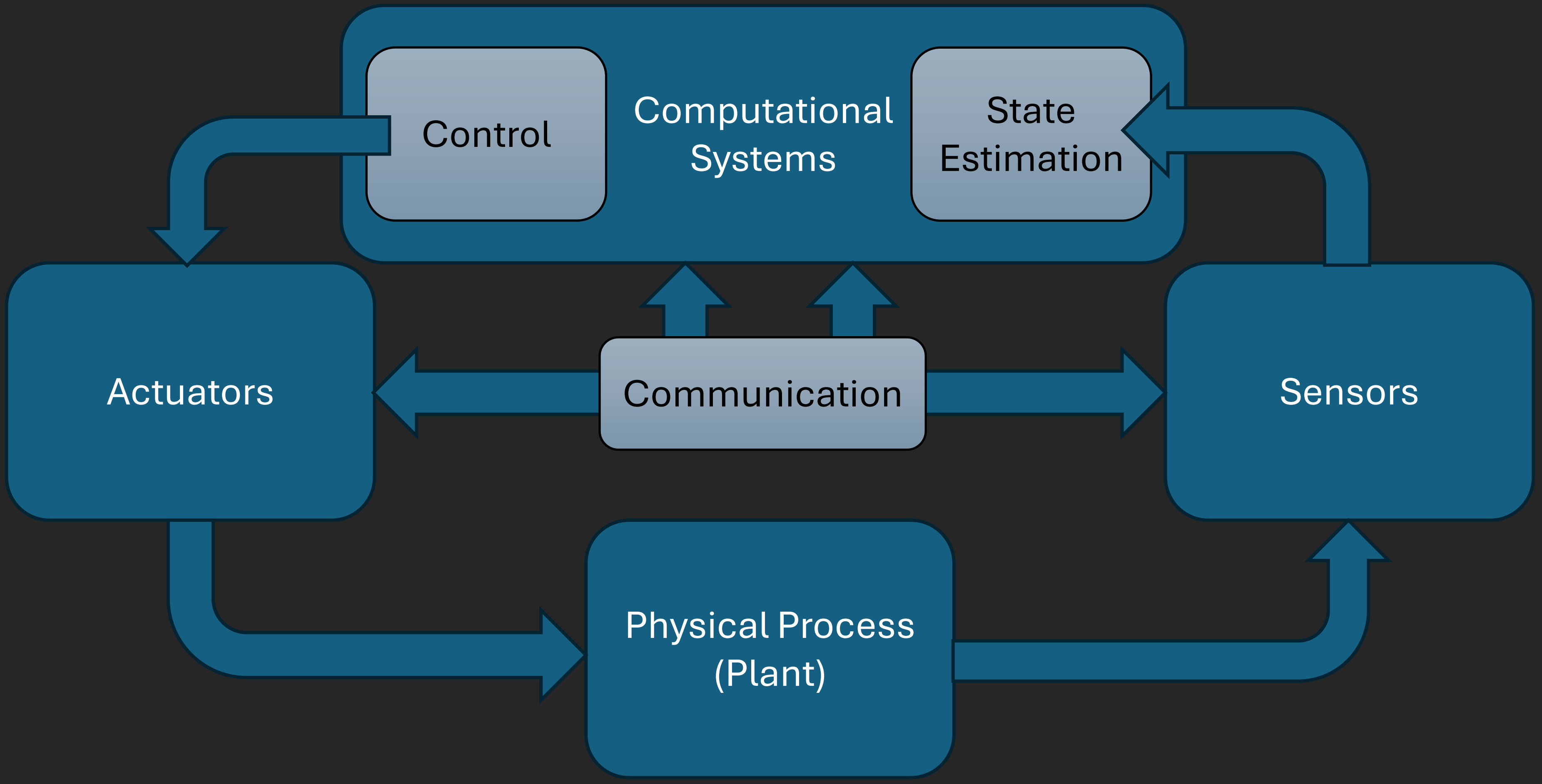
Cyber-Physical Systems

Dr. Jonathan Jaramillo

Computer Vision

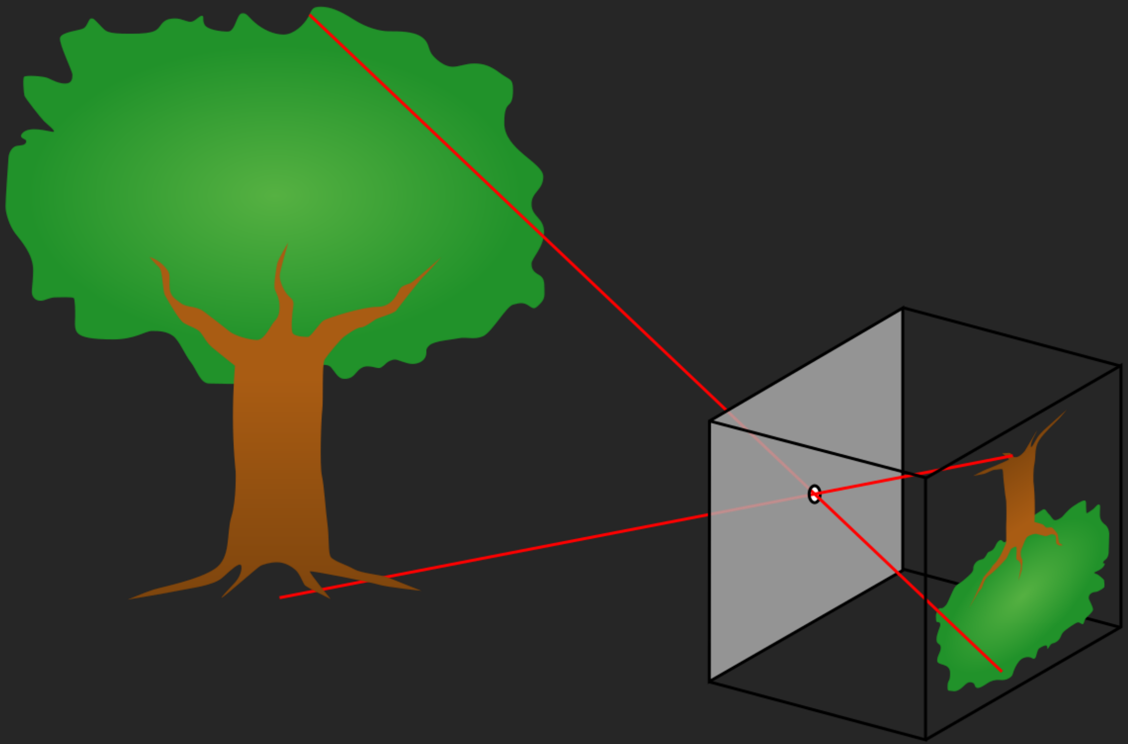
What are Cyber-Physical Systems?

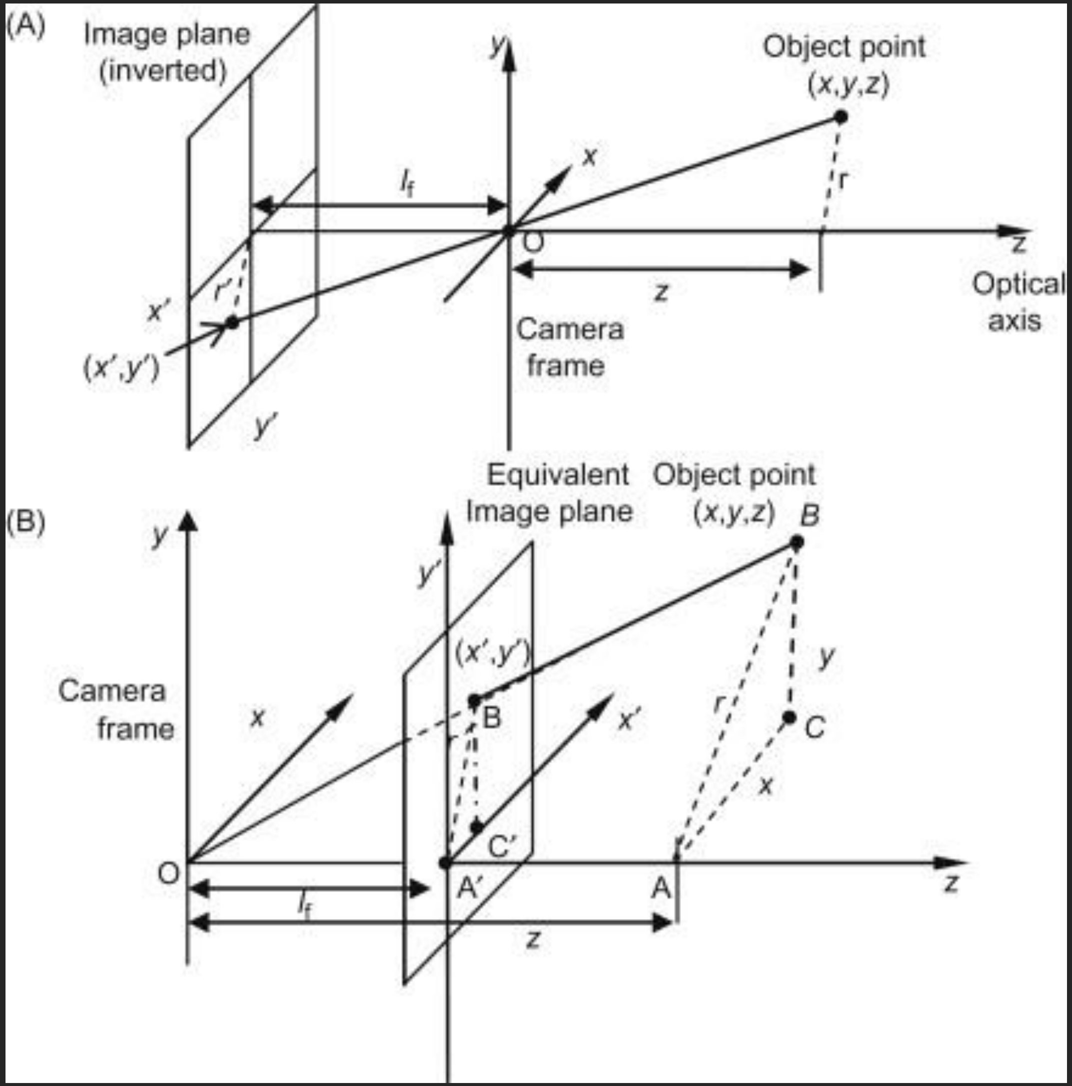


Cameras

- Light passes through a lens and produces a focused image on a sensor
  - Charged couple device (CCD) or Complementary metal-oxide-semiconductor (CMOS)
- Lense causes aberrations and distortions
- Use a first order approximation to the behavior of light
  - “Pinhole Model”

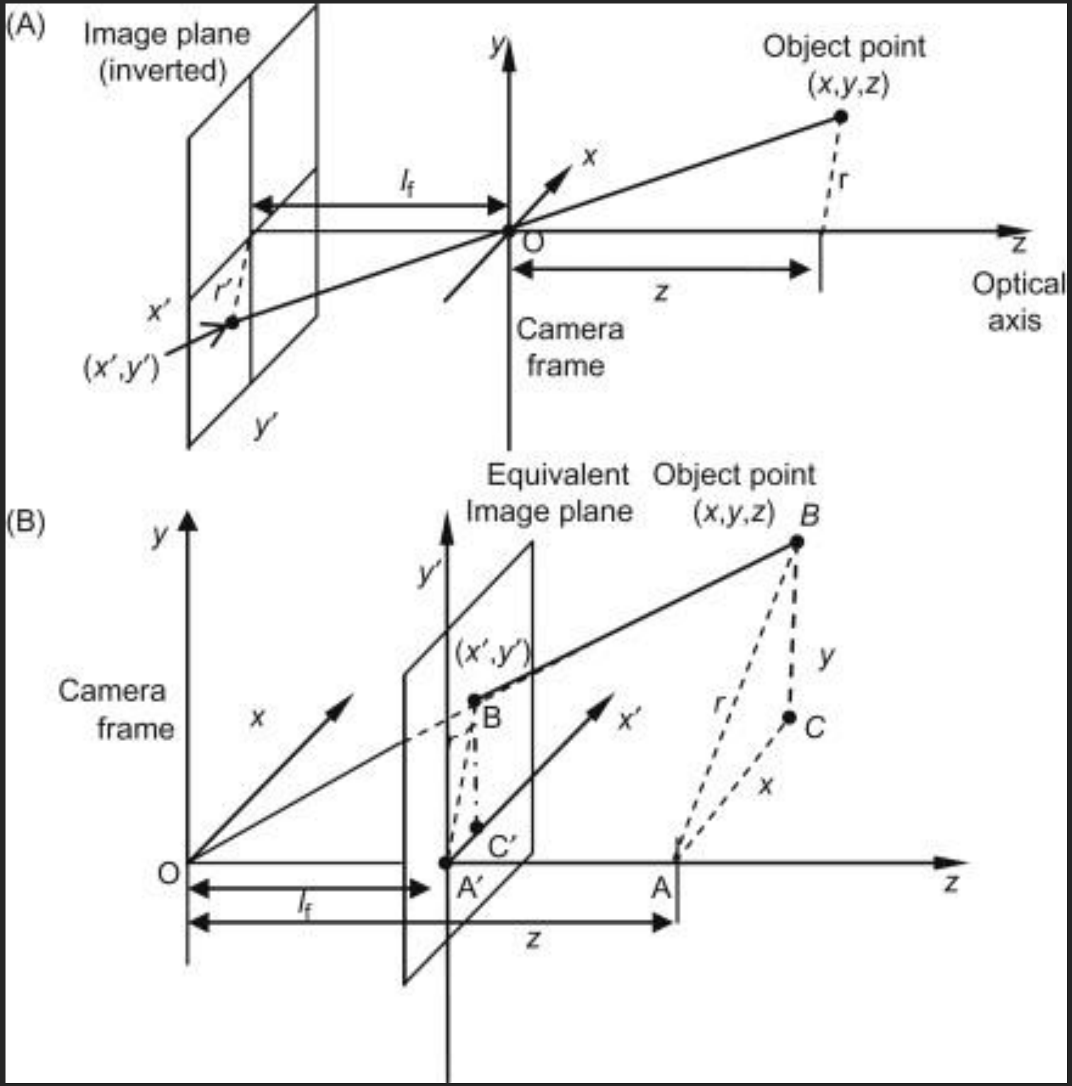
Pinhole Camera Model





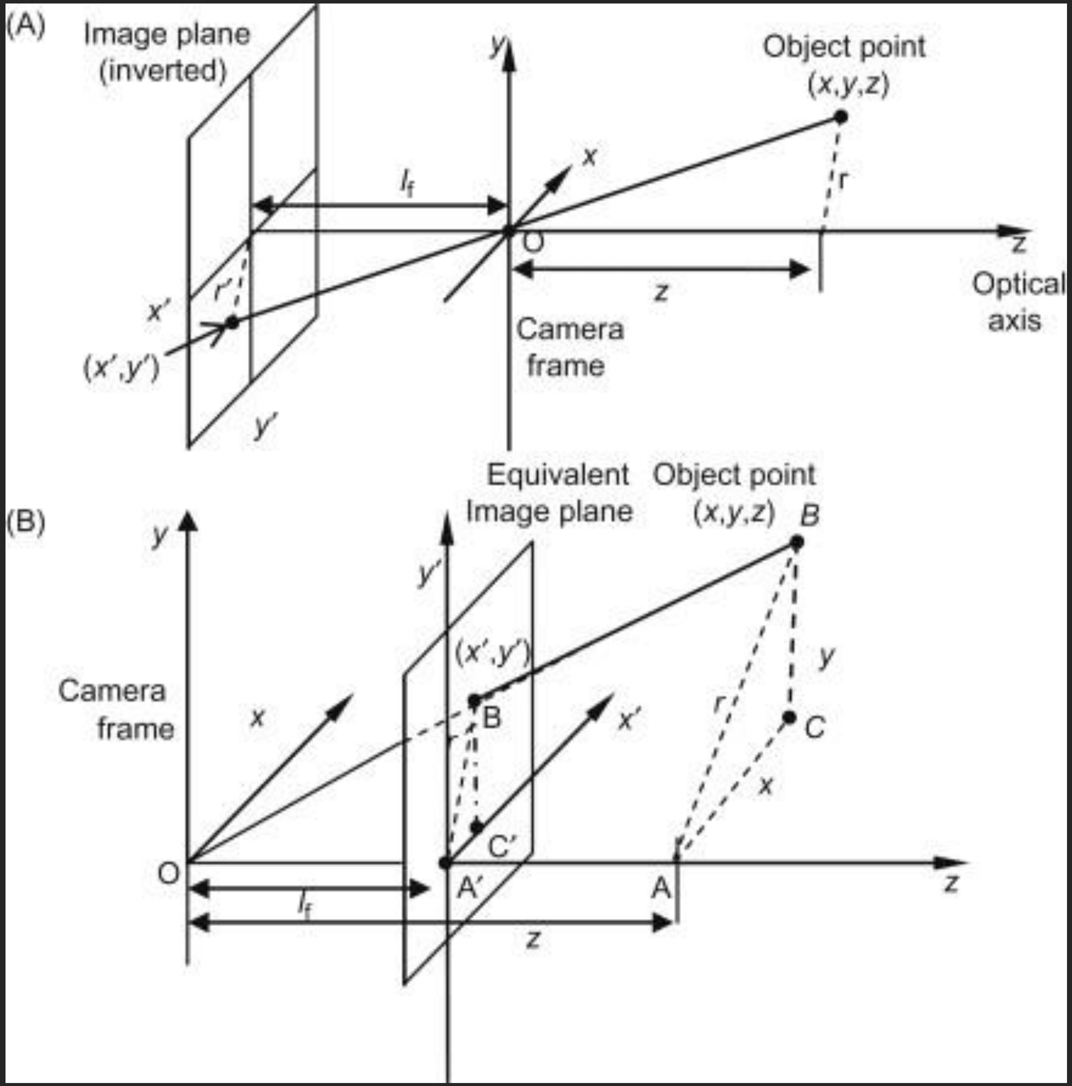
Pinhole Camera Model

- $x' = f\frac{x}{z}$
- $y' = f\frac{y}{z}$
- $s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$



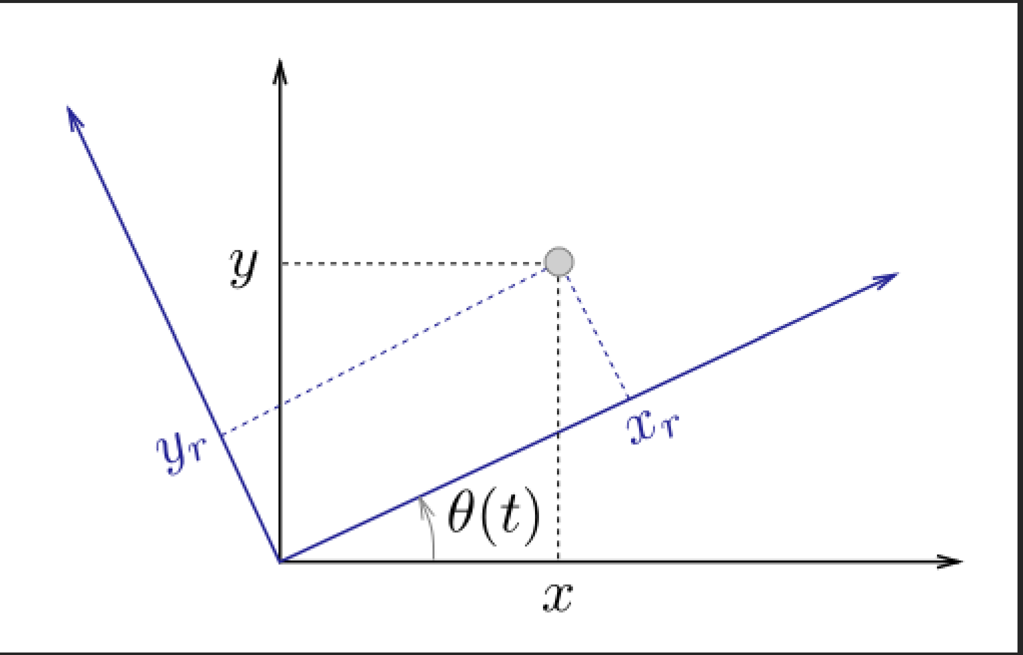
Pinhole Camera Model

- $x' = \frac{f}{s}x + c_x$
- $y' = \frac{f}{s}y + c_y$
- $s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$
- https://ksimek.github.io/2013/08/13/intrinsic/



Rotation Matrix

- Matrix that converts a point in space to a rotated coordinate system
- $R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$
- $\begin{bmatrix} x_r \\ y_r \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$



Rotation Matrix in 3D

- $R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}$

- $R_y = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$

- $R_z = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- $R = R_x R_y R_z$

- $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$

Rotation + Translation

- $\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}$

- $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{r}_{11} & r_{12} & r_{13} & \Delta x \\ \mathrm{r}_{21} & r_{22} & \mathrm{r}_{23} & \Delta y \\ r_{31} & r_{32} & r_{33} & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

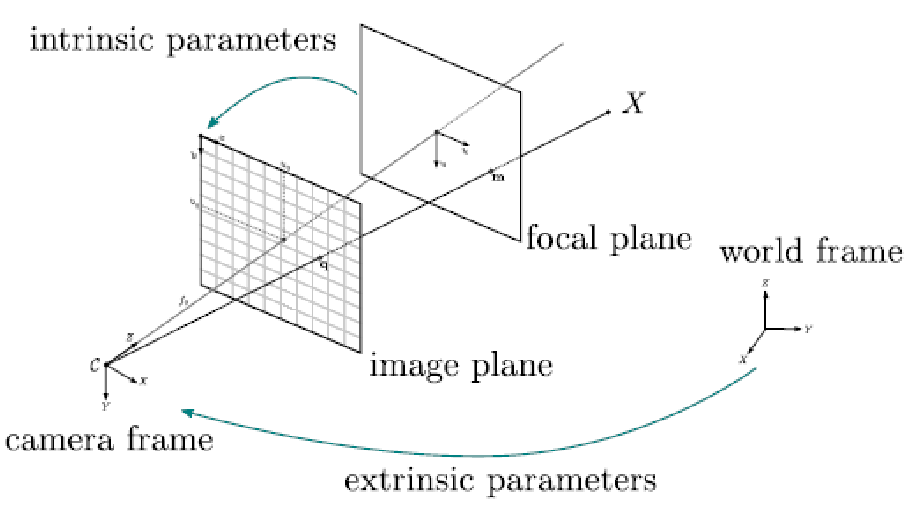
- $v' = [R|t]v$

Extrinsic and Intrinsic Matrices

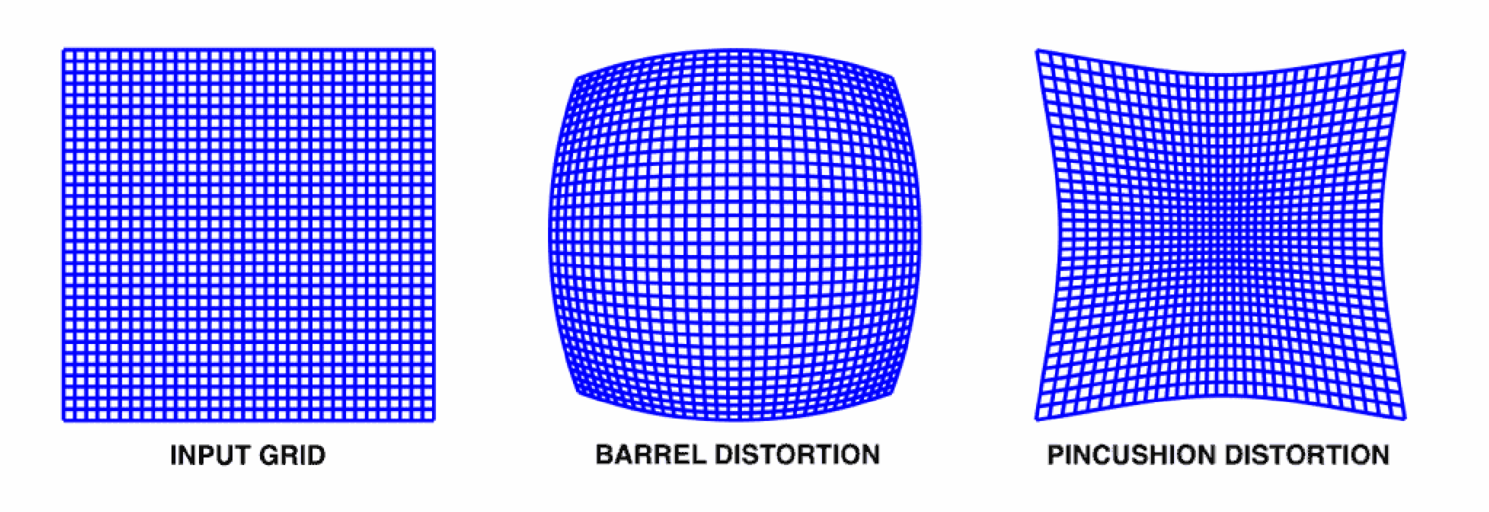
- Camera Extrinsic Matrix – Converts from world coordinate system to camera coordinate system
  - $\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{r}_{11} & r_{12} & r_{13} & \Delta x \\ \mathrm{r}_{21} & r_{22} & \mathrm{r}_{23} & \Delta y \\ r_{31} & r_{32} & r_{33} & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$
- Camera Intrinsic Matrix – Converts from 3-space in camera coordinate system to image plane
  - $s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$

Camera Matrix

- Camera Matrix – Converts from world coordinates to image plane
- $s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{r}_{11} & r_{12} & r_{13} & \Delta x \\ \mathrm{r}_{21} & \mathrm{r}_{22} & \mathrm{r}_{23} & \Delta y \\ r_{31} & r_{32} & r_{33} & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$
- $s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathrm{K}[\mathrm{R}|\mathrm{t}] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$



Distortion Models



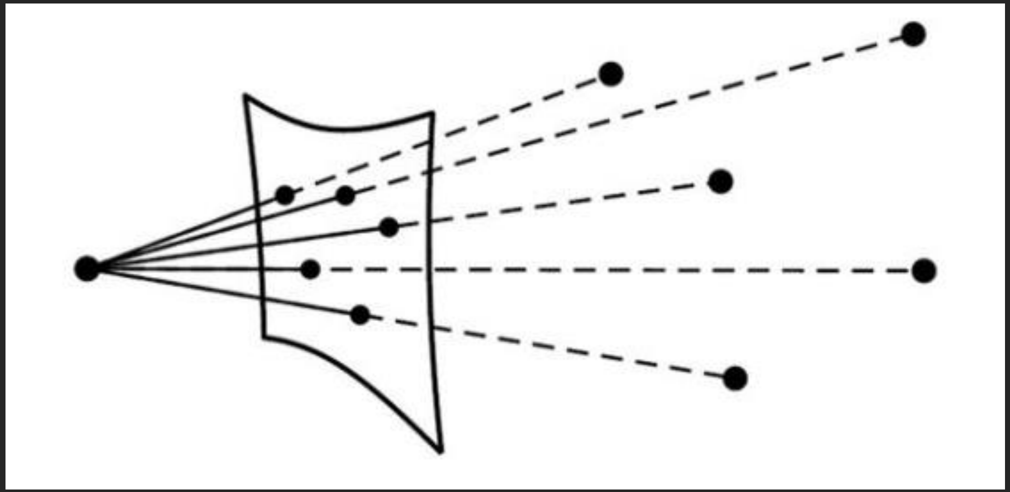
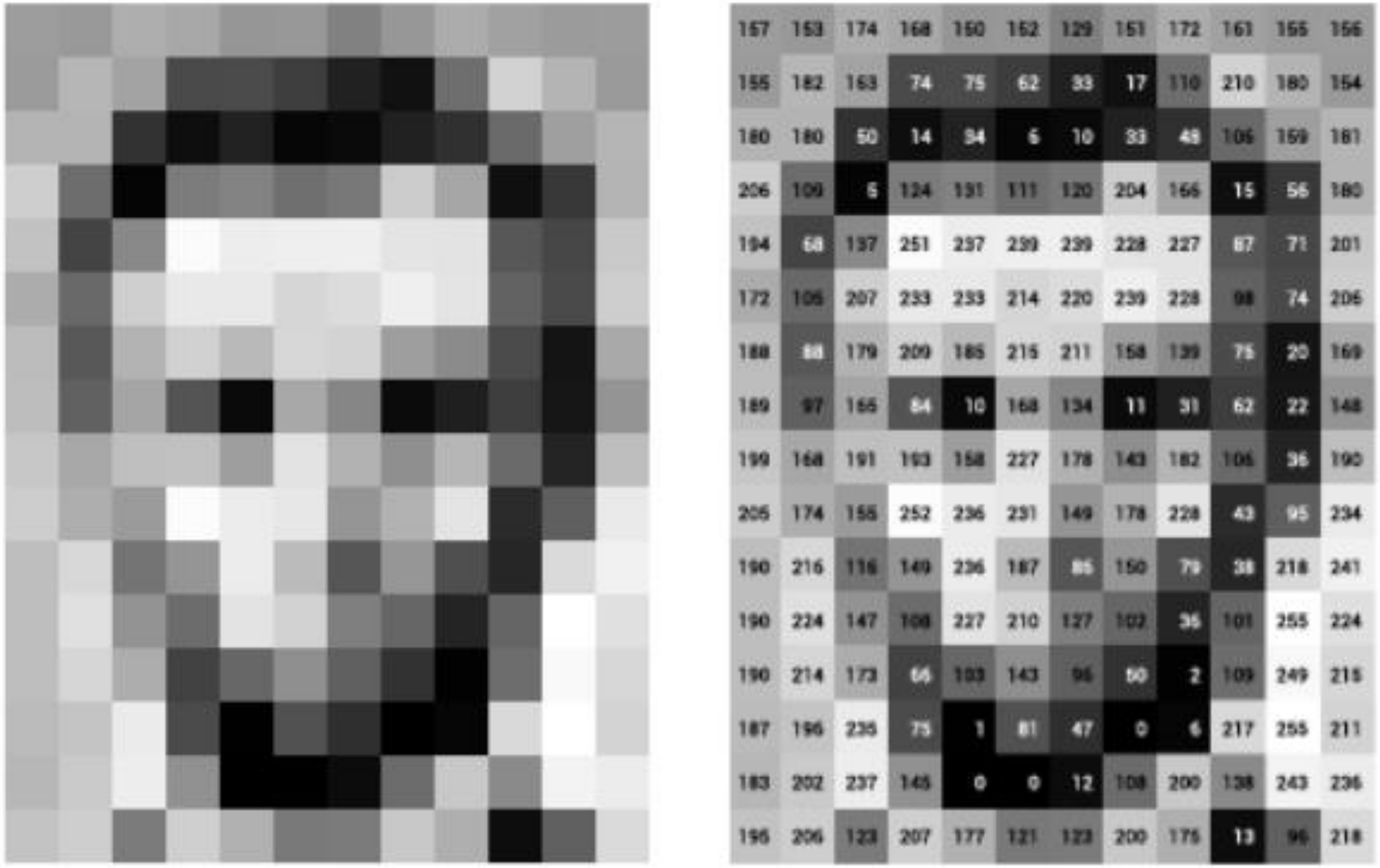


Image as Matrix

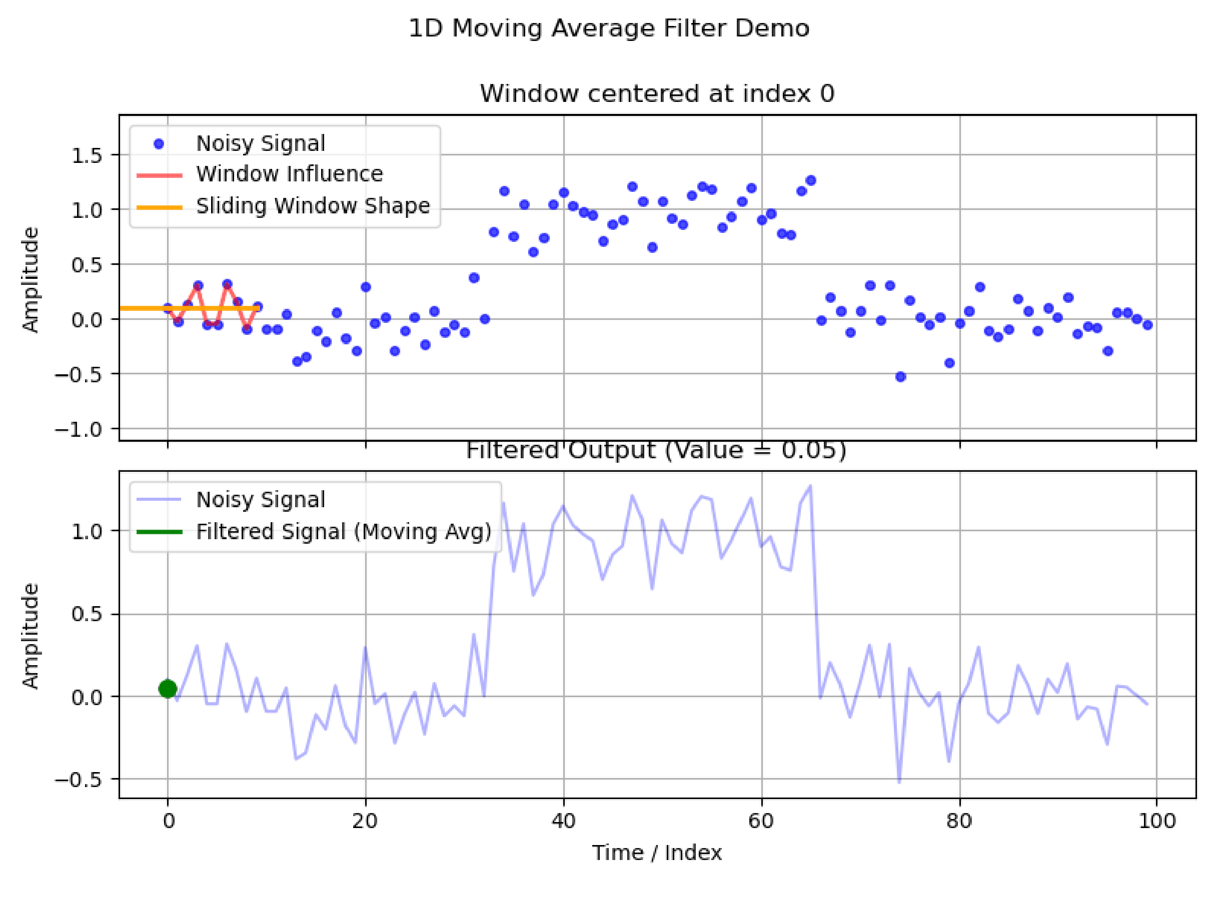


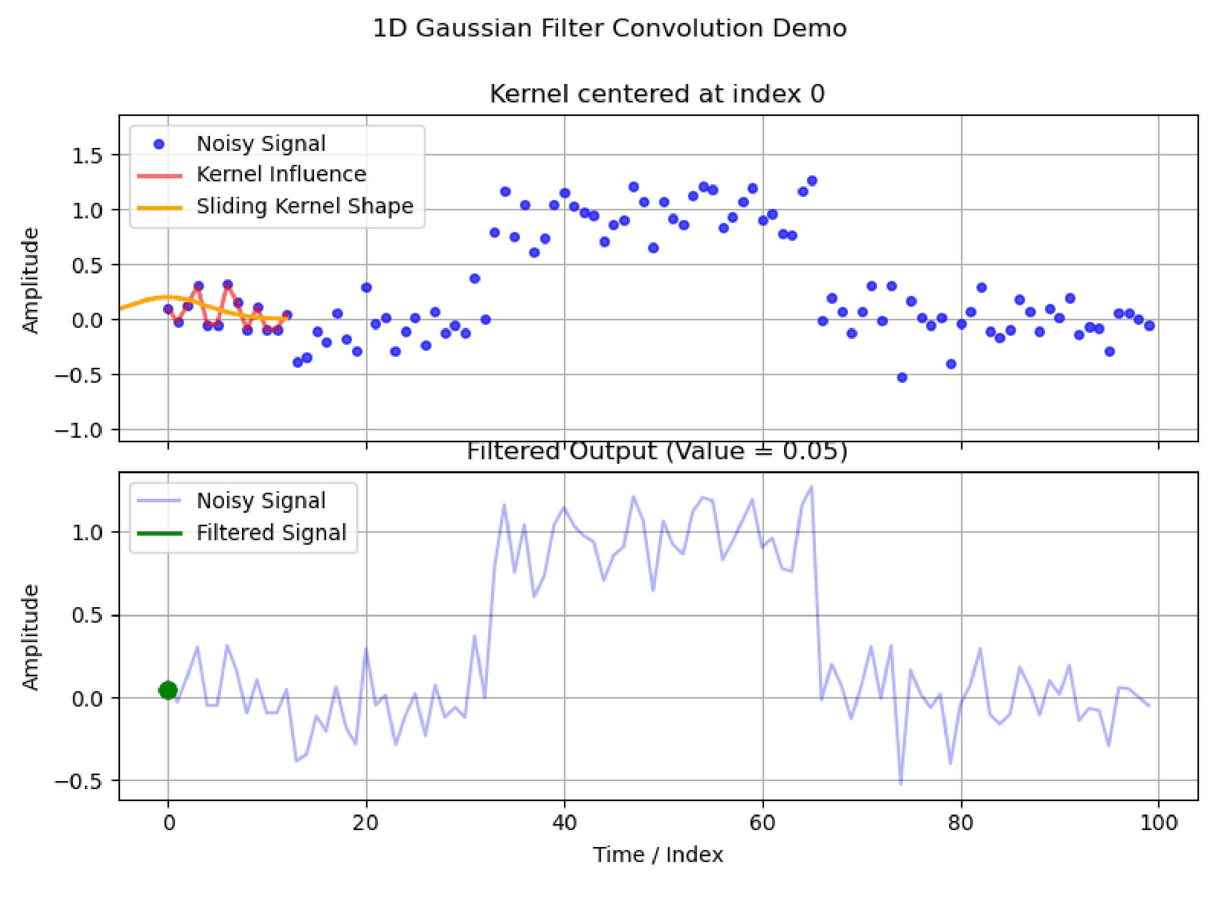
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Basic Image Manipulation

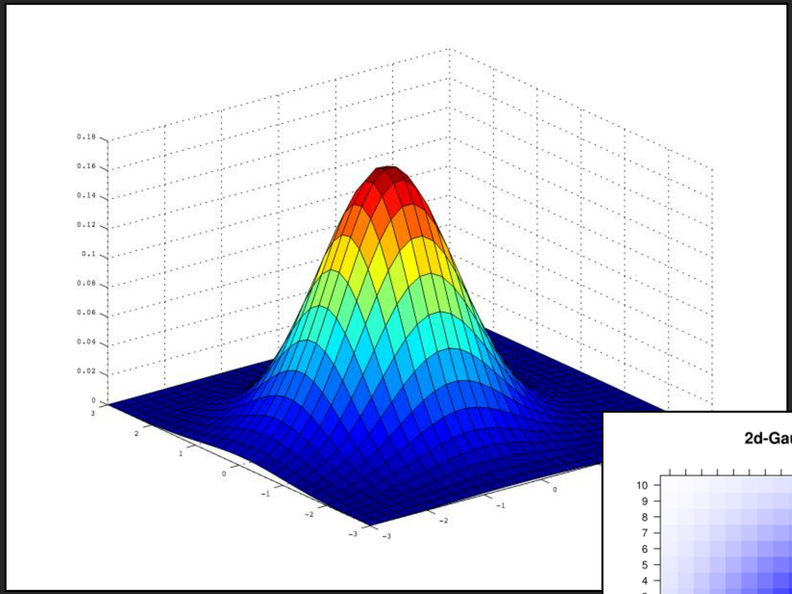
- Brightness
  - Increase or decrease all pixel values by a given factor
- Thresholding
  - Binary classification of pixels based in intensity
- Color isolation
  - Isolation individual color channels

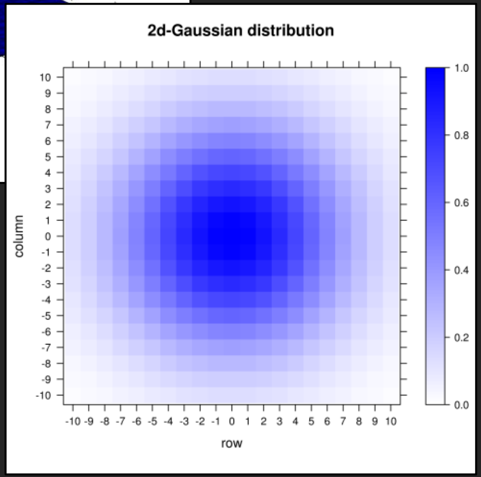
Filtering in 1D



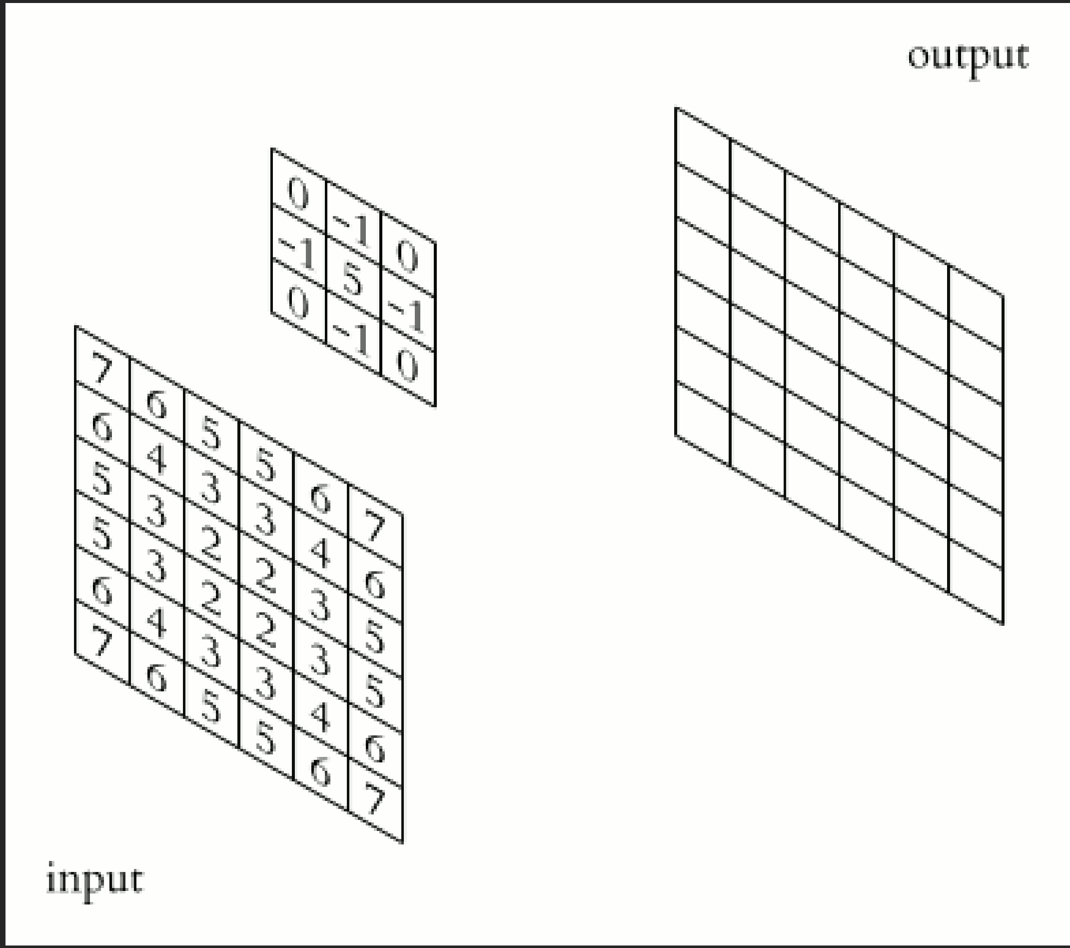


Filtering in 2D

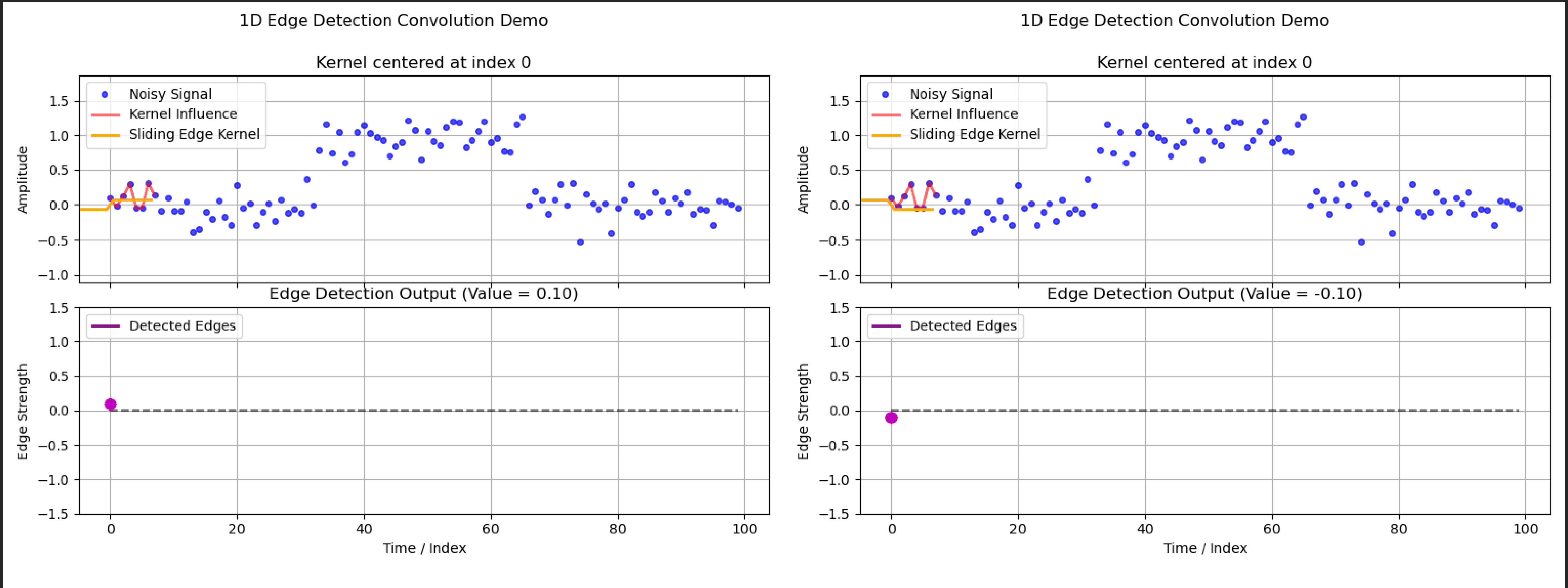




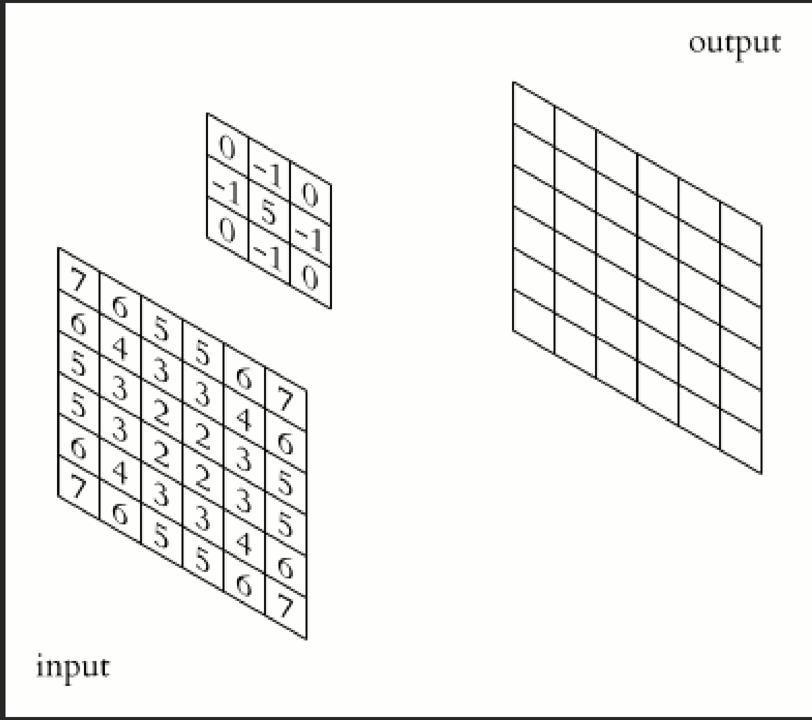
Filtering in 2D



Edge Detection in 1D



Edge Detection in 2D



-1	0	+1
-2	0	+2
-1	0	+1

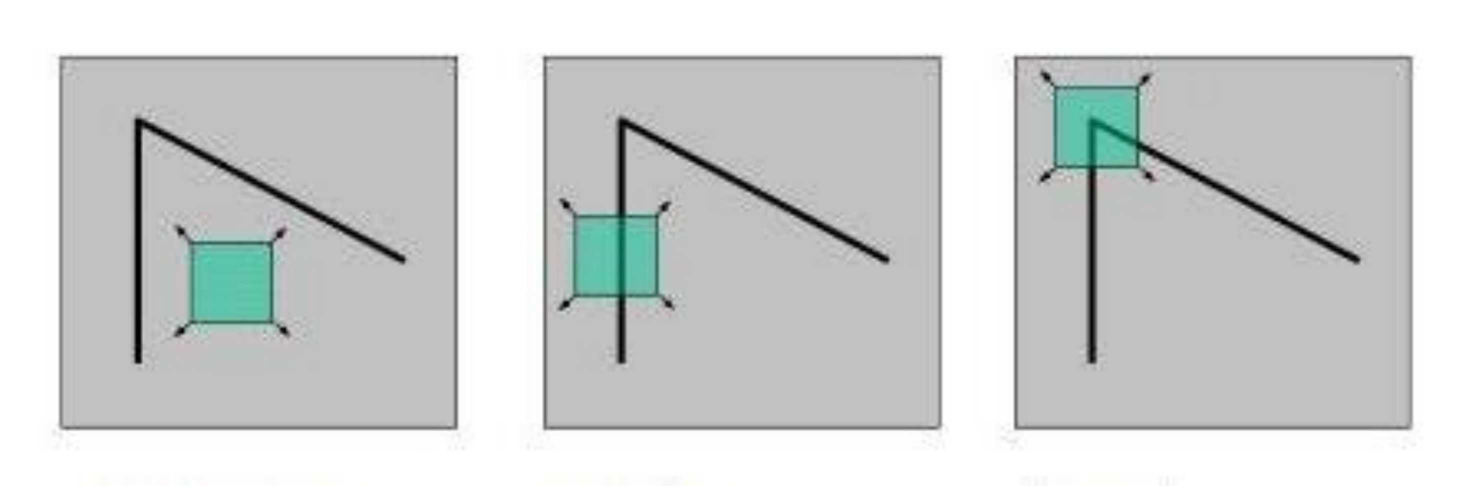
Gx

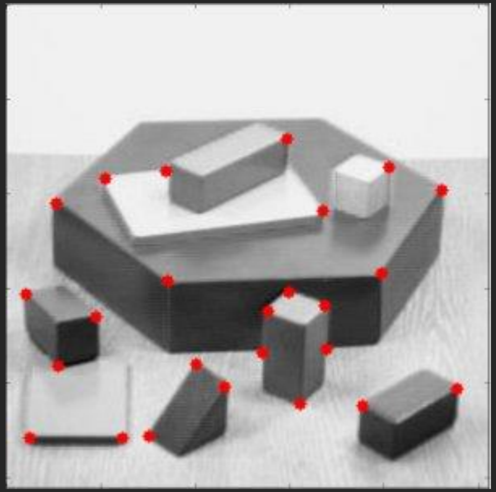
+1	+2	+1
0	0	0
-1	-2	-1

Gy

Corner Detection

- Slide a small window over an image
  - If the pixel intensity doesn't change – it's a "flat" part of the image
  - If the pixel intensity changes in one direction and not the other, there is an edge
  - If the pixel intensity changes in both directions, it's a corner



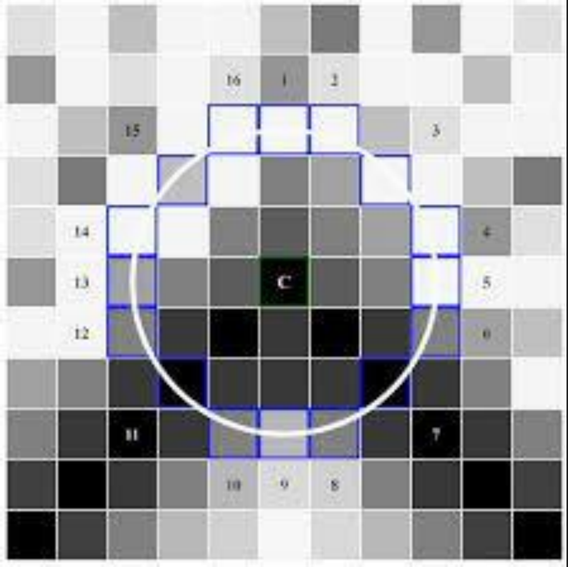


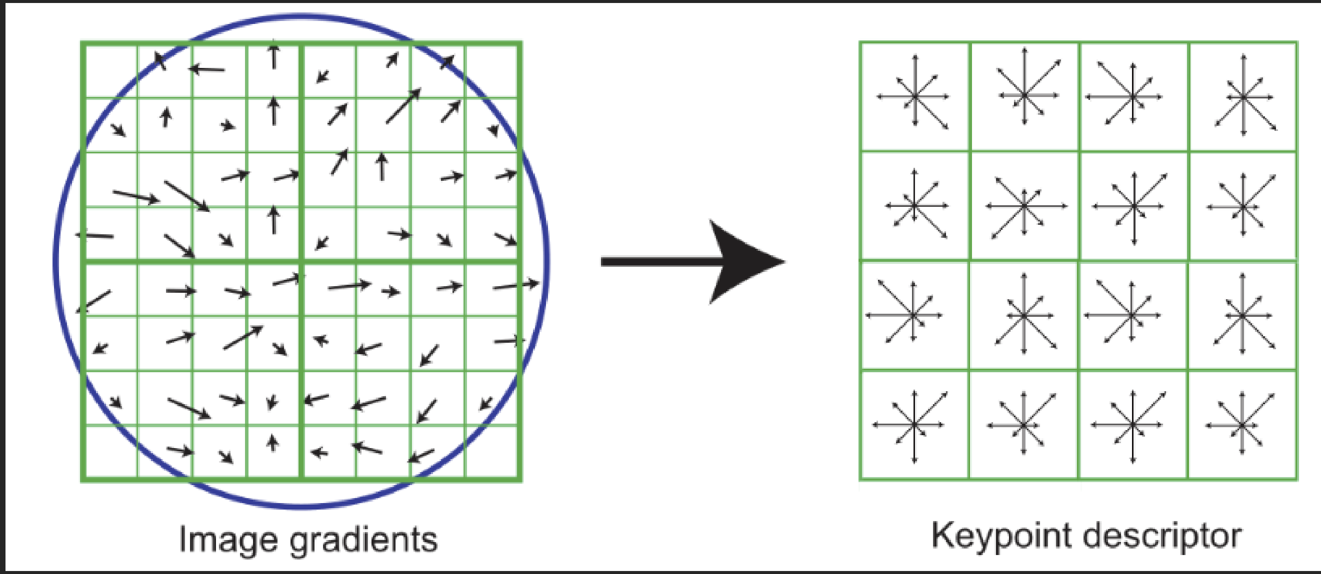
Optical Flow – Lucas-Kanade

- A point (x, y) in frame t moves to $(x + u, y + v)$ in frame $t + \Delta t$, its intensity stays the same
  - $I(x, y, t) = I(x + u, y + v, t + \Delta t)$
- If the motion is small, you can do a Taylor series expansion
  - $I(x + u, y + v, t + \Delta t) \approx I(x, y, t) + I_x u + I_y v + I_t$
- Assuming the intensity stays the same
  - $I_x u + I_y v + I_t = 0$
- Apply this to a small window

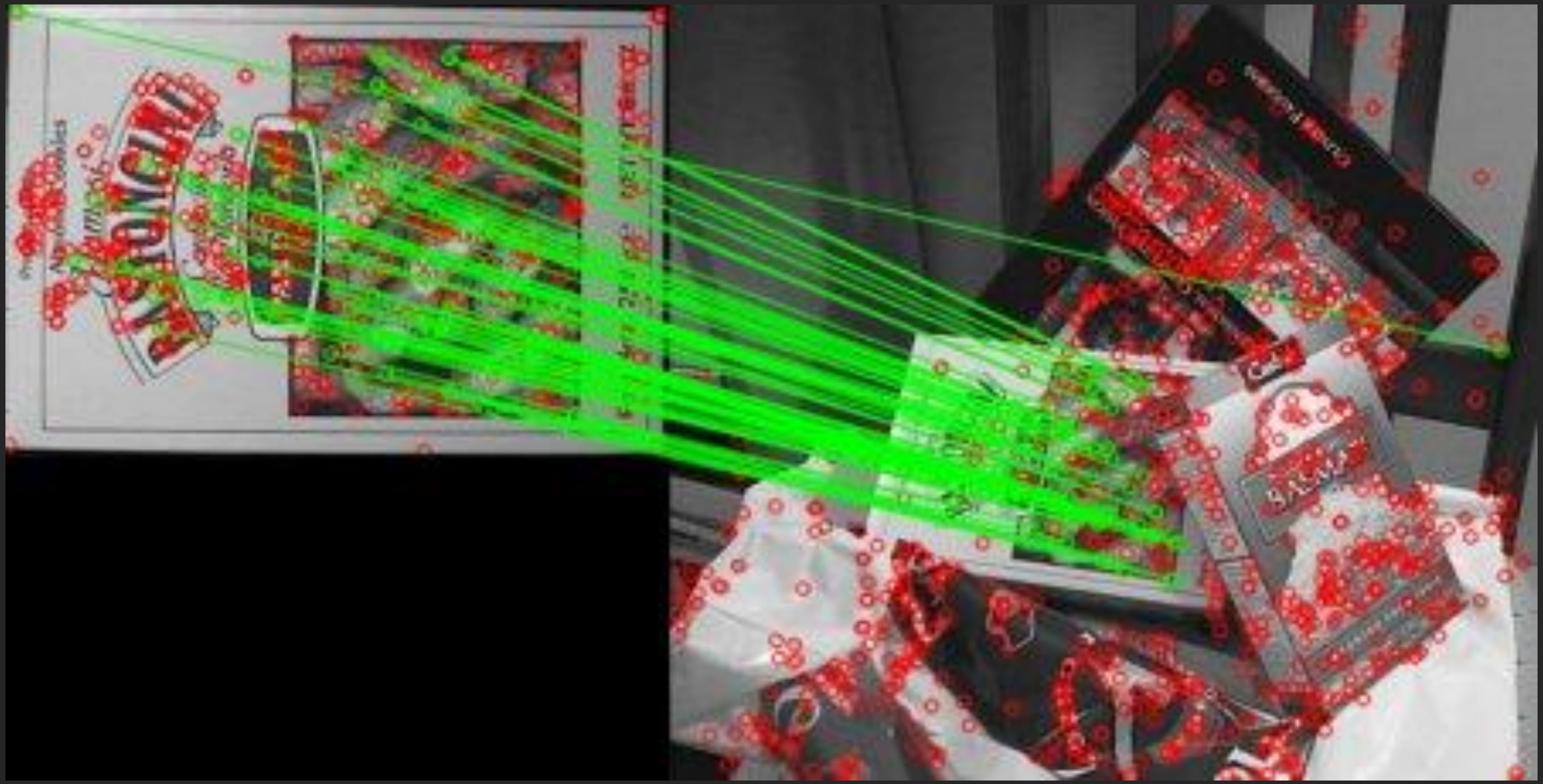
Feature Descriptors

- Mathematical way of “describing” the patch of image around a corner or other key point
- Allows the ability to match corners in two different images



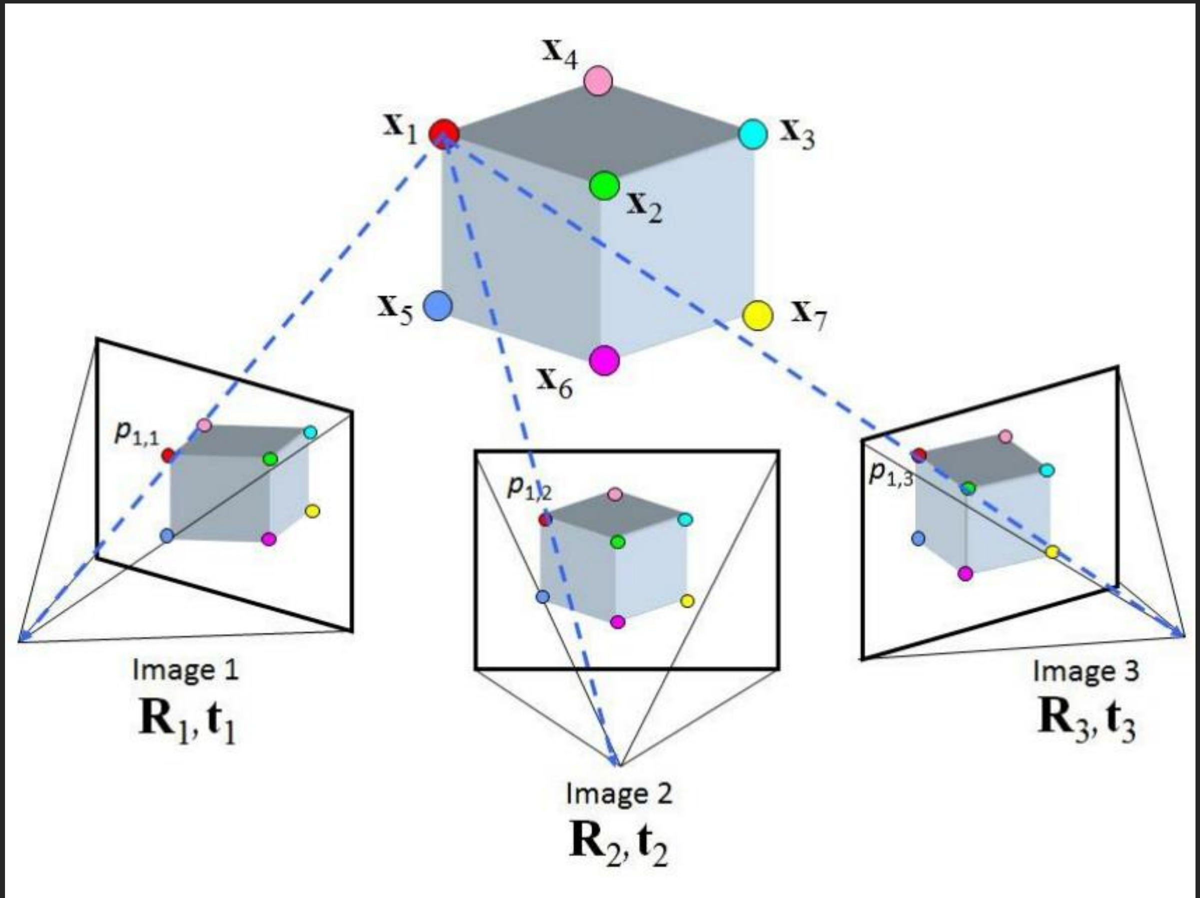


Feature Matching

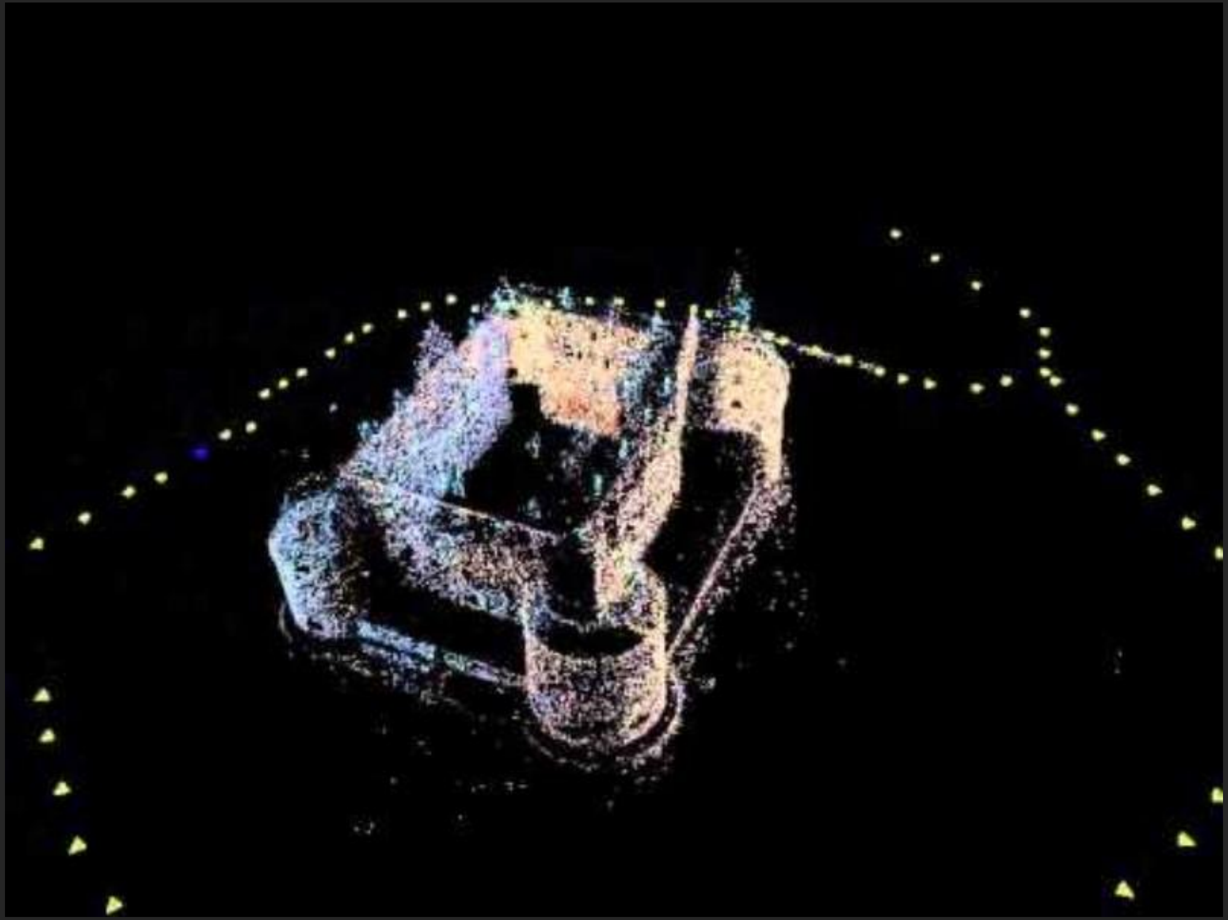


Structure From Motion

- Assume 6 unknowns per image and 7 unknown points
  - (6 x 3) + (7 x 3) = 39 unknowns
- Assume 7 points, 3 images, 2 knowns per point
  - 7 x 3 x 2 = 42 knowns



Structure From Motion



SLAM