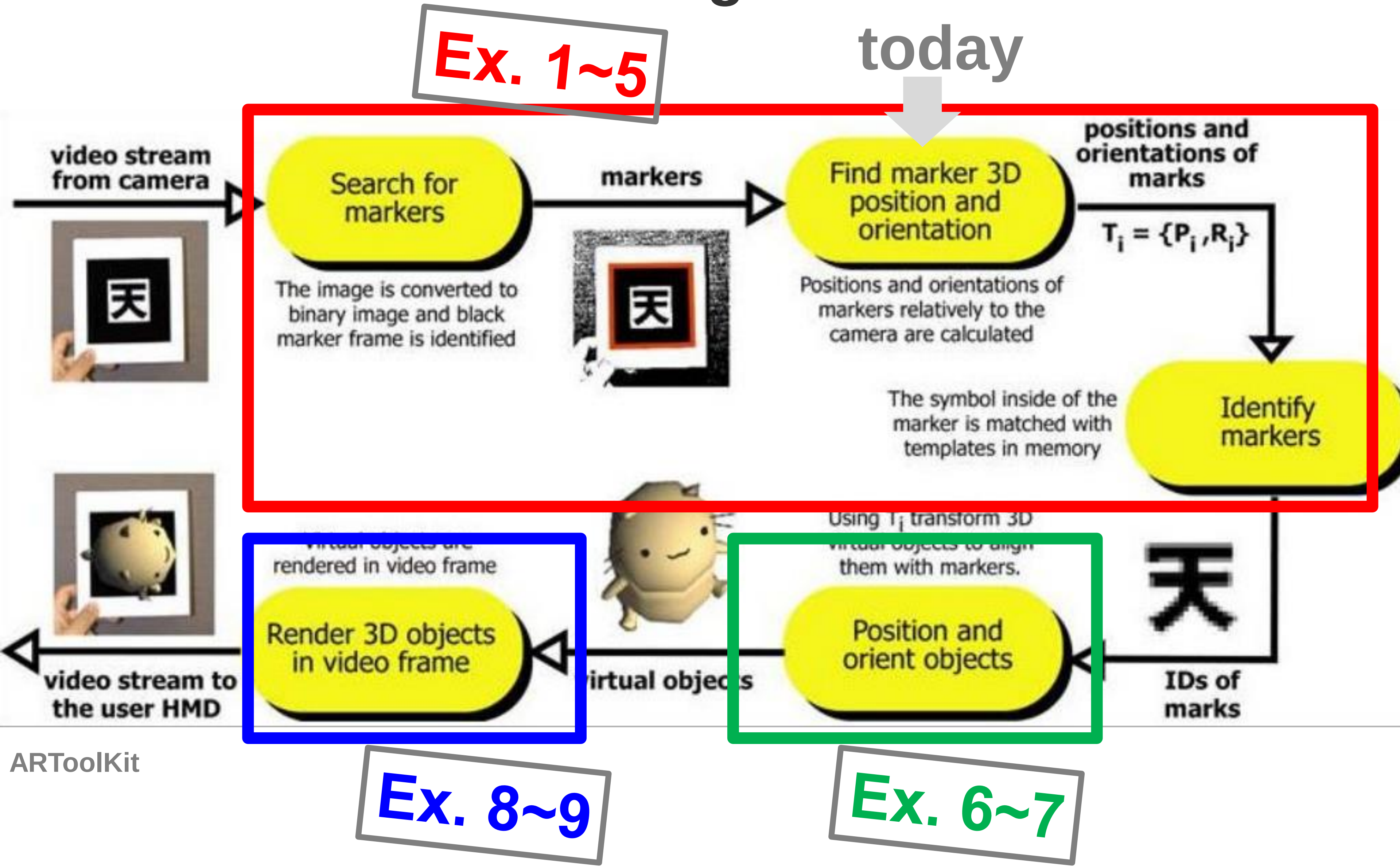
Introduction to Augmented Reality

Tutorial 5: Marker Tracking Part 5
May 16th 2018

Andreas Langbein, Adnane Jadid, David Plecher

Fachgebiet Augmented Reality
Technische Universität München

Marker-based Tracking



Ex. 1~5

today

Ex. 8~9

Ex. 6~7

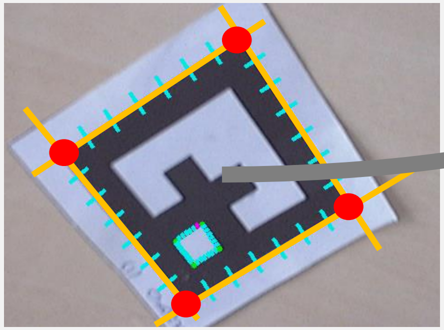
video stream from camera
Search for markers
The image is converted to binary image and black marker frame is identified
markers
Find marker 3D position and orientation
Positions and orientations of markers relatively to the camera are calculated
positions and orientations of marks
$T_i = \{P_i, R_i\}$
Identify markers
The symbol inside of the marker is matched with templates in memory
天
IDs of marks
Position and orient objects
Using T_i transform 3D virtual objects to align them with markers.
virtual objects
Render 3D objects in video frame
Virtual objects are rendered in video frame
video stream to the user HMD

ARToolKit

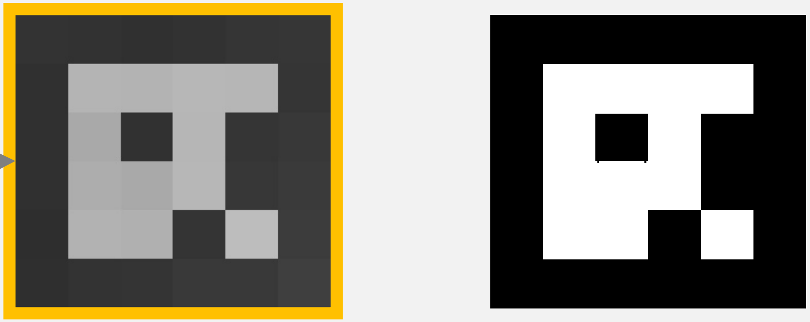
Solution for the Previous Tutorial

Ex. 4

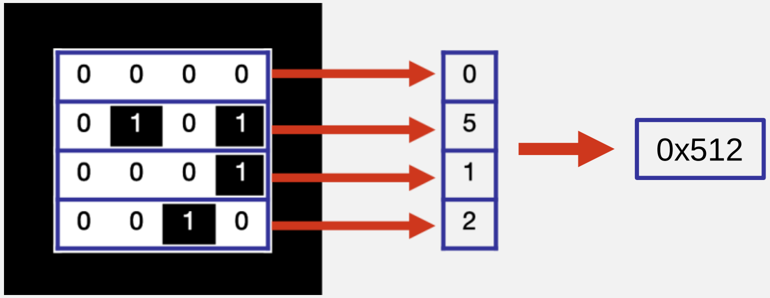
Find marker corners **precisely**



RECTIFY marker



Identify marker ID



0	0	0	0	0
0	1	0	1	5
0	0	0	1	1
0	0	1	0	2

0x512

Code walkthrough

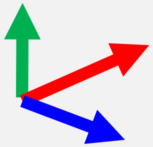
Today's Tutorial

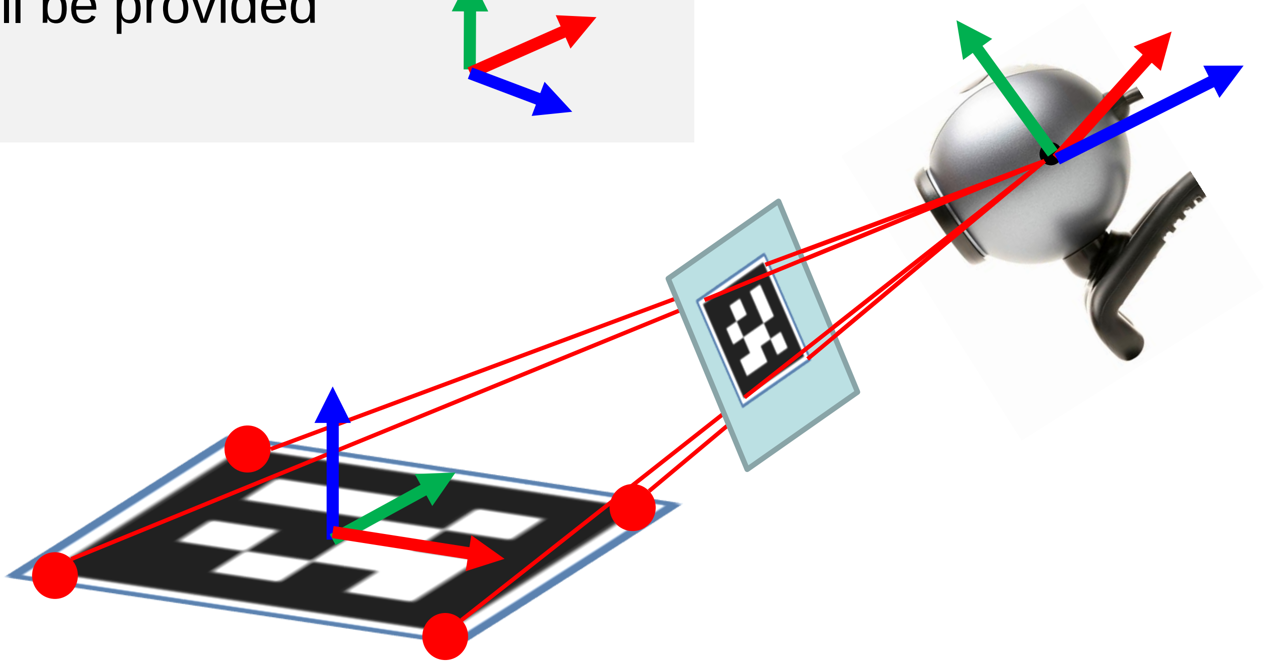
Ex. 5

Marker-Pose Estimation

Pose = Position + Orientation

A code will be provided





3D Transformations Revisited

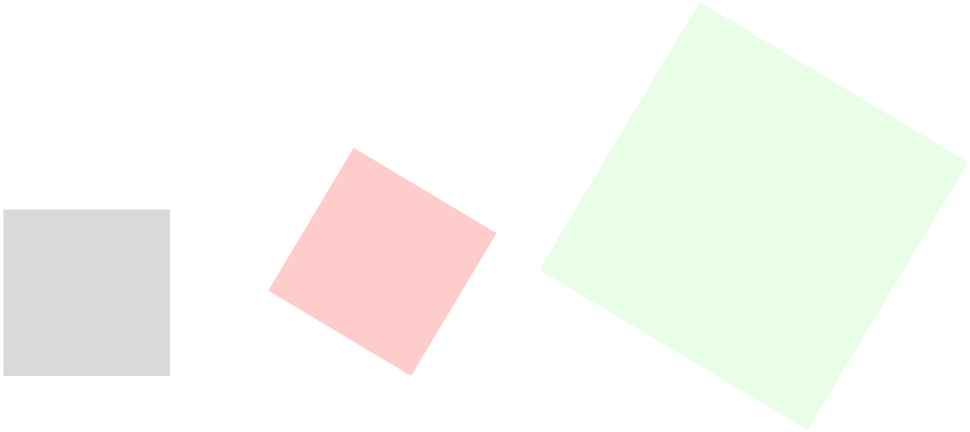
Homogeneous notation in R^3 → 4x4 matrix

Similarity

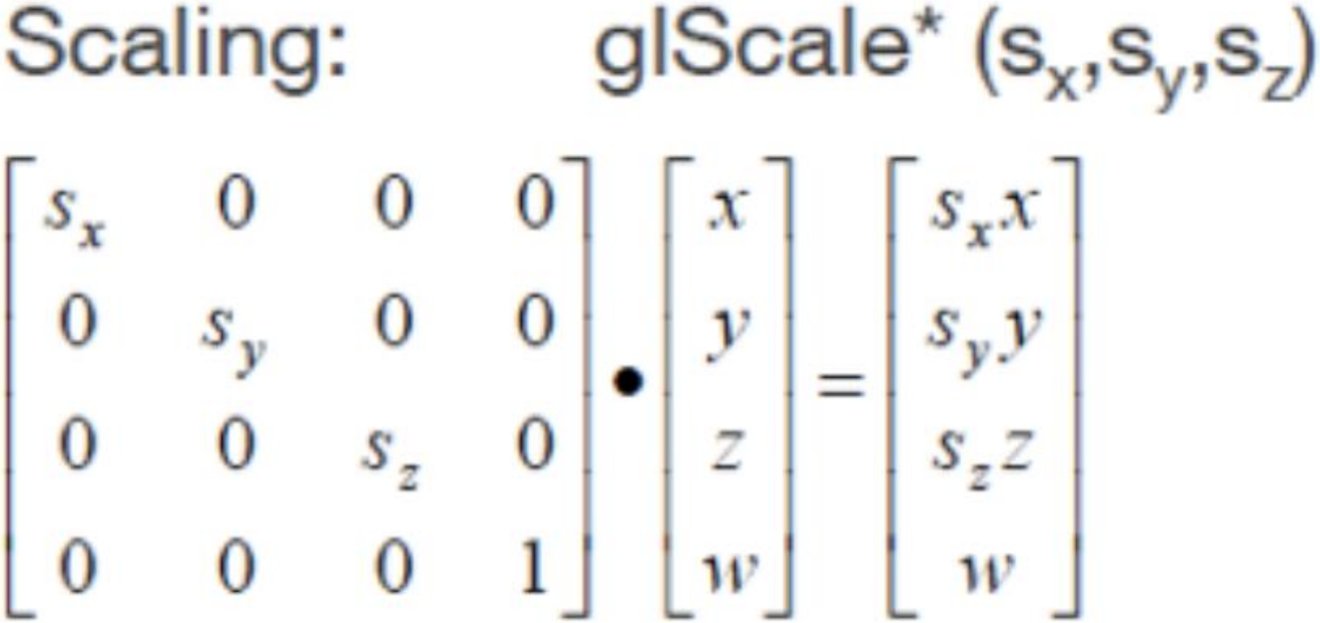
Euclidean

Translation, Rotation, Scaling

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} \approx \begin{bmatrix} R_{00} & R_{01} & R_{02} & s_0 t_0 \\ R_{10} & R_{11} & R_{12} & s_1 t_1 \\ R_{20} & R_{21} & R_{22} & s_2 t_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



Scaling



Scaling: glScale* (s_x,s_y,s_z)

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ s_z z \\ w \end{bmatrix}$$

Translation

Translation: glTranslate* (t_x, t_y, t_z)

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x + wt_x \\ y + wt_y \\ z + wt_z \\ w \end{bmatrix}$$

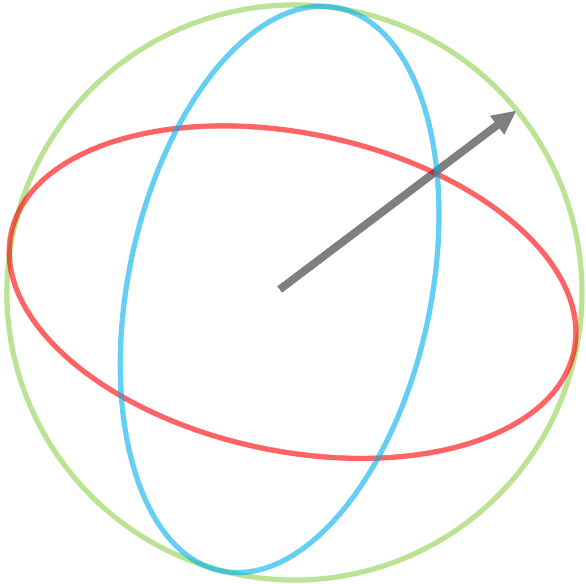
Rotation

$$R=\begin{bmatrix} R_{00} & R_{01} & R_{02} \\ R_{10} & R_{11} & R_{12} \\ R_{20} & R_{21} & R_{22} \end{bmatrix}$$

→ $I=RR^T$

(Unitary matrix)

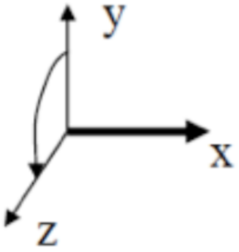
How to create a desired rotation?



e.g. Euler Angles (Around-axis Rotations)

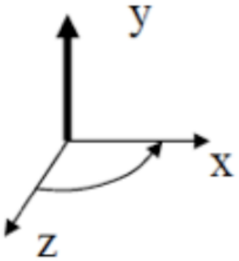
Rotation: glRotate* (a,ex,ey,ez)

- Around x



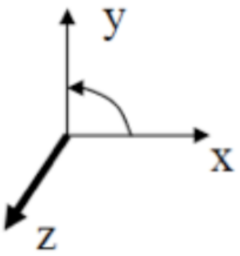
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x \\ \cos\alpha\, y - \sin\alpha\, z \\ \sin\alpha\, y + \cos\alpha\, z \\ w \end{bmatrix}$$

- Around y



$$\begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} \cos\alpha\, x + \sin\alpha\, z \\ y \\ -\sin\alpha\, x + \cos\alpha\, z \\ w \end{bmatrix}$$

- Around z



$$\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} \cos\alpha\, x - \sin\alpha\, y \\ \sin\alpha\, x + \cos\alpha\, y \\ z \\ w \end{bmatrix}$$

We do use this for exercises with OpenGL, but…

Why Euler Angles are Evil

See video now!

http://www.youtube.com/watch?v=zc8b2Jo7mno

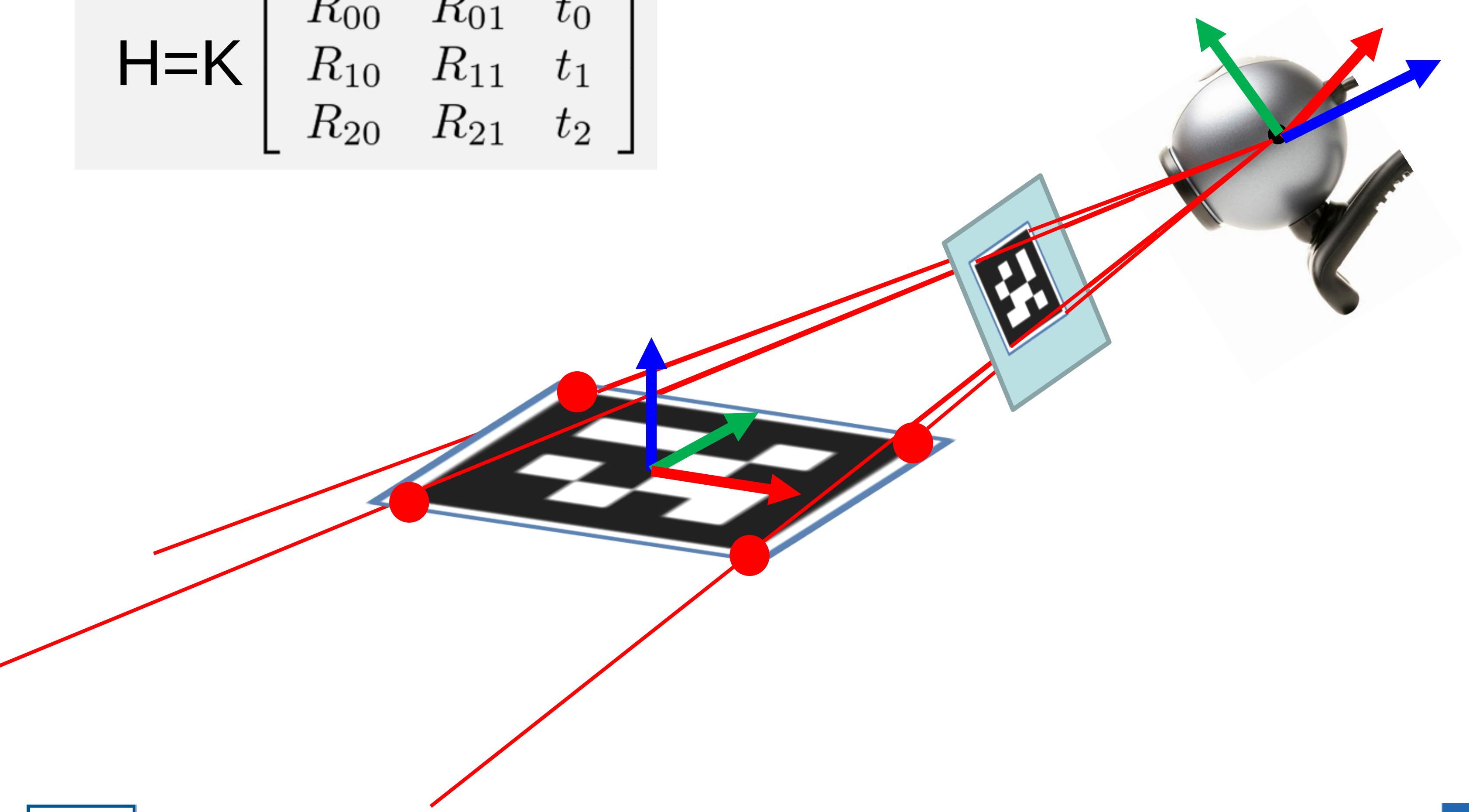
Pose Estimation via Homography

If H and K are known

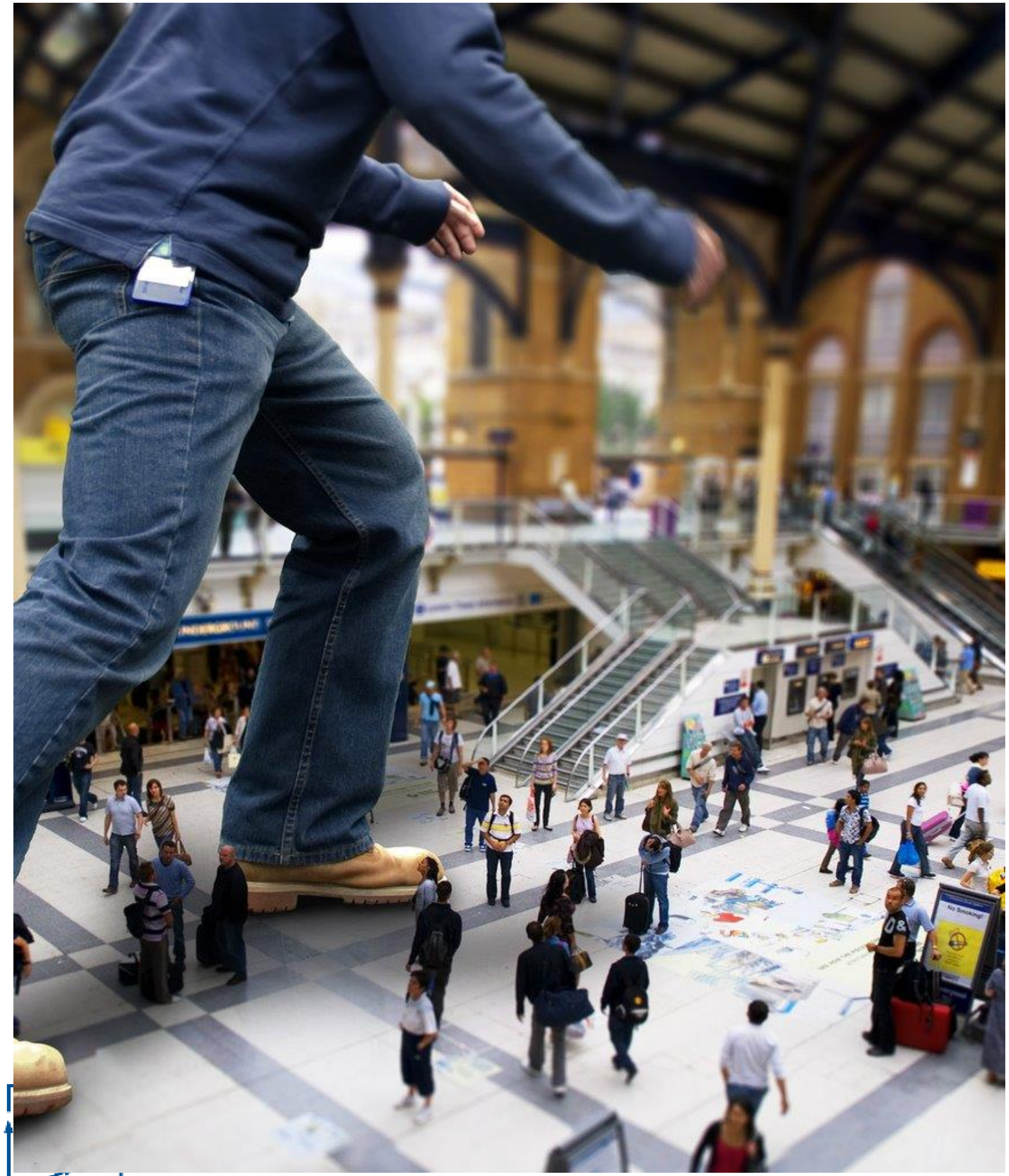
→(R, t)

(Investigate Tsai's and Zhang's methods for details)

$$H = K \begin{bmatrix} R_{00} & R_{01} & t_0 \\ R_{10} & R_{11} & t_1 \\ R_{20} & R_{21} & t_2 \end{bmatrix}$$



2D Image Does not Give 3D Scale



by Jamie Durrant

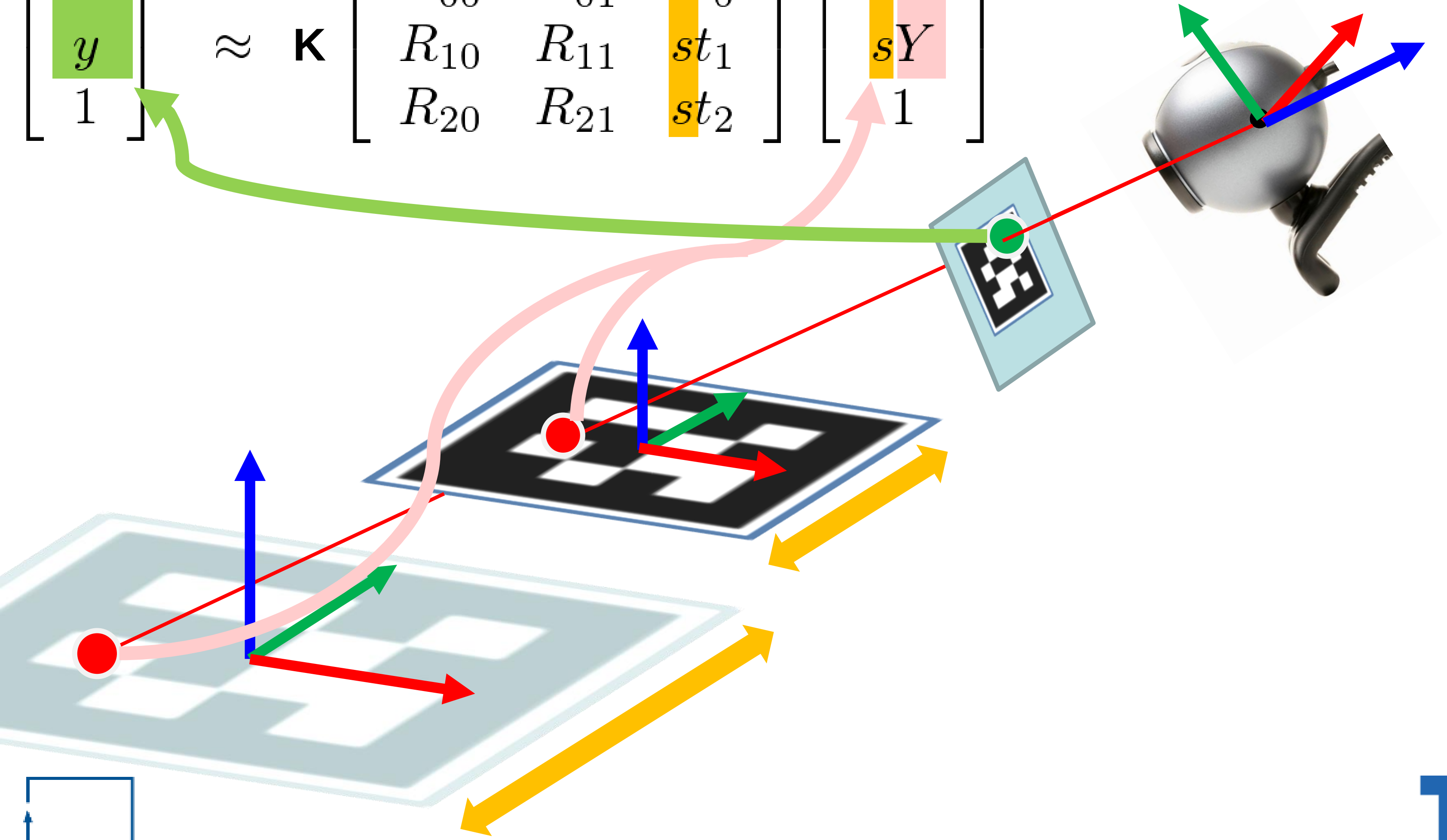
(without prior knowledge)

Scale Disambiguation

If **the marker size** is known

→ S (**s*t** gives physically correct distance)

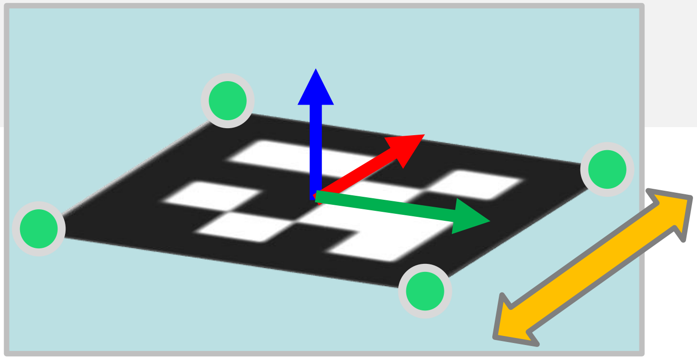
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \approx \mathbf{K} \begin{bmatrix} R_{00} & R_{01} & st_0 \\ R_{10} & R_{11} & st_1 \\ R_{20} & R_{21} & st_2 \end{bmatrix} \begin{bmatrix} sX \\ sY \\ 1 \end{bmatrix}$$



Our Pose Estimation Function

```
/**
* computes the orientation and translation of a square
* @param result result as 4x4 matrix
* @param p2D coordinates of the four corners in clock-wise order.
* the origin is assumed to be at the camera's center of projection
* @param markerSize side-length of marker. Origin is at marker center.
*/
void estimateSquarePose
                 ( float* result, const CvPoint2D32f* p2D,float markerSize );

void estimateSquarePose
                 ( float* result, const cv::Point2f* p2D, float markerSize );
```

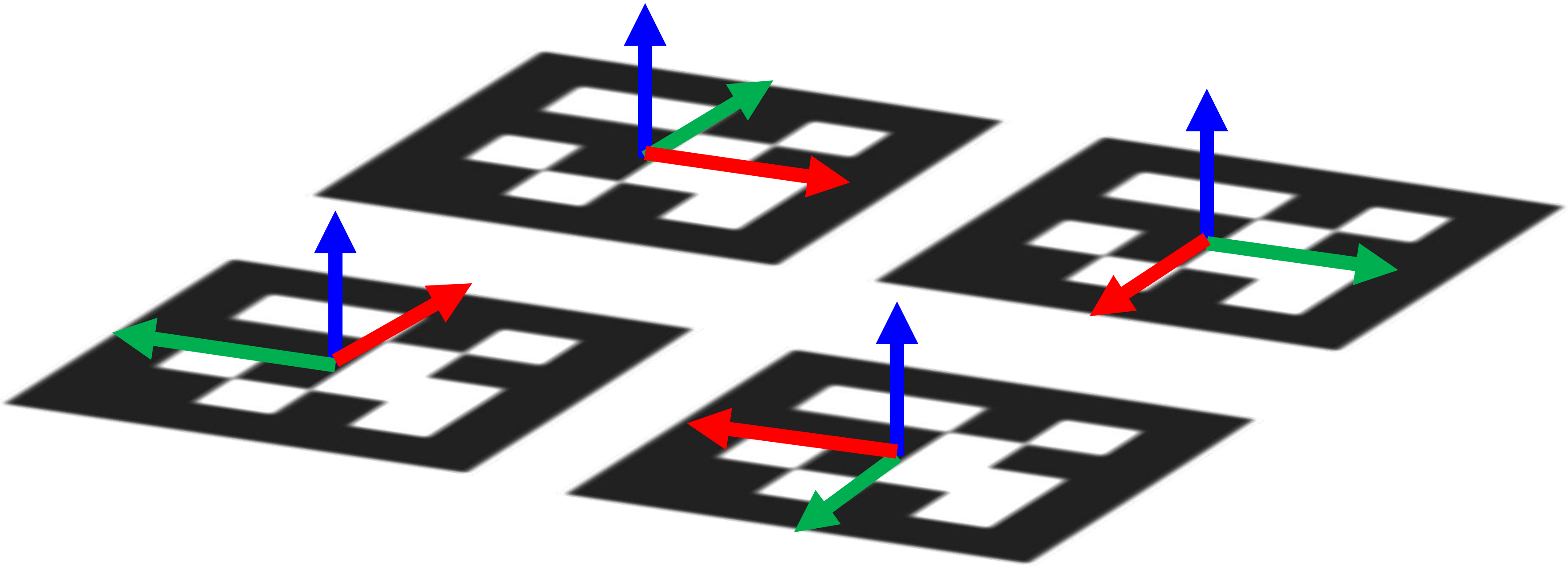


Pose Estimation

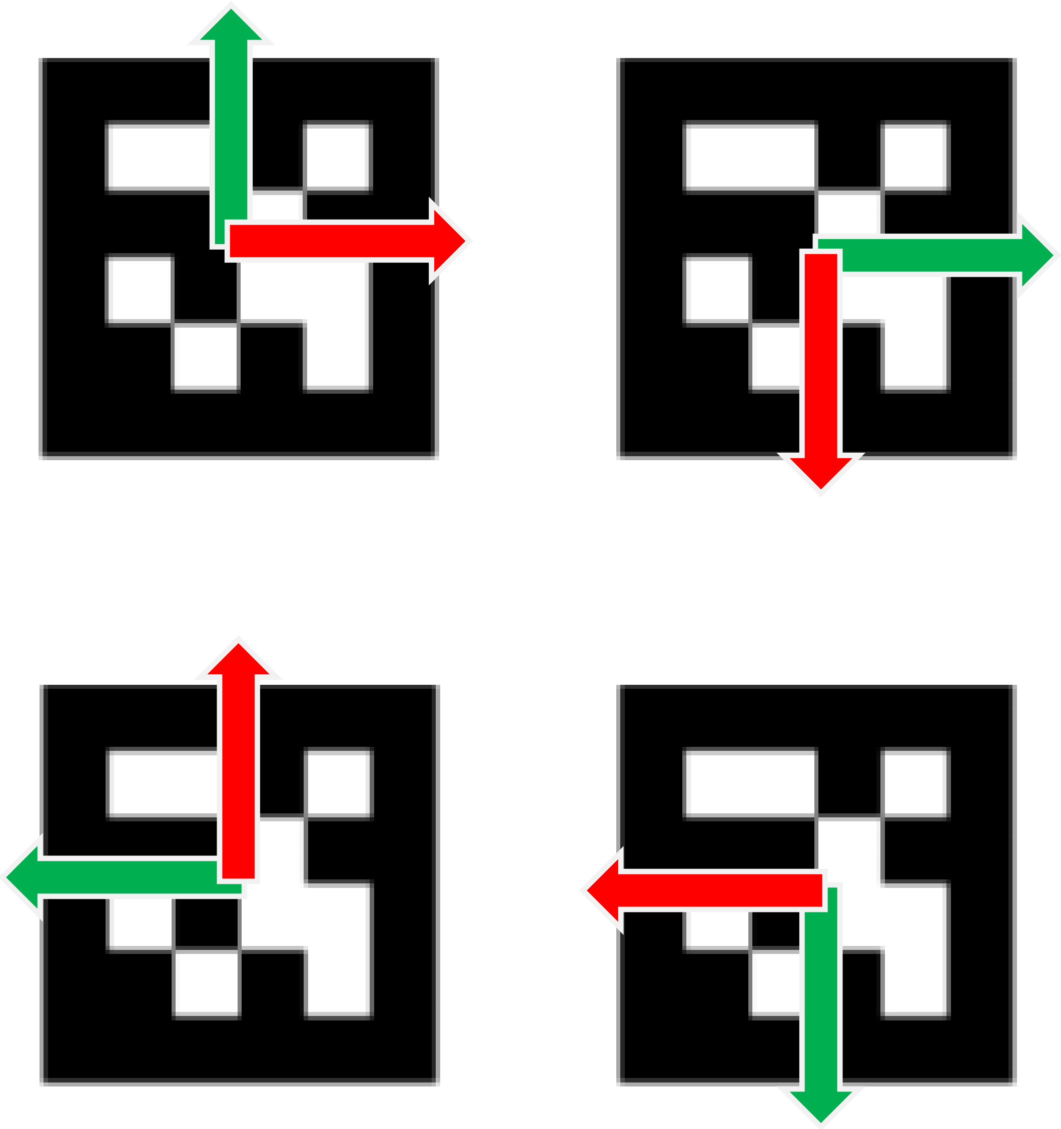
```
float* result = [   0,  1,  2,  3,
                    4,  5,  6,  7,
                    8,  9, 10, 11,
                   12, 13, 14, 15];
```

$$\begin{bmatrix} R_{00} & R_{01} & R_{02} & t_0 \\ R_{10} & R_{11} & R_{12} & t_1 \\ R_{20} & R_{21} & R_{22} & t_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

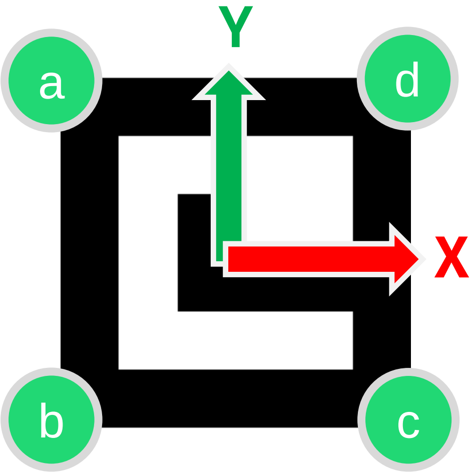
Ambiguity of the rotation around Z-axis



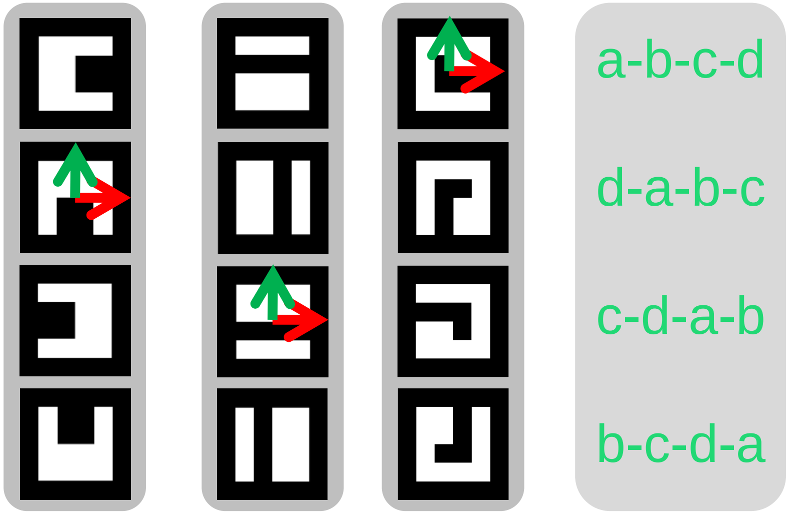
Ambiguity of the rotation around Z-axis



Preparation for Pose Estimation



Define a consistent rotation around **Z-axis**



Adujst the order of

```
const cv::Point2f* p2D
```

```
estimateSquarePose()
```

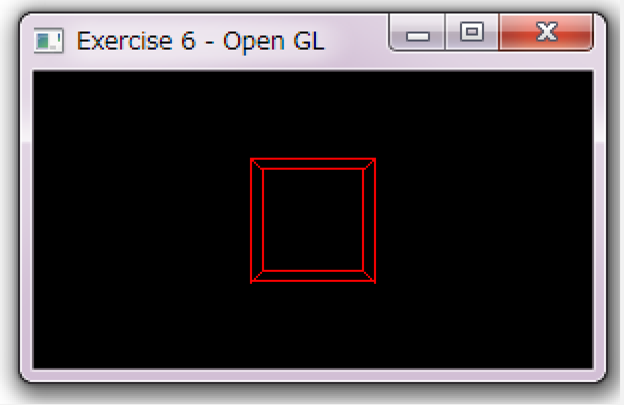
Homework

- Pose estimation: see PDF sheet on Moodle

Spoiler of the next tutorial

OpenGL basics

-with GLFW



OpenGL®

That‘s it…

- Questions