3D Scanning & Motion Capture

Exercise - 1

Armen Avetisyan, Dejan Azinović

Team

Lecturers

Dr. Justus Thies

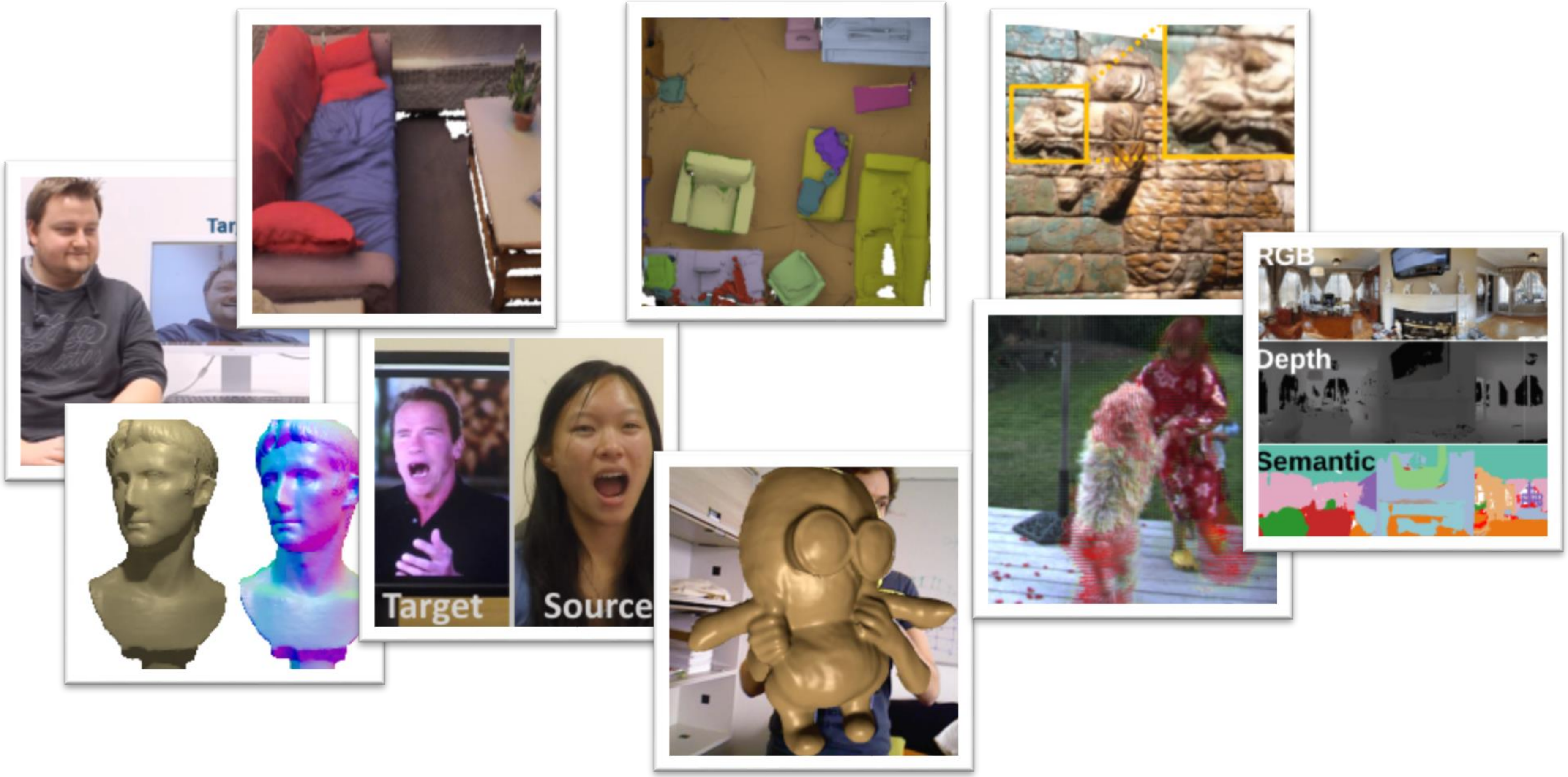
Angela Dai

TAs

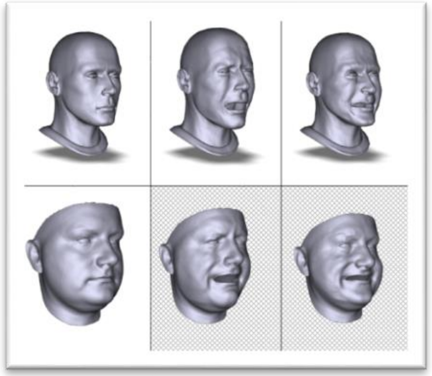
Armen Avetisyan

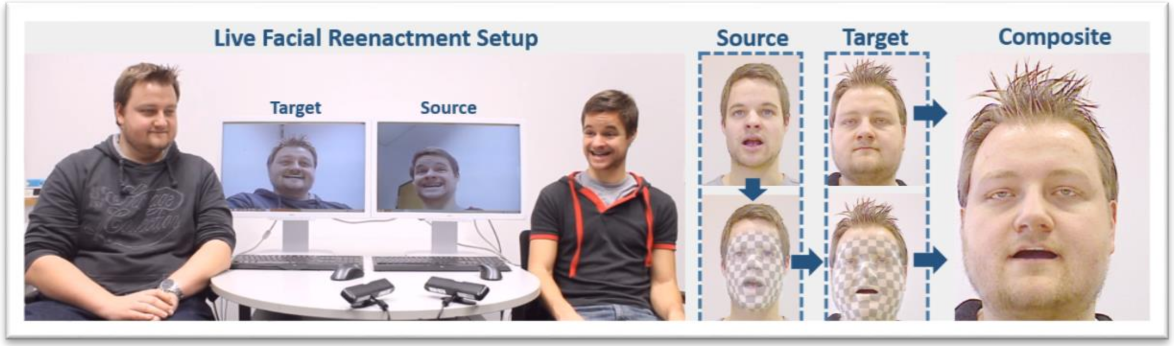
Dejan Azinović

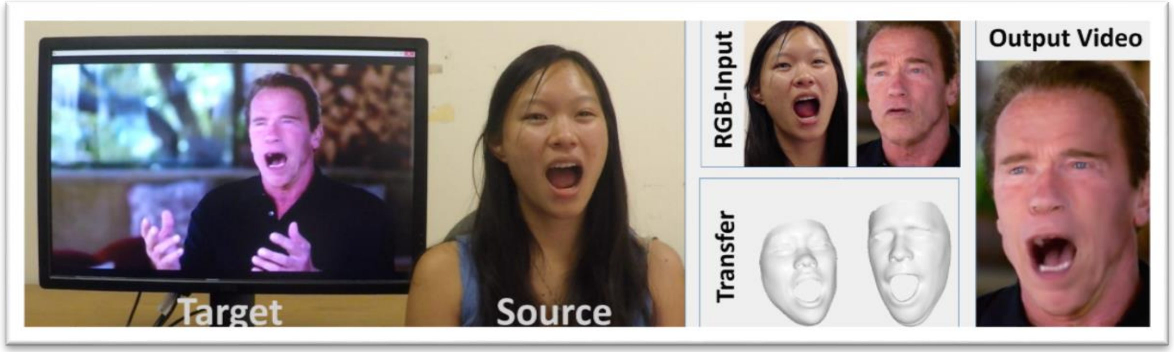
Research Projects

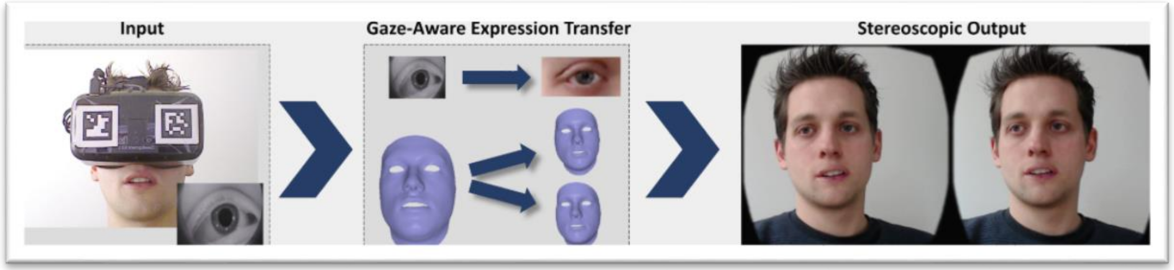


Research Projects









Lecture+Tutorials

- Requirements
  - C++ is a must
  - Profound knowledge of linear algebra
  - Basic concepts of 3D graphics

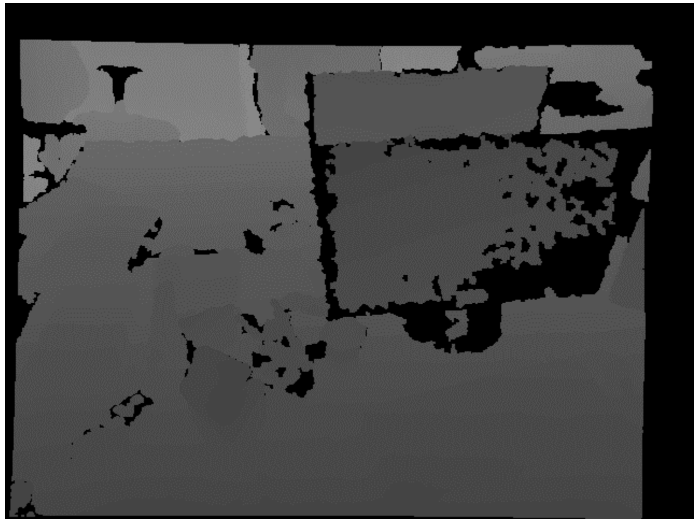
Tutorials

- Basic 3D reconstruction algorithms
  1. Exercise (Camera Intrinsics, Back-projection, Meshes)
  2. Exercise (Surface Representations, Volumetric Fusion, SDF)
  3. Exercise (Object Alignment, ICP)
  4. Exercise (Feature Matching)
- 1 week of working time
- **Groups of two** are allowed
- Need to pass at least three exercise submissions, with the fourth being at least borderline accepted for 0.3 bonus

Project

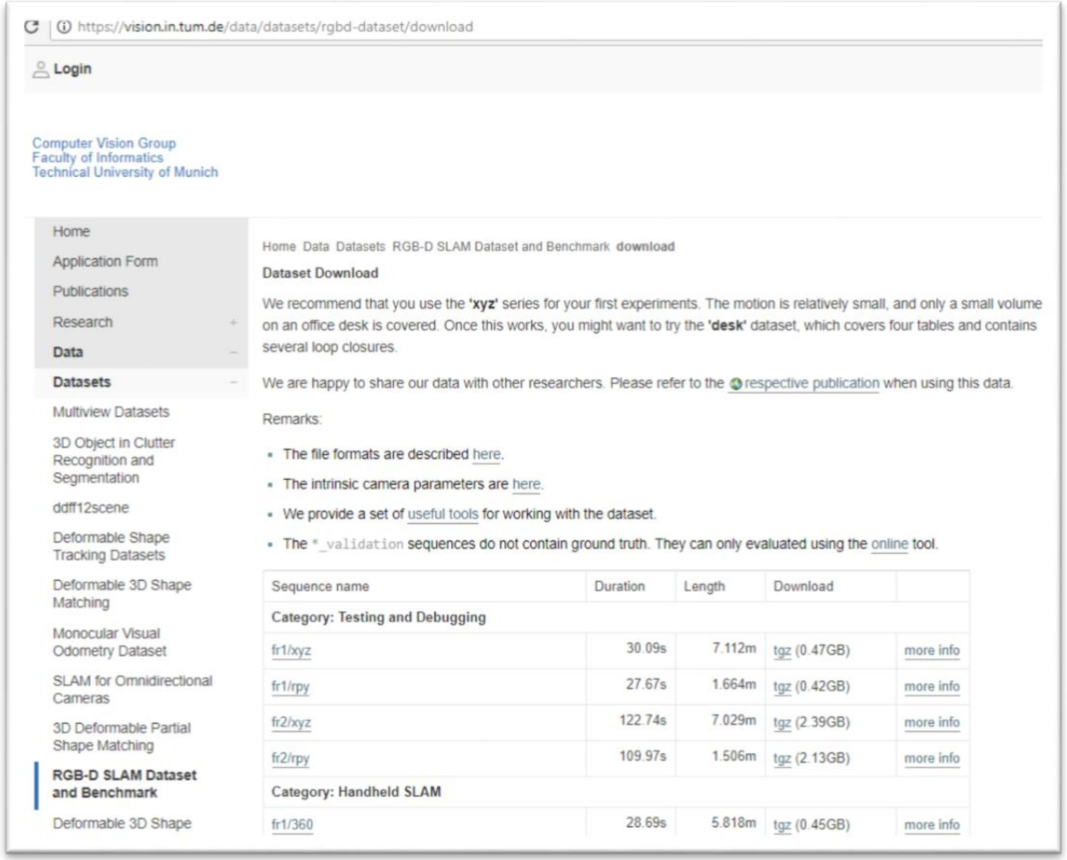
- 3D reconstruction / tracking project
  - KinectFusion, Face Fitting, Bundling etc. …
- 6 weeks
- Groups of 4
- Proposal (abstract 1-2 pages)
- Presentation of the project (poster) + abstract (2 pages with results)
- 40% of the exam

Kinect



Kinect – RGB-D Dataset

- https://vision.in.tum.de/data/datasets/rgbd-dataset



https://vision.in.tum.de/data/datasets/rgbd-dataset/download

Login

Computer Vision Group
Faculty of Informatics
Technical University of Munich

Home
Application Form
Publications
Research
Data
Datasets
Multiview Datasets
3D Object in Clutter Recognition and Segmentation
ddff12scene
Deformable Shape Tracking Datasets
Deformable 3D Shape Matching
Monocular Visual Odometry Dataset
SLAM for Omnidirectional Cameras
3D Deformable Partial Shape Matching
RGB-D SLAM Dataset and Benchmark
Deformable 3D Shape

Home Data Datasets RGB-D SLAM Dataset and Benchmark **download**

Dataset Download

We recommend that you use the **'xyz'** series for your first experiments. The motion is relatively small, and only a small volume on an office desk is covered. Once this works, you might want to try the **'desk'** dataset, which covers four tables and contains several loop closures.

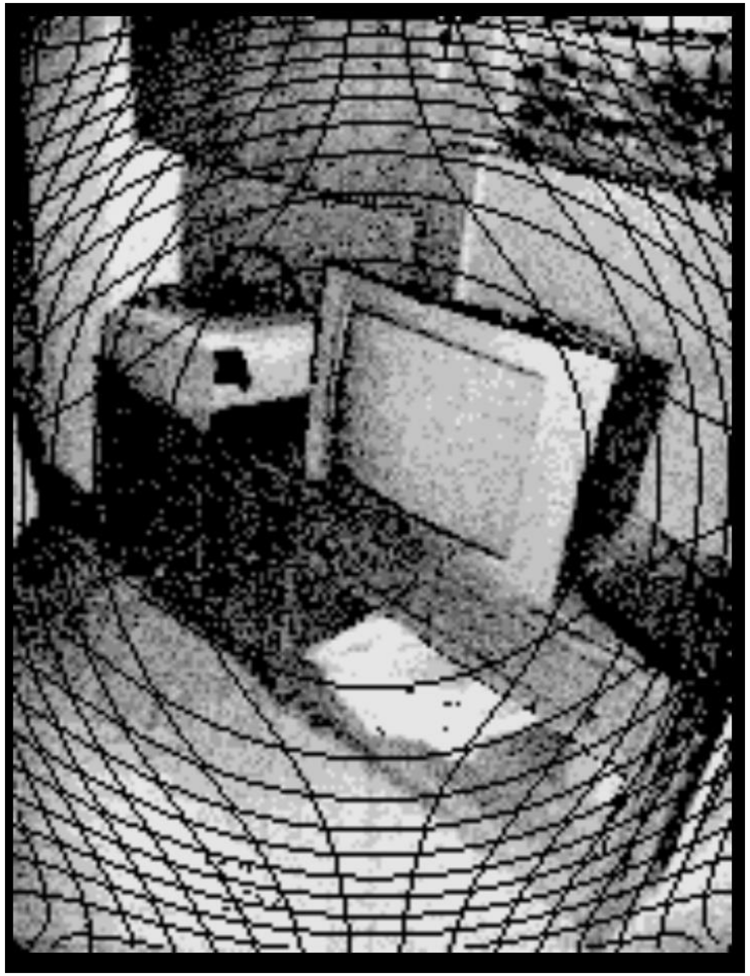
We are happy to share our data with other researchers. Please refer to the respective publication when using this data.

Remarks:

- The file formats are described here.
- The intrinsic camera parameters are here.
- We provide a set of useful tools for working with the dataset.
- The `*_validation` sequences do not contain ground truth. They can only evaluated using the online tool.

Sequence name	Duration	Length	Download	
Category: Testing and Debugging				
fr1/xyz	30.09s	7.112m	tgz (0.47GB)	more info
fr1/rpy	27.67s	1.664m	tgz (0.42GB)	more info
fr2/xyz	122.74s	7.029m	tgz (2.39GB)	more info
fr2/rpy	109.97s	1.506m	tgz (2.13GB)	more info
Category: Handheld SLAM				
fr1/360	28.69s	5.818m	tgz (0.45GB)	more info

Example: RGB and Depth Pair from Google Tango



Tasks

- Back-Projection
  - Use the given intrinsics, extrinsics and the camera trajectory to project the camera observation back to world space
  - Assign the color to the back-projected points
- Write a 3D mesh
  - Write an OFF file containing the back-projected position and color information
  - Make use of the grid structure of the observation to perform the triangulation

Projection Pipeline (from World to Pixels)

$$p_{world} := \text{arbitary point} \in \mathbb{R}^4 = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}^T$$

$$M_{camera} := \text{camera pose} \in \mathbb{R}^{4\times4}$$

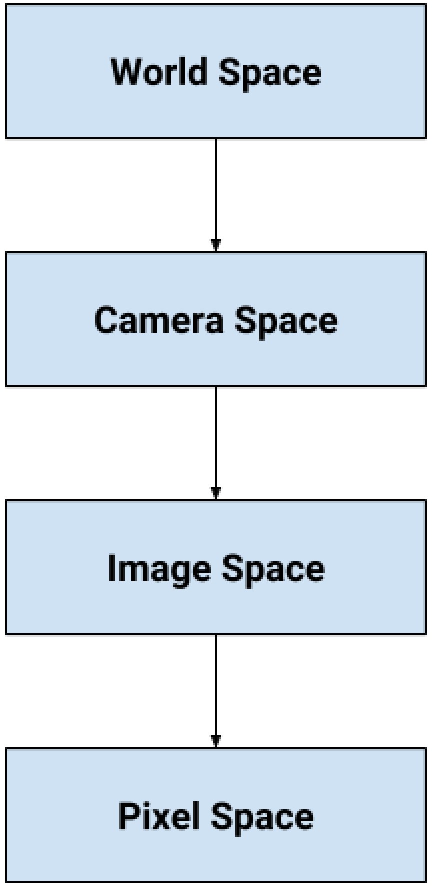
$$K := \text{intrinsic matrix} \in \mathbb{R}^{3\times3} = \begin{bmatrix} \frac{f\cdot w}{W} & 0 & c_x \\ 0 & \frac{f\cdot h}{H} & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$E := \text{extrinsic matrix} \in \mathbb{R}^{4\times4} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

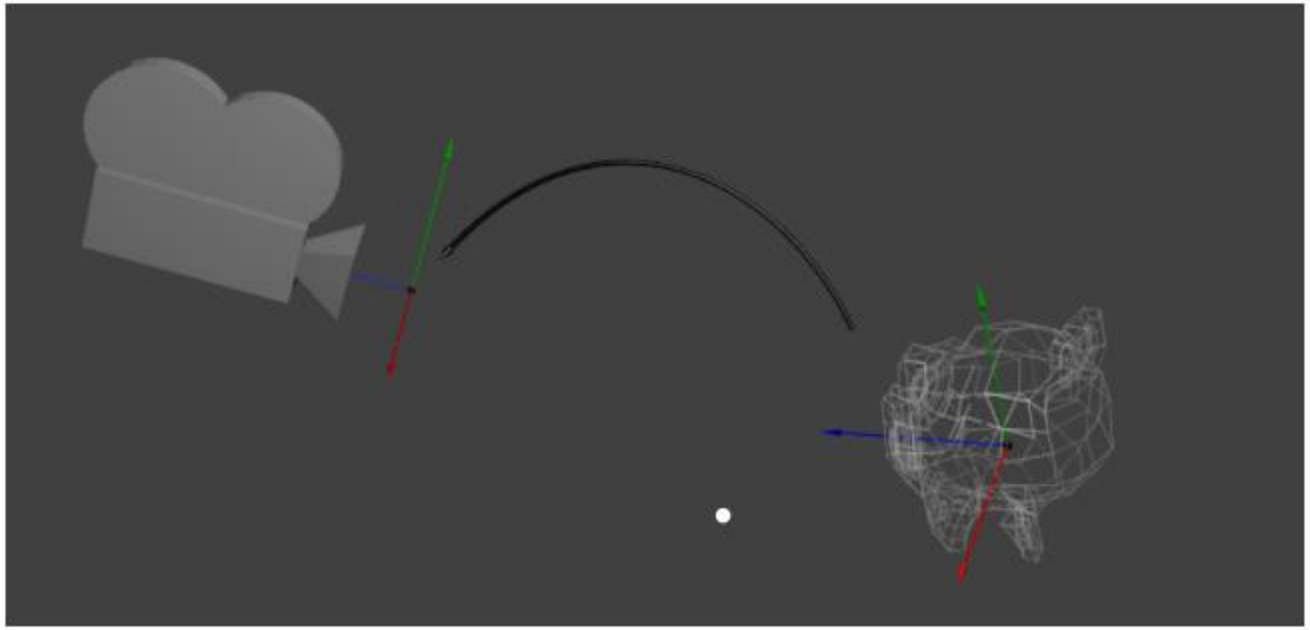
$$p_{camera} = E \cdot M_{camera} \cdot p_{world} \in \mathbb{R}^4$$

$$p_{image} = K \cdot \begin{bmatrix} I_3 & 0 \end{bmatrix} \cdot p_{camera} \in \mathbb{R}^3$$

$$p_{pixel} = \frac{1}{p_{image,z}} \begin{bmatrix} p_{image,x} \\ p_{image,y} \end{bmatrix} \in \mathbb{R}^2$$

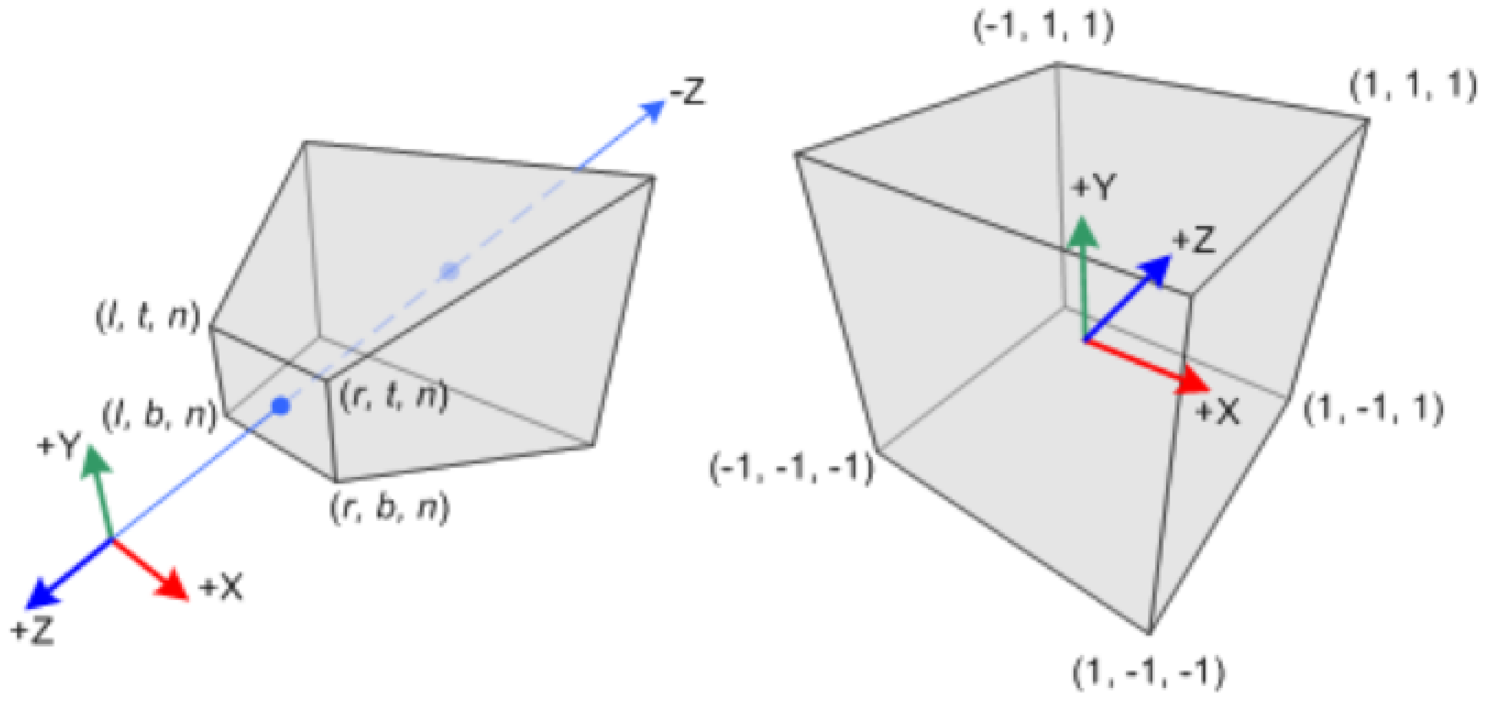


Projection Pipeline



Arbitrary point in the world E.g. point on the monkey	$p_{world} \in \mathbb{R}^3$
Camera pose Somewhere in the world with some look-at direction	$M_{camera} \in \mathbb{R}^{4\times4}$

Perspective Projection in CG



Perspective Frustum and Normalized Device Coordinates (NDC)

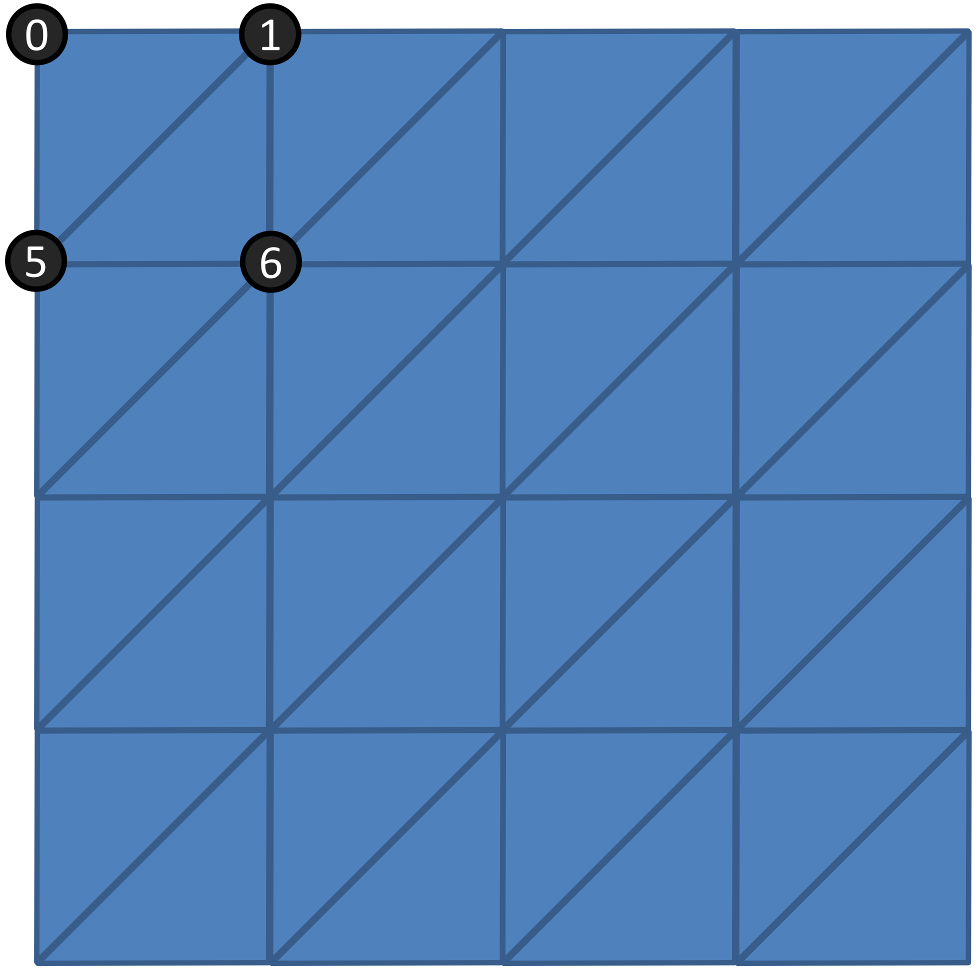
- http://www.songho.ca/opengl/gl_projectionmatrix.html
- https://www.scratchapixel.com/lessons/3d-basic-rendering/perspective-and-orthographic-projection-matrix/opengl-perspective-projection-matrix

Perspective Projection in CV

$$\begin{pmatrix} fov_X & 0 & c_x \\ 0 & fov_Y & c_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ \boldsymbol{z} \end{pmatrix} = \begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} \xrightarrow[\textbf{Dehomogenization}]{} \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} u'/_{w'} \\ v'/_{w'} \end{pmatrix}$$

- Keep track of the unmapped z values!
- For backprojection, perform the transformations in reverse order

Mesh Structure



Ensure consistent orientation of the triangles!

Example:

First triangle: 0-5-1

Second triangle: 5-6-1

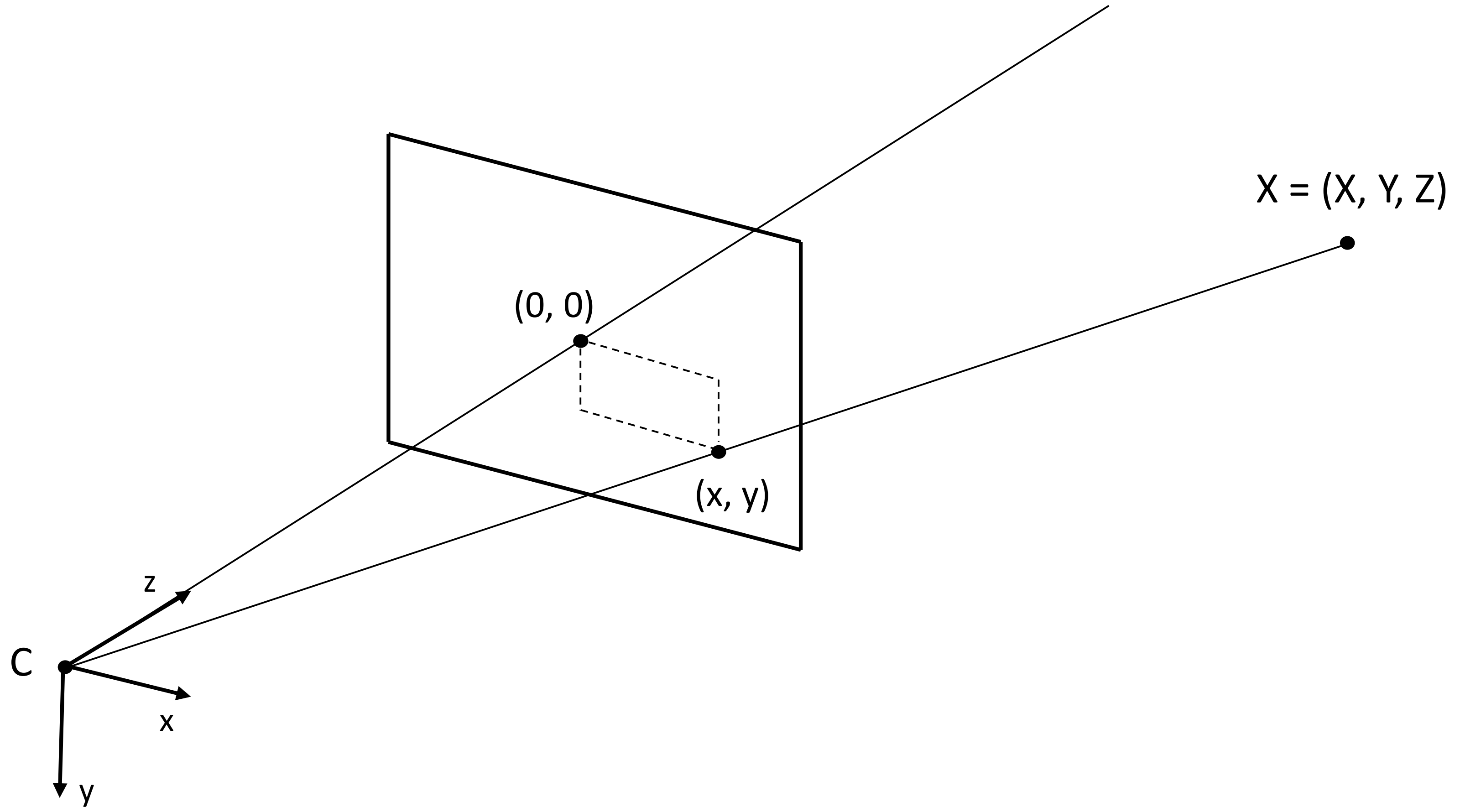
Visual Studio 2017 Community

- https://www.visualstudio.com/de/downloads/

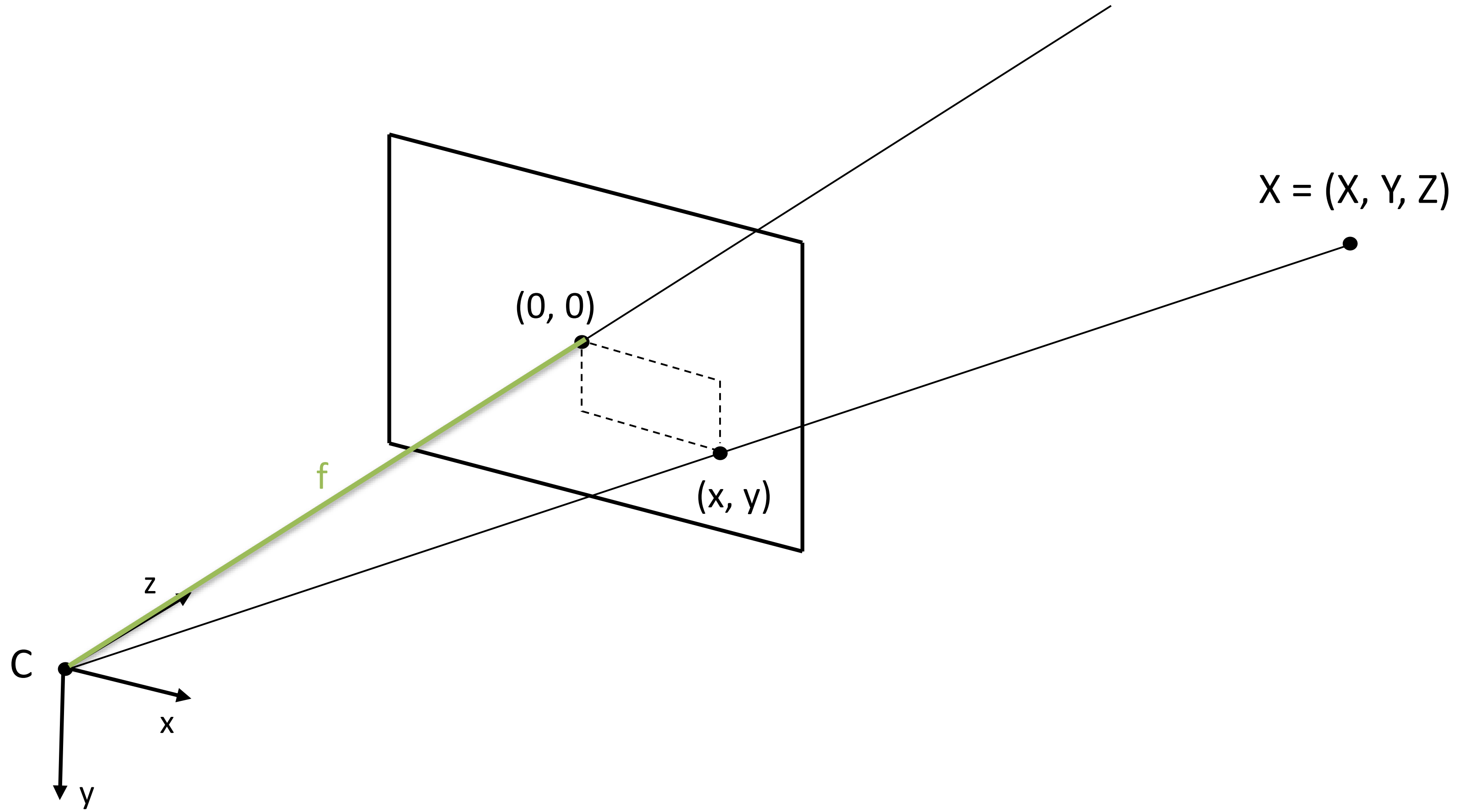
- Known issues:
  - fatal error LNK1104: cannot open file ‘gdi32.lib‘
    - https://stackoverflow.com/questions/33599723/fatal-error-lnk1104-cannot-open-file-gdi32-lib

Supplemental material for perspective projection

Pinhole camera model

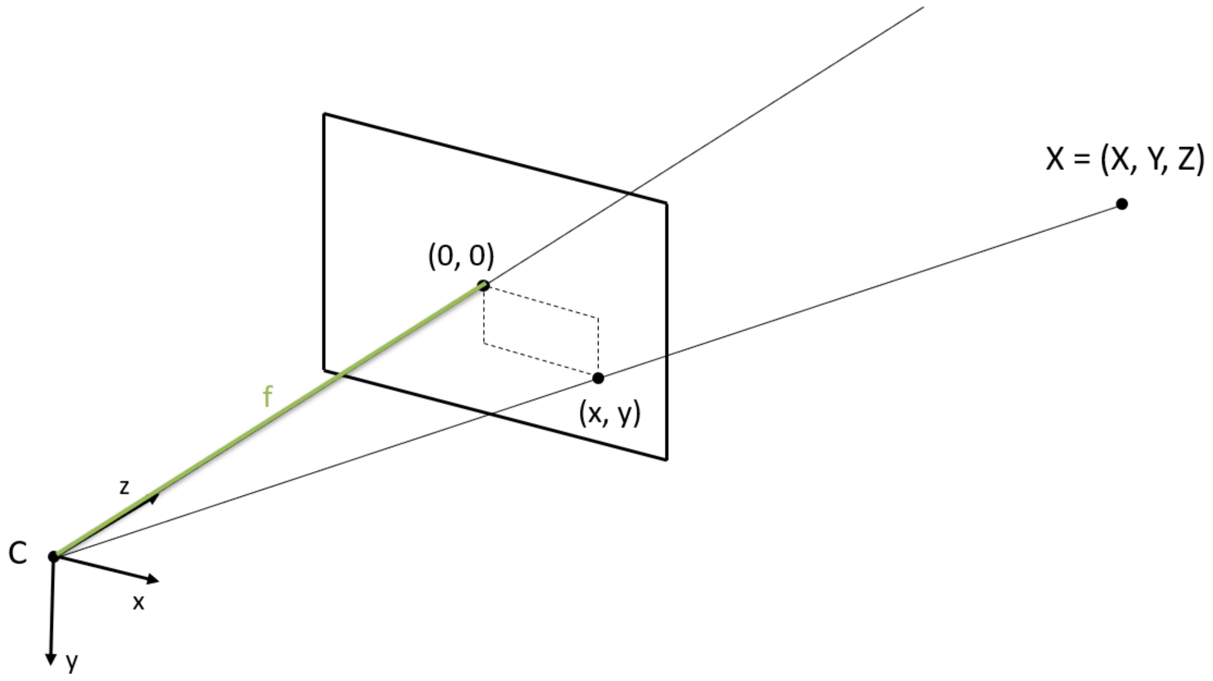


Pinhole camera model

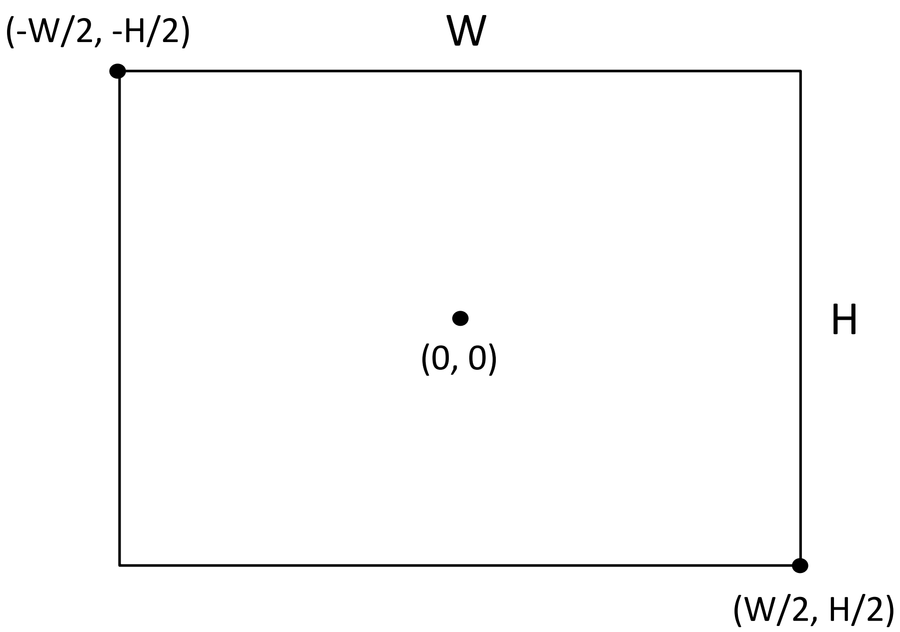


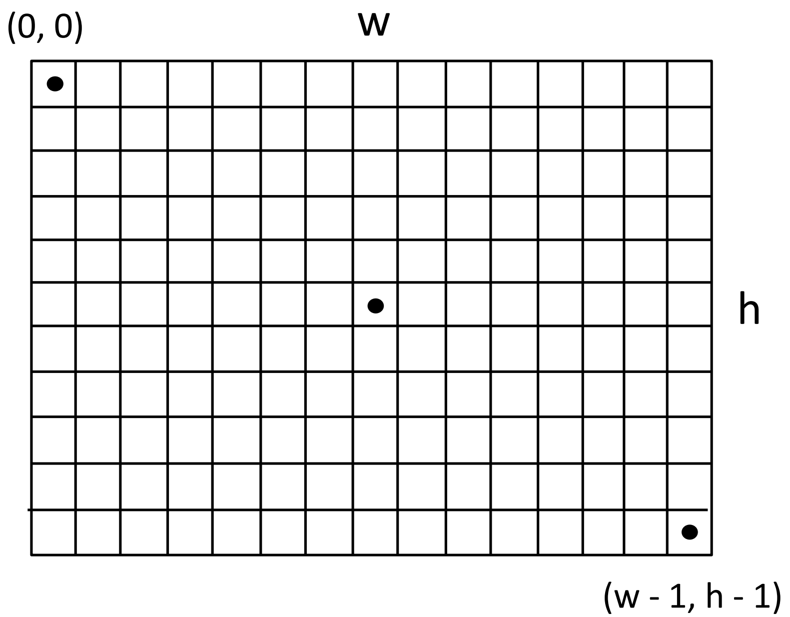
Pinhole camera model

$$\begin{pmatrix} x \\ y \end{pmatrix} = f \cdot \begin{pmatrix} X/Z \\ Y/Z \end{pmatrix}$$

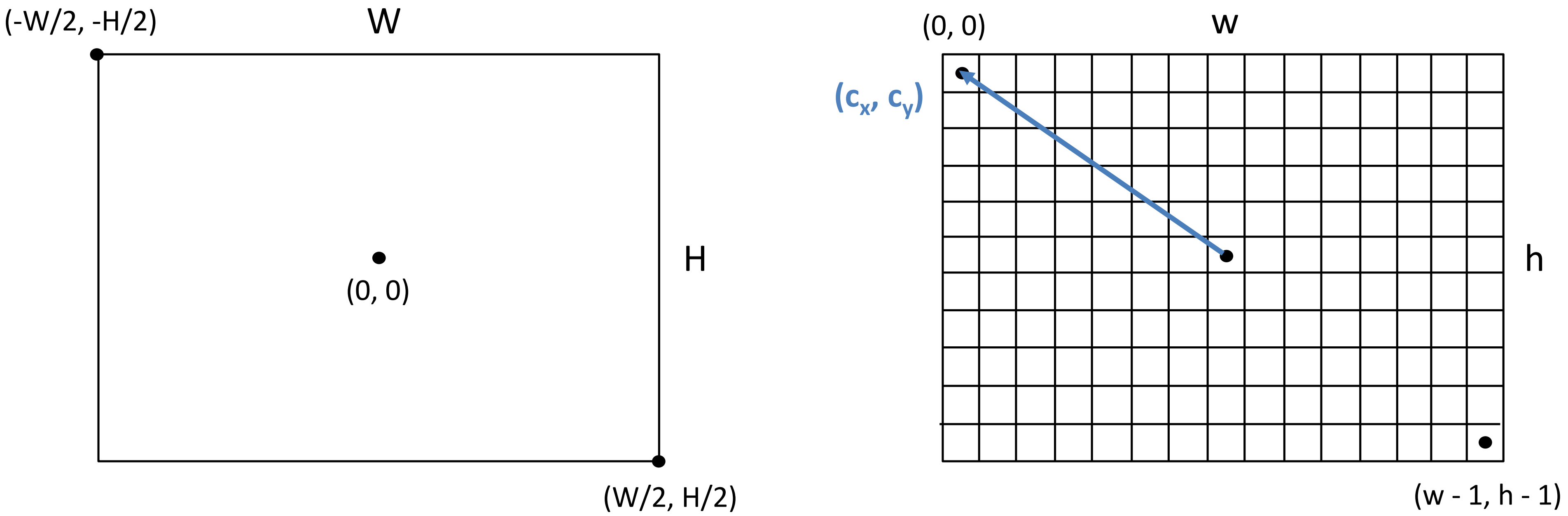


From sensor to pixels





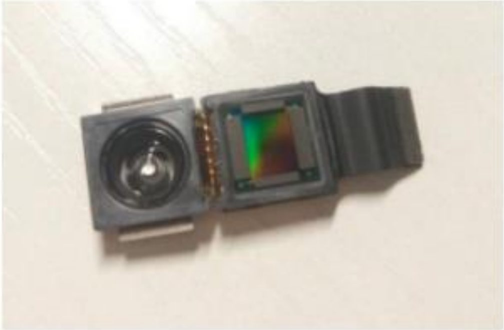
From sensor to pixels



Intrinsic parameters

Camera sensor

- Sensor width = 4.54 mm
- Sensor height = 3.42 mm
- focal length = 4.1 mm



Compare: Professional cameras have 35mm sensor!

Intrinsic matrix

$$
\begin{aligned}
f &:= \text{focal length} = 4.1mm \\
W &:= \text{sensor width} = 4.54mm \\
H &:= \text{sensor height} = 3.42mm \\
w &:= \text{image width} = 640 \\
h &:= \text{image width} = 480 \\
c_x &:= \text{image center x} = 320 \\
c_y &:= \text{image center y} = 240
\end{aligned}
$$

$$
\text{Resulting intrisic matrix}: \begin{bmatrix} \frac{f \cdot w}{W} & 0 & c_x \\ 0 & \frac{f \cdot h}{H} & c_y \\ 0 & 0 & 1 \end{bmatrix}
$$