Image-Based Situation Awareness Audit 1.3.2018

Sakari Lampola

Previous Audit 11.1.2018

Previous Audit

Open questions:

- Role of classical object tracking alrorithms?
- What to do with multiple bounding boxes around one object?
- Appropriate minimum confidence level?
- What to do with false detections inside other objects?
- What to do with false detections from the background?
- How to set Kalman filter parameters for image object filtering?
- Hungarian algorithms special case for hidden objects

To do:

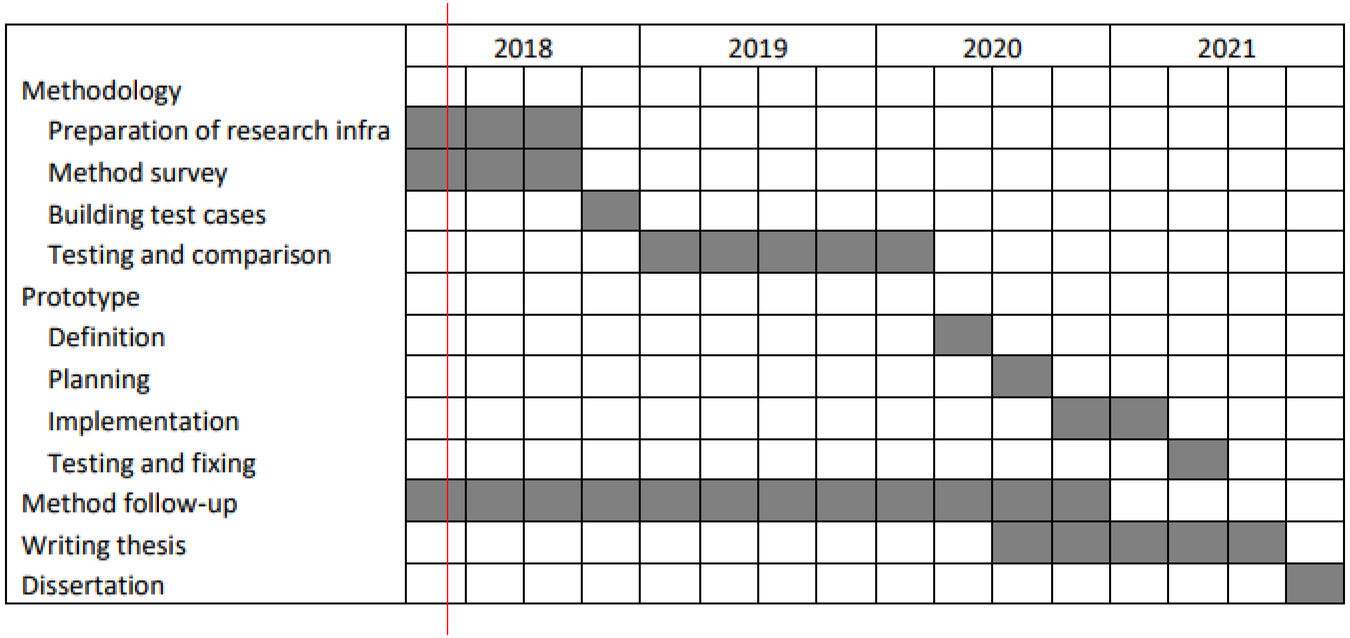
- Close open questions
- Image object status
- Image object velocity estimation
- Probabilistic approach for matching detected and image objects
- 2d -> 3d transformation
- World object state estimation

Other:

- Semantic segmentation
- Organisations to follow: ICCV, ICRA, NIPS, IROS, arXiv
- Camera motion (yaw, pitch)
- Grid or continuos presentation?
- Class specific attributes
- Object history

Project Plan

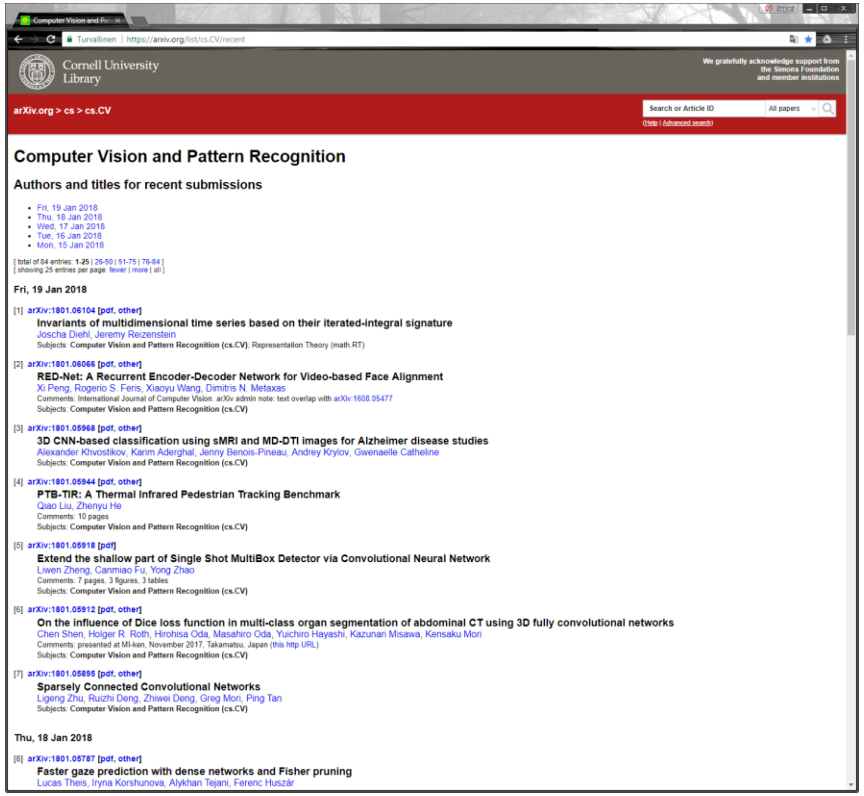
Project Plan



1. Methodology / Preparation of research infra
    a. Software platforms are constructed and tested
    b. Off-the-shelf models are acquired and tested
    c. Necessary skills on platforms are learned
2. Methodology / Method survey
    a. Current state-of-art methods are studied
    b. Methods are constructed and tested on the software platforms
3. Method follow-up
    a. Screening of conference papers related to the subject
    b. Possibly integrating new methods to the project

Work Done

Method Follow-Up



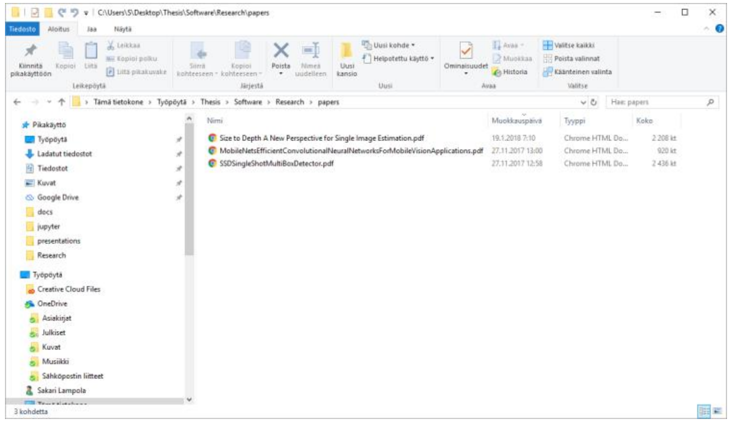


Image Object Velocity Estimation

Image object velocity is necessary for:

- predicting image object locations when matching new measurements
- identifying image objects
- predicting image object locations for hidden objects

Image object

- id
- status
- x_min
- x_max
- y_min
- y_max
- vx_min
- vx_max
- vy_min
- vy_max
- class
- confidence
- appearance

Estimation algorithm

Image Object Kalman Filtering

Bounding box corner location

State vector s:

$$s = \begin{bmatrix} l \\ v \end{bmatrix}$$

where

l = location coordinate (x_{min}, x_{max}, y_{min}, y_{max}) of the bounding box corner in the image
v = velocity (vx_{min}, vx_{max}, vy_{min}, vy_{max}) of the bounding box corner in the image

State equation in differential form:

$$\frac{ds(t)}{dt} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} * s(t) + \epsilon(t) = A_1 * s$$

State equation in difference form:

$$s(k+1) = (I + \Delta * A_1) * s(k) + \epsilon(k)$$

$$= \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} * s(k) + \varepsilon(k) = A * s(k) + \varepsilon(k)$$

where Δ is the time increment and ε Gaussian noise with co-variance R.

Measurement equation

$$z(k) = [1 \quad 0] * s(k) + \delta(k) = C * s(k) + \delta(k)$$

Where δ is Gaussian noise with covariance matrix Q.

Kalman filter initialization:

$$\mu(0) = \begin{bmatrix} l(0) \\ 0 \end{bmatrix}$$

where l(0) is the first location measurement.

$$\Sigma(0) = \begin{bmatrix} 10.0 & 0 \\ 0 & 10000.0 \end{bmatrix}$$

where 10.0 and 10000.0 are believed initial error variances of location and velocity.

$$R = \begin{bmatrix} 1.0 & 0 \\ 0 & 1.0 \end{bmatrix}$$

where diagonal elements are believed state equation vari-ances of location and velocity.

$$Q = [10.0]$$

Where 10.0 is the believed measurement variance.

Kalman filter update:

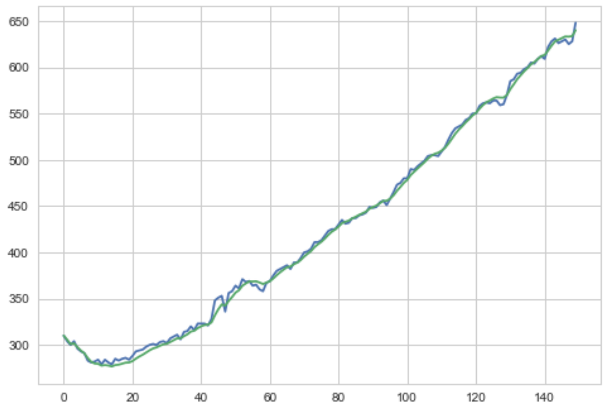
$$\mu_1(k) = A * \mu(k-1)$$
$$\Sigma_1(k) = A * \Sigma(k-1) * A^T + R$$
$$K(k) = \Sigma_1(k) * C^T * (C * \Sigma_1(k) * C^T + Q)^{-1}$$
$$\mu(k) = \mu_1(k) + K(k) * (z(k) - C * \mu_1(k))$$
$$\Sigma(k) = (I - K(k) * C) * \Sigma_1(k)$$

Asiakirjan loppu

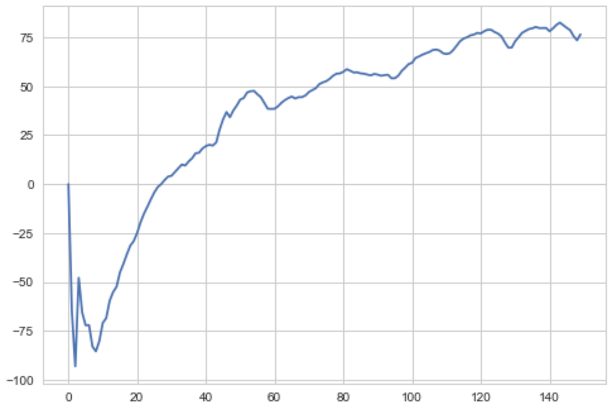
Numerical values are estimated using grid search and 10 step ahead mean prediction error. Values rounded. Later adjusted by experiments.

Image Object Velocity Estimation

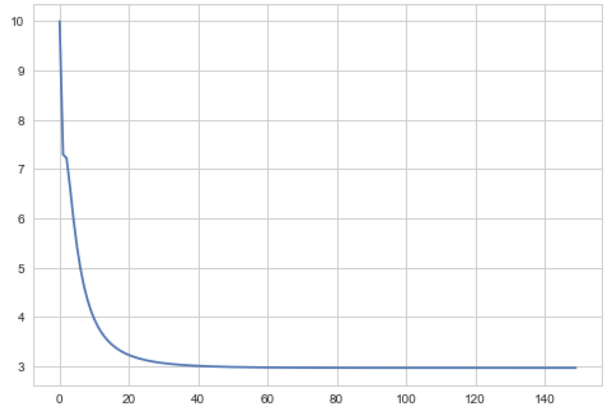
Moving object (car)



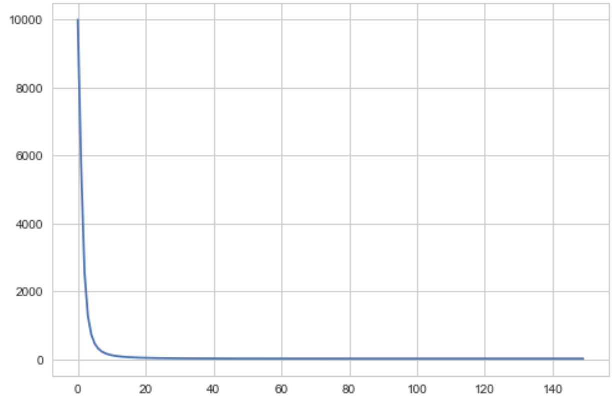
Measured and filtered location (upper left corner)



Estimated velocity



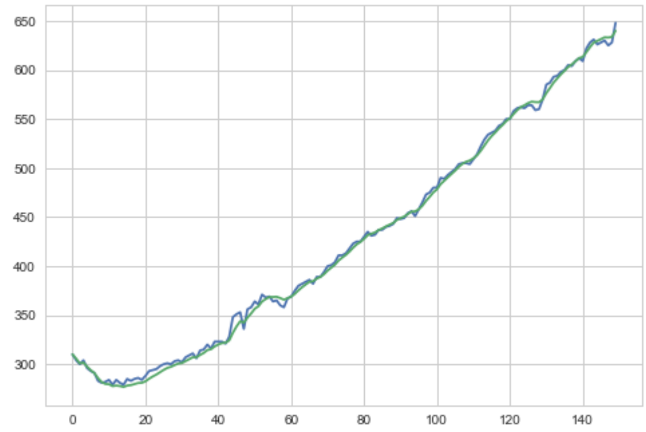
Location variance

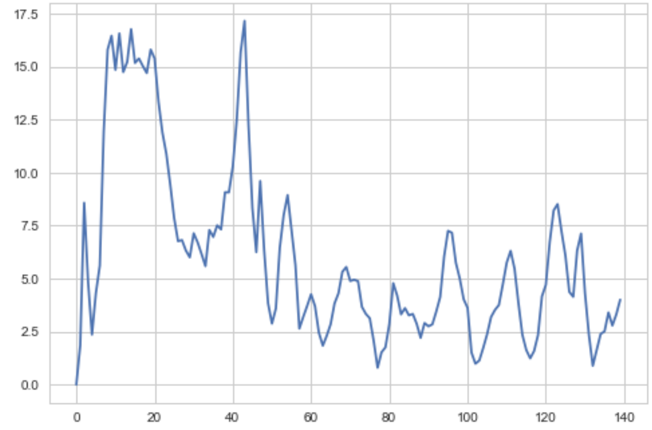


Velocity variance

Image Object Velocity Estimation

Moving object (car)

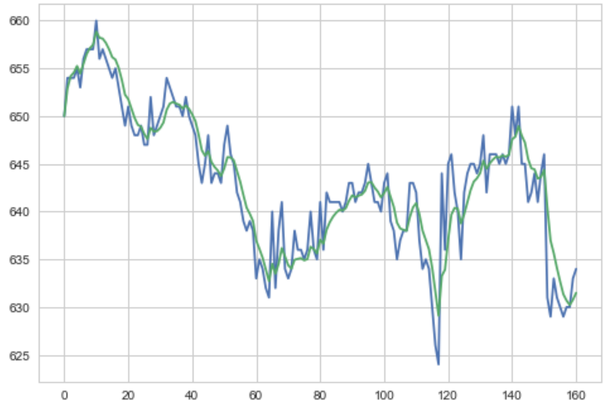




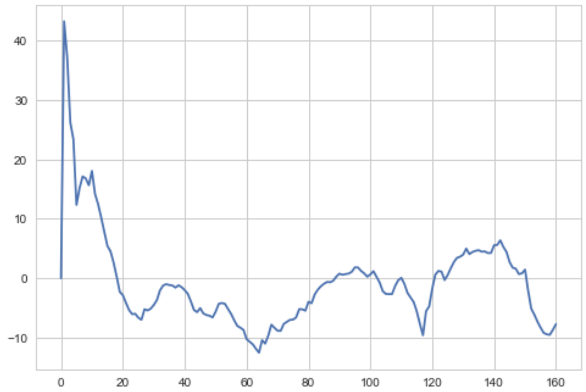
10 step ahead mean prediction error

Image Object Velocity Estimation

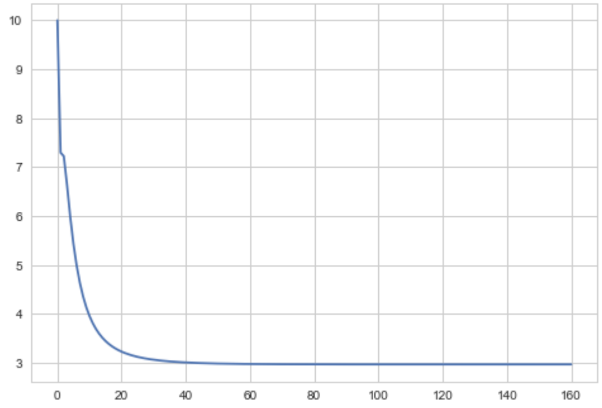
Static object (calf)



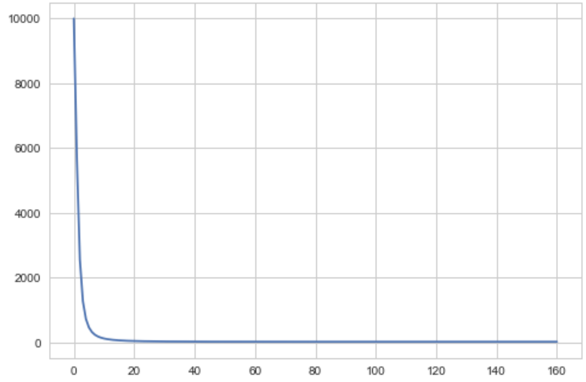
Measured and filtered location (upper left corner)



Estimated velocity



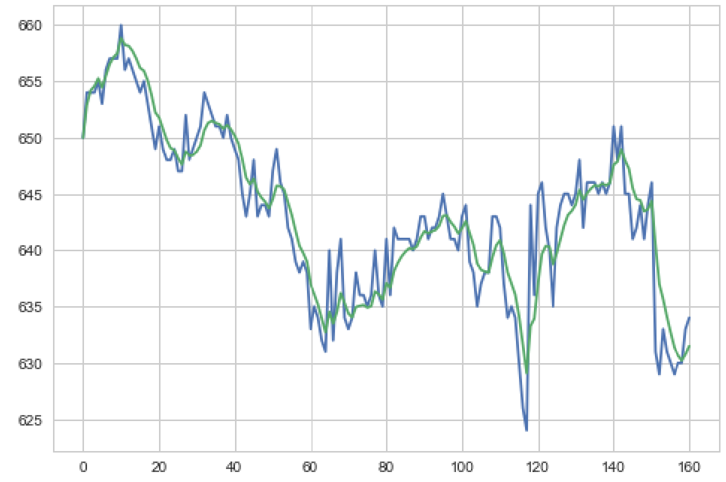
Location variance

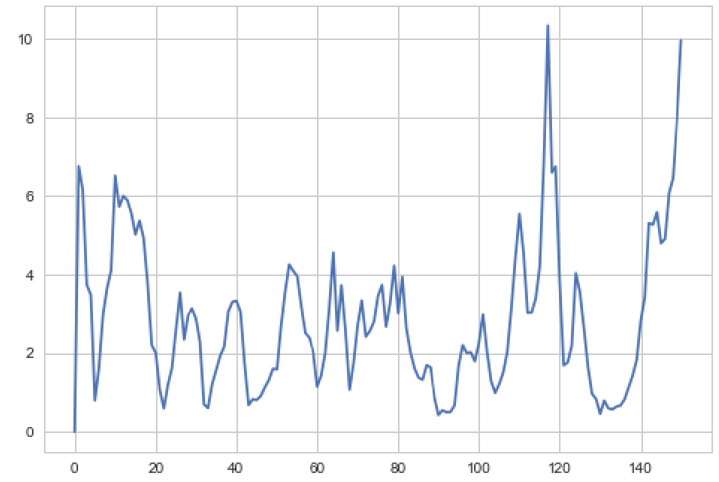


Velocity variance

Image Object Velocity Estimation

Static object (calf)

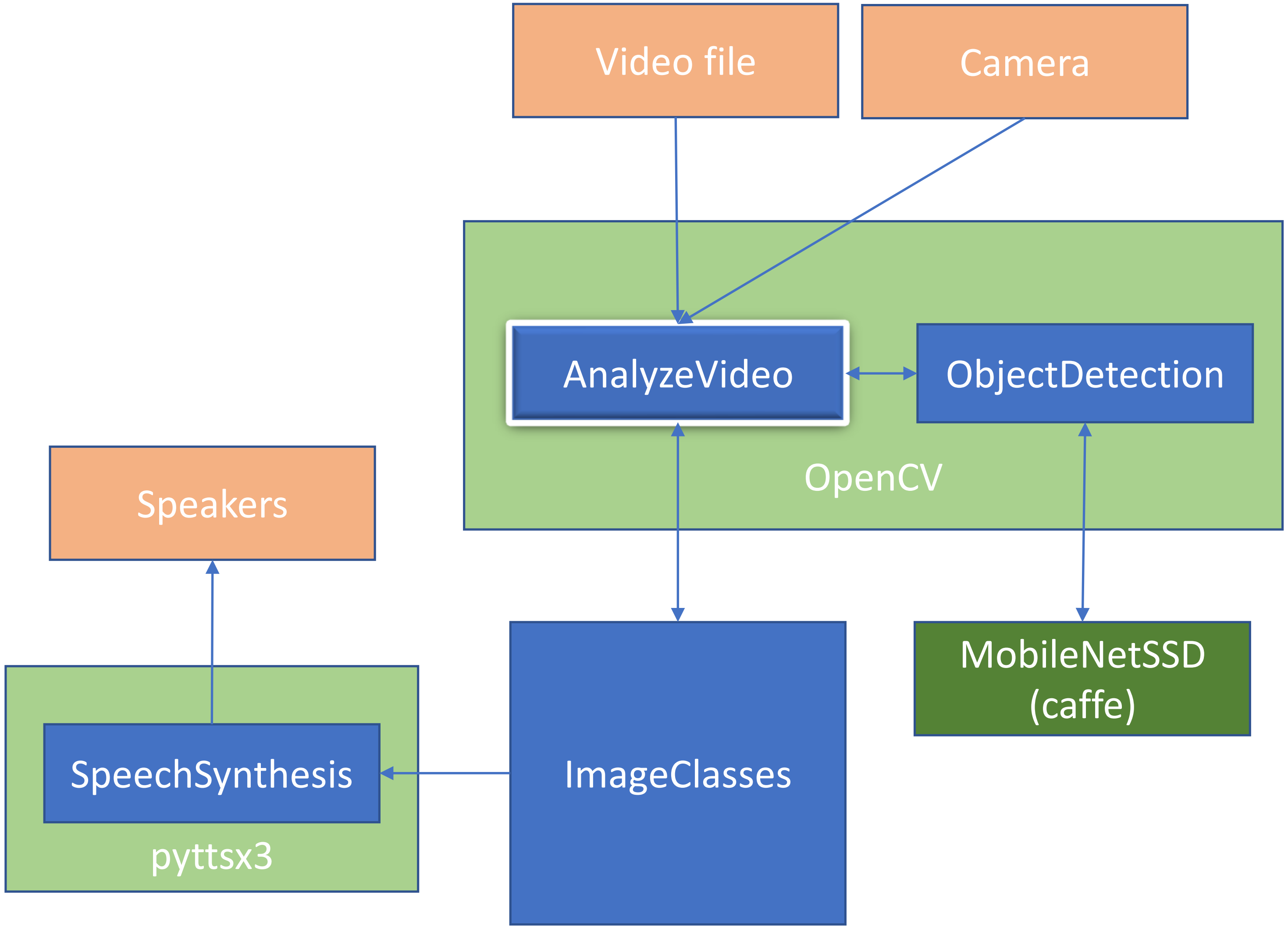




10 step ahead mean prediction error

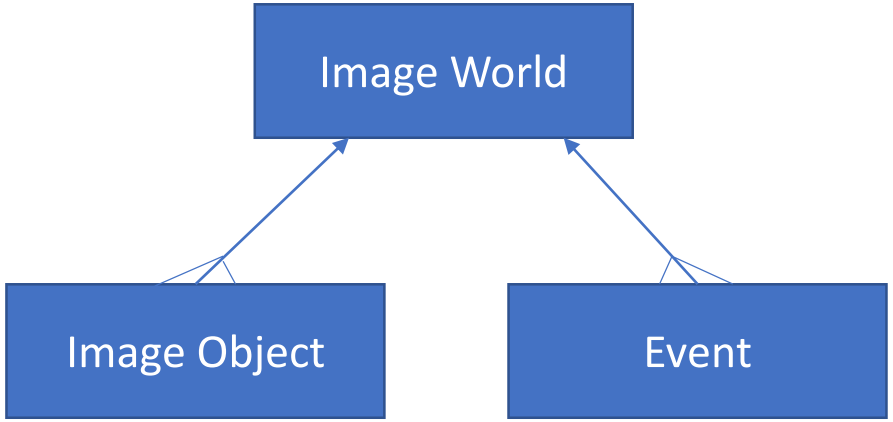
Speech Synthesis

Software Architecture



Speech Synthesis

Entities



Event
- time
- text
- priority

- Event is generated when
  - new image object is created
  - image object status is changed
- Event will pause the video for the duration of speech (not in the final version)
- Events are collected (history)

Confidence Level

SSD Mobilenet implementation:
extract the confidence (i.e., probability) associated with the prediction

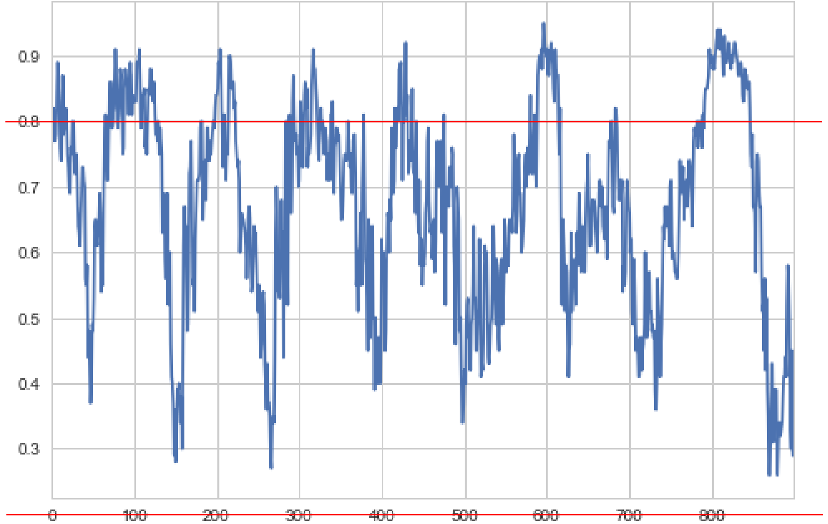
Objects detected		Confidence level							
Video	Correct	0,00	0,20	0,40	0,60	0,80	0,90	0,95	1,00
CarsOnHighway001.mpg	39	49	49	39	36	34	32	32	0
Calf-2679.mp4	1	2	2	2	2	1	1	1	0
Dunes-7238.mp4	1	7	7	6	5	2	2	2	0
Sofa-11294.mp4	1	2	2	1	1	1	1	1	0
Cars133.mp4	5	9	9	6	5	5	5	5	0
BlueTit2975.mp4	1	3	3	2	1	1	1	1	0
Railway-4106.mp4	1	10	10	5	3	3	1	1	0
Hiker1010.mp4	1	4	4	0	0	0	0	0	0
Cat-3740.mp4	1	3	3	2	2	1	1	1	0
SailingBoat6415.mp4	1	1	1	1	1	1	1	1	0
AWomanStandsOnTheSeashore-10058.mp4	1	1	1	1	1	1	1	1	0
Dog-4028.mp4	1	4	4	2	1	1	1	1	0
Boat-10876.mp4	1	2	2	1	1	1	1	0	0
Horse-2980.mp4	1	3	3	3	2	2	1	1	0
Sheep-12727.mp4	1	1	1	1	1	1	1	1	1

Good value for creating a new image object is between 0.8 and 0.9.

The 'good' value also depends on other hyperparameters.

Confidence Level

Confidence level has dynamics



create

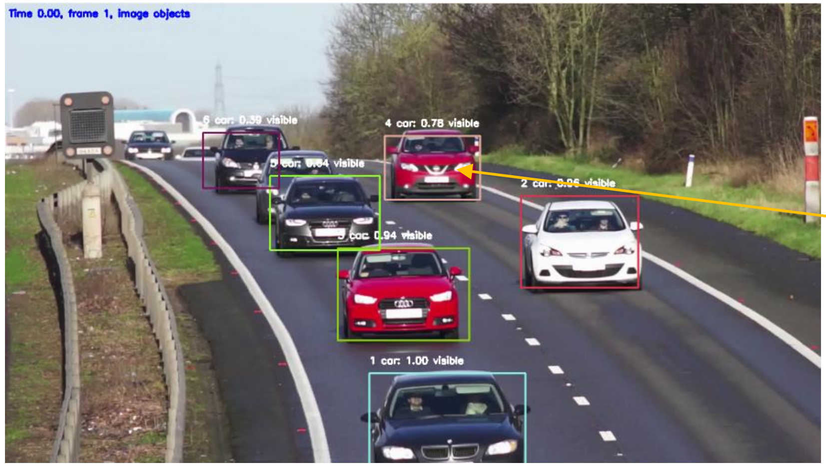
Update (not class)

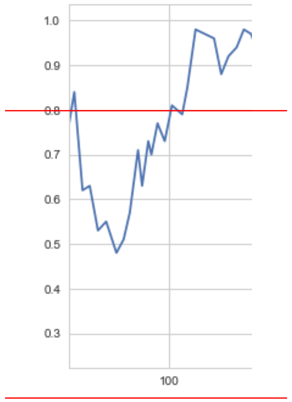
ignore

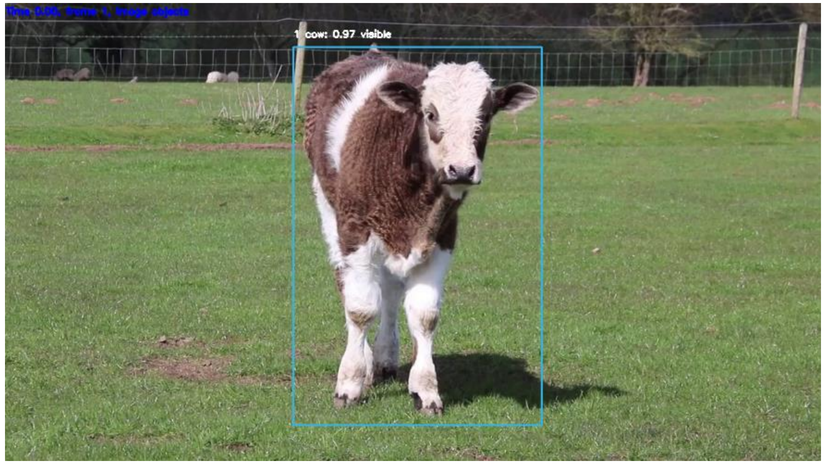
Different levels for creating and updating image object. Hyperparameters:

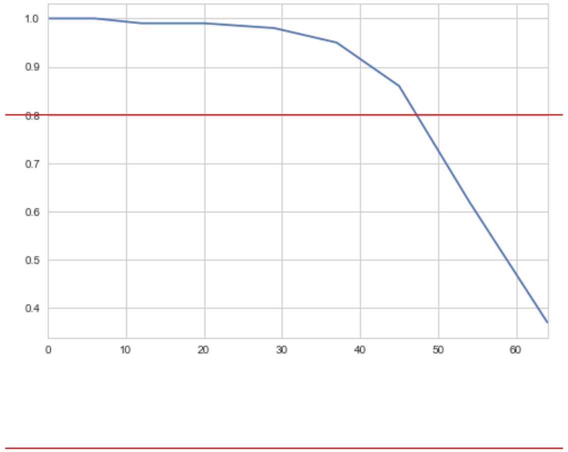
- CONFIDENCE_LEVEL_CREATE (0.8)
- CONFIDENCE_LEVEL_UPDATE (0.2)

Confidence Level

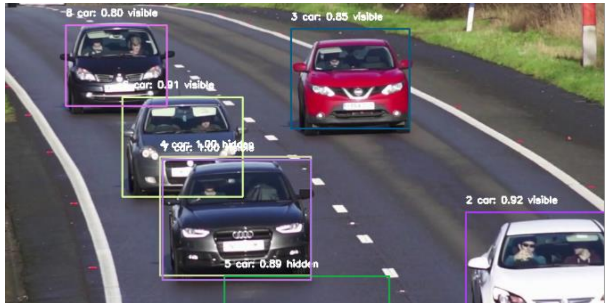








Border Behaviour



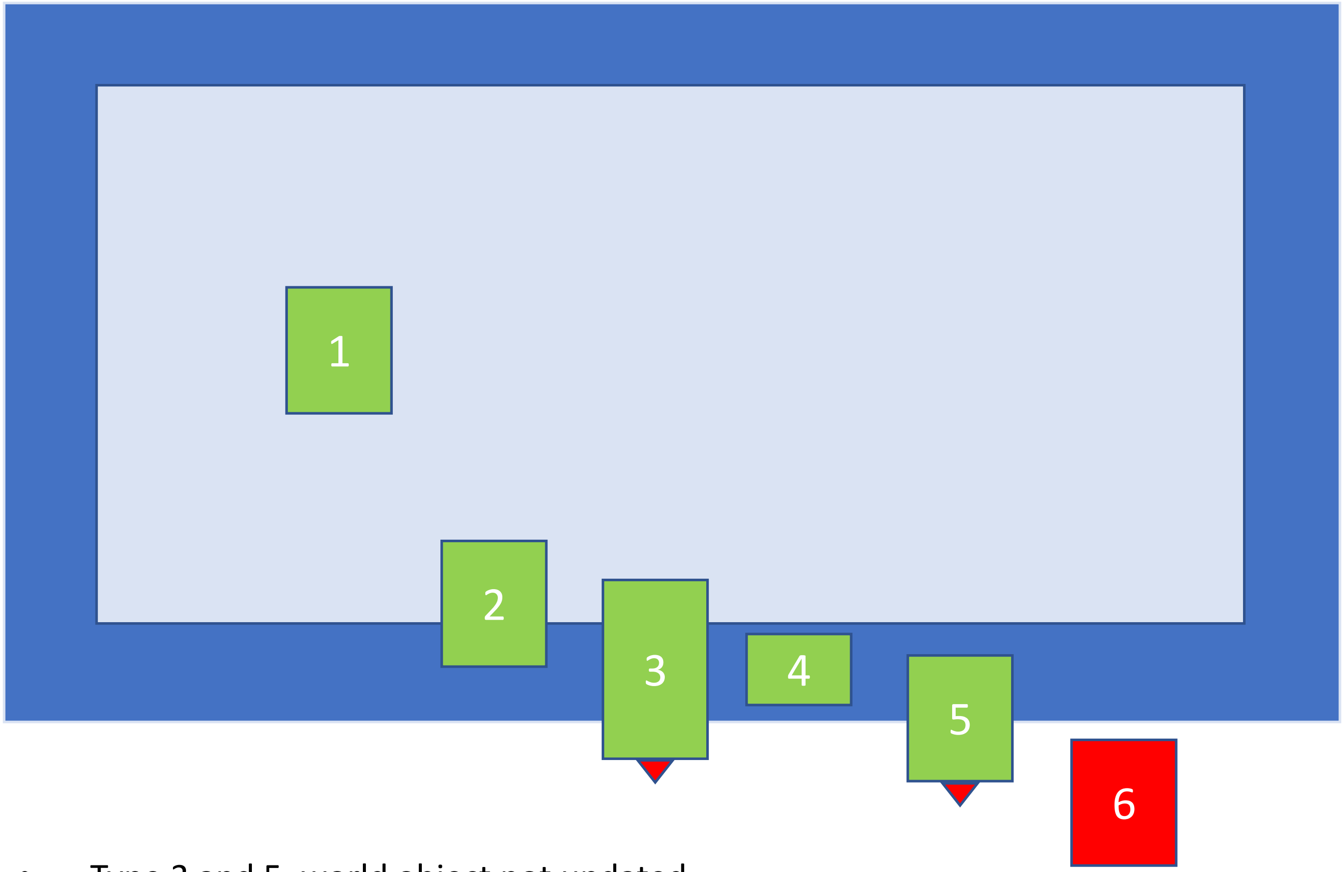
Box size and form distorded

time	x_max_c	x_max_m	x_max_p	y_max_c	y_max_m	y_max_p
1.48	1208.859	1209.0	1205.616	646.300	652.0	640.731
1.52	1221.500	1236.0	1212.044	653.697	656.0	649.501
1.56	1232.488	1242.0	1224.941	660.427	661.0	656.939
1.60	1241.599	1246.0	1236.095	668.758	673.0	663.679
1.64	1251.081	1256.0	1245.282	677.391	682.0	672.083
1.68	1258.430	1258.0	1254.848	687.143	694.0	680.794
1.72	1265.965	1266.0	1262.190	694.428	695.0	690.663
1.76	1272.740	1271.0	1269.725	704.340	711.0	697.956
1.80	1280.741	1282.0	1276.471	711.433	711.0	707.979
1.84	1287.573	1286.0	1284.493	717.291	714.0	715.066
1.88	1292.323	1286.0	1291.299	722.517	718.0	720.869
1.92	1292.517	1276.0	1295.946	728.172	725.0	726.022
1.96	1291.385	1273.0	1295.873	731.168	722.0	731.626
2.00	1291.974	1279.0	1294.445	732.465	720.0	734.474
2.04	1291.500	1277.0	1294.826	732.500	718.0	735.572
2.08	1290.547	1276.0	1294.121	733.994	724.0	735.375
2.12	1289.259	1275.0	1292.938	736.016	728.0	736.711
2.16	1289.533	1280.0	1291.424	736.959	727.0	738.606
2.20	1290.113	1282.0	1291.548	737.402	727.0	739.392
2.24	1290.640	1283.0	1292.000	735.994	722.0	739.671

```
In [10]: # image size 1280 * 720
```

Hyperparameter BORDER_WIDTH (30)

Border Behaviour

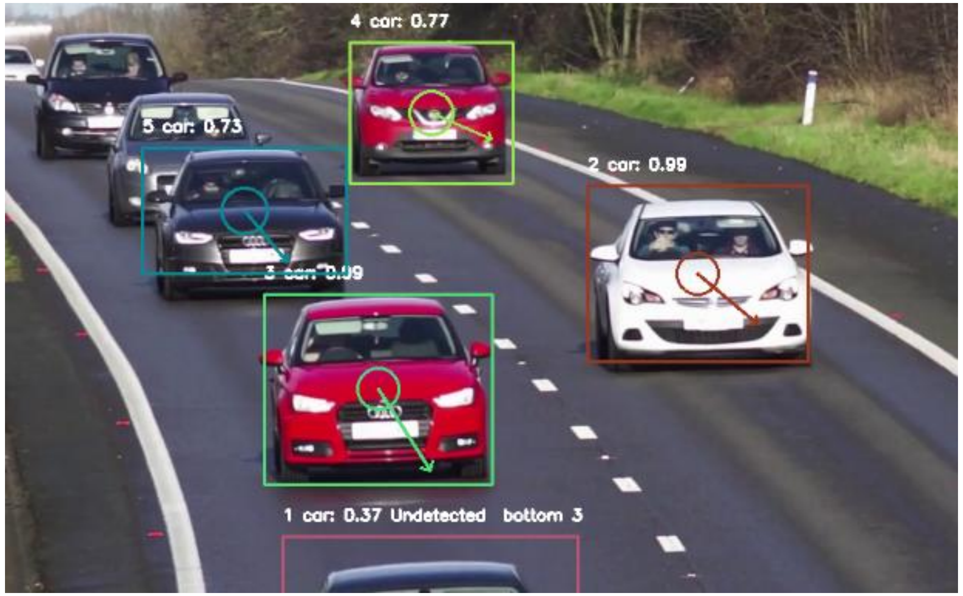


- Type 3 and 5: world object not updated
- Type 6: removed, world object acceleration fixed
- If an object touches 3 borders, it is removed

Done for:

- left
- right
- top
- bottom

Visual Presentation



- Ellipse axes proportional to the standard deviation of the location (2*std, corresponding to 95% probability)
- Arrow direction and length proportional to velocity (measured in pixels/second)

Object Retention

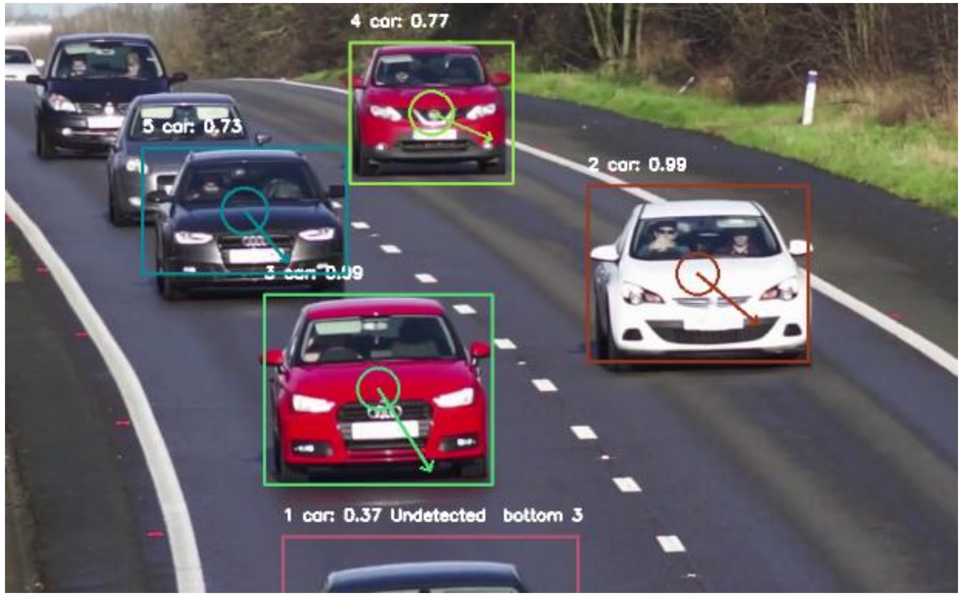
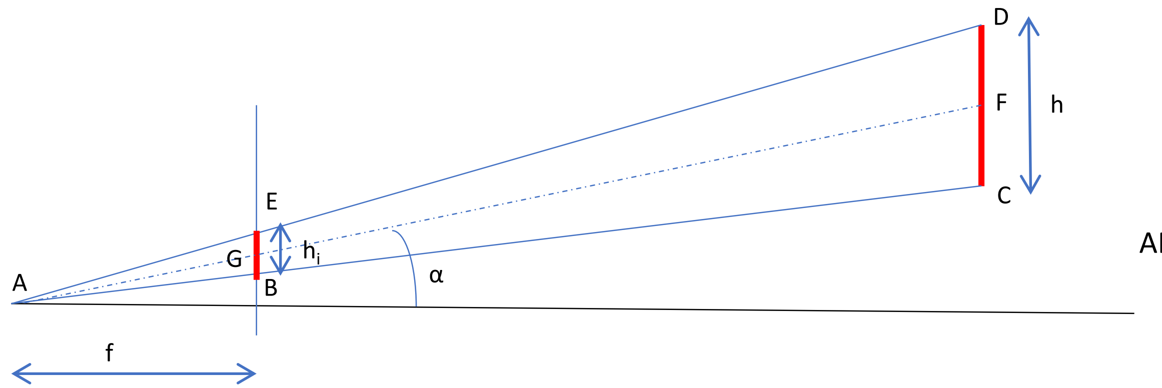


Image objects are removed if not detected in RETENTION_COUNT_MAX (30) successive frames.

Distance Estimation



AF=d (distance)

Similar triangles AGE and AFD:

$$\frac{0.5 * h_i}{0.5 * h} = \frac{AG}{d} = \frac{\frac{f}{\cos(\alpha)}}{d} = \frac{f}{d * \cos(\alpha)}$$

$$\boxed{d = \frac{f * h}{\cos(\alpha) * h_i}}$$

Simalar equations for horizontal direction (β=azimuth)

Distance Estimation

$$d = \frac{f * h}{\cos(\alpha) * \cos(\beta) * h_i} = \frac{f * h}{\cos(\alpha) * \cos(\beta) * h_i * s_h / p_h}$$

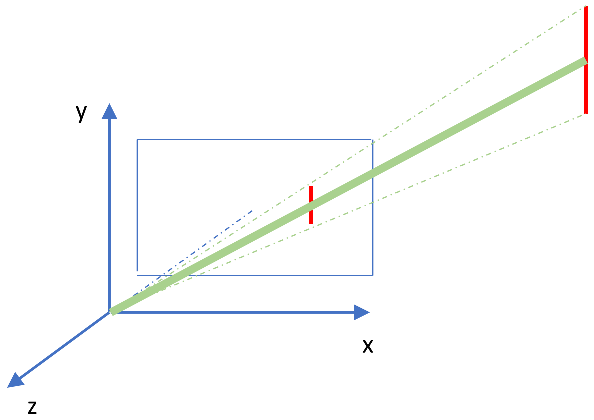
s_w= sensor width (m)
s_h= sensor height (m)
p_w= image width (pixels)
p_h= image height (pixels)
h_i = object height (pixels)
h = object height (m)
f = focal length (m)
α = altitude (rad)
β = azimuth (rad)

Example (Nikon D800E):

s_w= sensor width (m) = 0.0359 m
s_h= sensor height (m) = 0.0240 m
p_w= image width (pixels) = 7360
p_h= image height (pixels) = 4912
h_i = object height (pixels) = 100
h = object height (m) = 1.0 m
f = focal length (m) = 0.050 m
α = *altitude (rad) = 0.0*
β = *azimuth (rad) = 0.0*

$$d = \frac{0.050m * 1m}{1.0 * 1.0 * 100 * 0.024m/4912} = 102.33\ m$$

2d -> 3d Transformation



From pixel coordinates (sensor plane) to 3d camera coordinates:

$$(x_c, y_c, z_c) = \left(-\frac{s_w}{2} + xp*\frac{s_w}{p_w},\ \frac{s_h}{2} - yp*\frac{s_h}{p_h},\ -f\right)$$

Object center will be on the line:

$$(x_o, y_o, z_o) = t*(x_c, y_c, z_c)$$

The length of the line is:

$$d = \frac{f * h}{\cos(\alpha) * \cos(\beta) * h_i * s_h/p_h}$$

$$\alpha = \arctan(y_c/f)$$
$$\beta = \arctan(x_c/f)$$

s_w = *sensor width (m)*
s_h = *sensor height (m)*
p_w = *image width (pixels)*
p_h = *image height (pixels)*
h_i = *object height (pixels)*
h = *object height (m)*
f = *focal length (m)*
α = *altitude (rad)*

2d -> 3d Transformation

$$t^2 * ({x_c}^2 + {y_c}^2 + {z_c}^2) = d^2$$

$$t = \frac{d}{\sqrt{{x_c}^2 + {y_c}^2 + {z_c}^2}}$$

2d -> 3d Transformation

Example:

- s_w = sensor width (m) = 0.0359 m
- s_h = sensor height (m) = 0.0240 m
- p_w = image width (pixels) = 7360
- p_h = image height (pixels) = 4912
- h_i = object height (pixels) = 100
- h = object height (m) = 1.0 m
- f = focal length (m) = 0.050 m
- x_p = 1200
- y_p = 2000

$$(x_c, y_c, z_c) = \left(-\frac{s_w}{2} + xp * \frac{s_w}{p_w},\ \frac{s_h}{2} - yp * \frac{s_h}{p_h},\ -f\right)$$

$$= \left(-\frac{0.0359}{2} + 1200 * \frac{0.0359}{7360},\ \frac{0.0240}{2} - yp * \frac{0.0240}{4912},\ -0.050\right) = (-0.0121, 0.0022, -0.0500)$$

$\alpha = \arctan(y_c/f) = 0.0445$ $\quad\quad$ $\beta = \arctan(x_c/f) = -0.2374$

$$d = \frac{f * h}{\cos(\alpha) * \cos(\beta) * h_i * s_h/p_h}$$

$$= \frac{0.050 * 1}{\cos(0.0445) * \cos(-0.2374) * 100 * 0.0240/4912} = 105.39$$

$$t = \frac{105.39}{\sqrt{-0.0121^2 + 0.0022^2 + -0.0500^2}} = 2.0468e+03$$

2d -> 3d Transformation

Object location in 3d camera coordinates:

$(x_o, y_o, z_o) = t^* (x_c, y_c, z_c)$

$= 2.0468e{+}03 * (-0.0121, 0.0022, -0.0500)$

$(-24.7593, 4.5602, -102.3389)$

Open questions:

- Derivation ok?
- Assumptions ok?
  - Optical axis in sensor center?

2d -> 3d Transformation

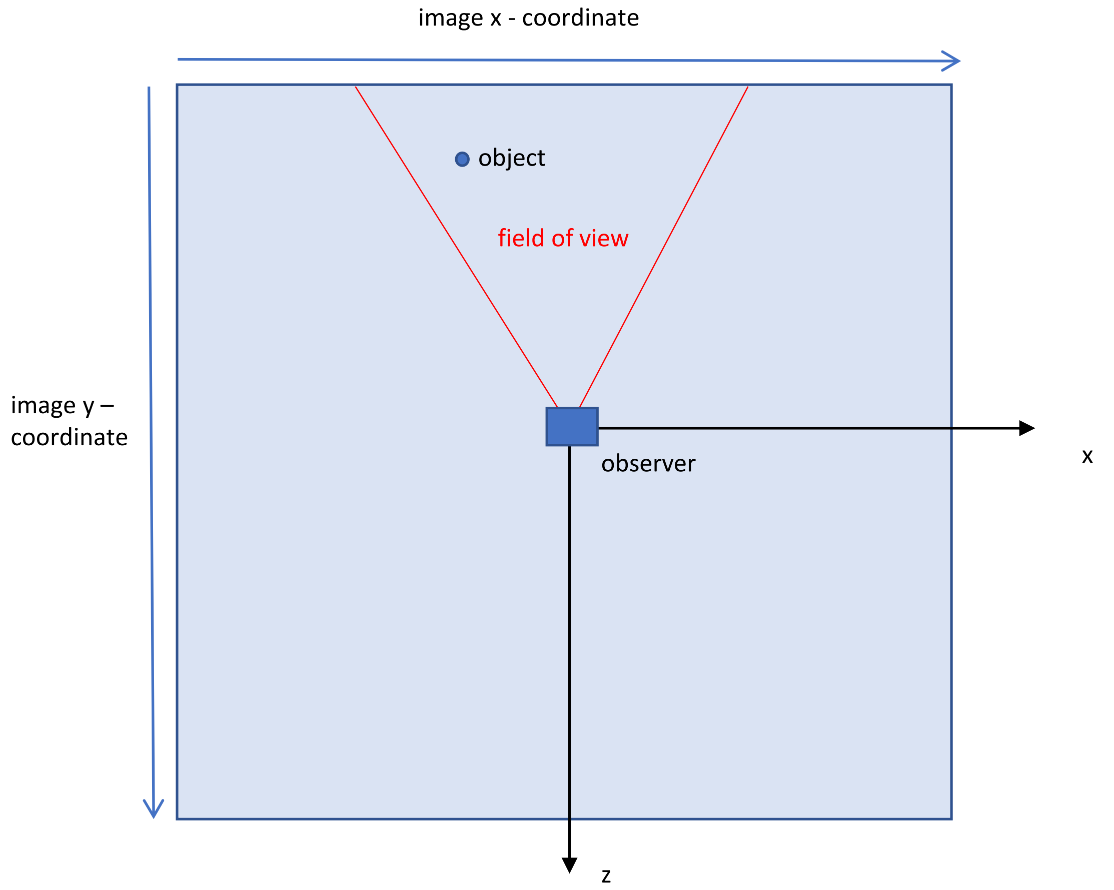
Parameters:

s_w= sensor width (m)
s_h= sensor height (m)
p_w= image width (pixels)
p_h= image height (pixels)
f = focal length (m)

Open questions:

- Video metadata often lacks sensor and focal parameters
- Focal length can change during shooting (zooming)

Map Presentation



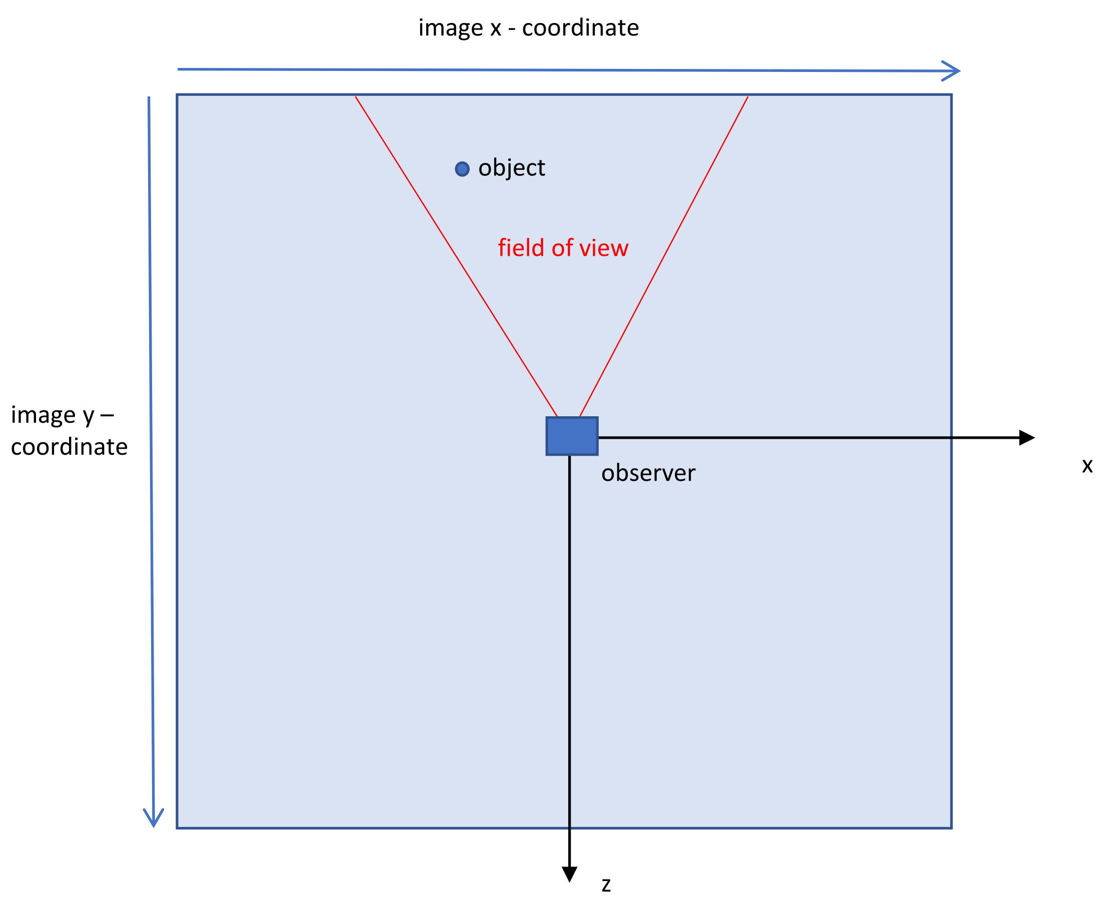
x_w= *object world x coordinate (m)*
z_w= *object world z coordinate(m)*
x_i= *object image x coordinate (pixel)*
y_i= *object image y coordinate (pixel)*
h_w= *image area world height (m)*
h_i = *image area image height (pixels)*
w_w= *image area world width (m)*
w_i = *image area image width (pixels)*

$$(x_i, y_i) = \left(\frac{w_i}{2} + x_w * \frac{w_i}{w_w}, \quad \frac{h_i}{2} + zw * \frac{h_i}{h_w} \right)$$

$$\frac{w_i}{w_w} = \frac{h_i}{h_w} = \text{p (pixel/meter ratio)}$$

$$(x_i, y_i) = \left(\frac{w_i}{2} + x_w * p, \quad \frac{h_i}{2} + zw * \text{p} \right)$$

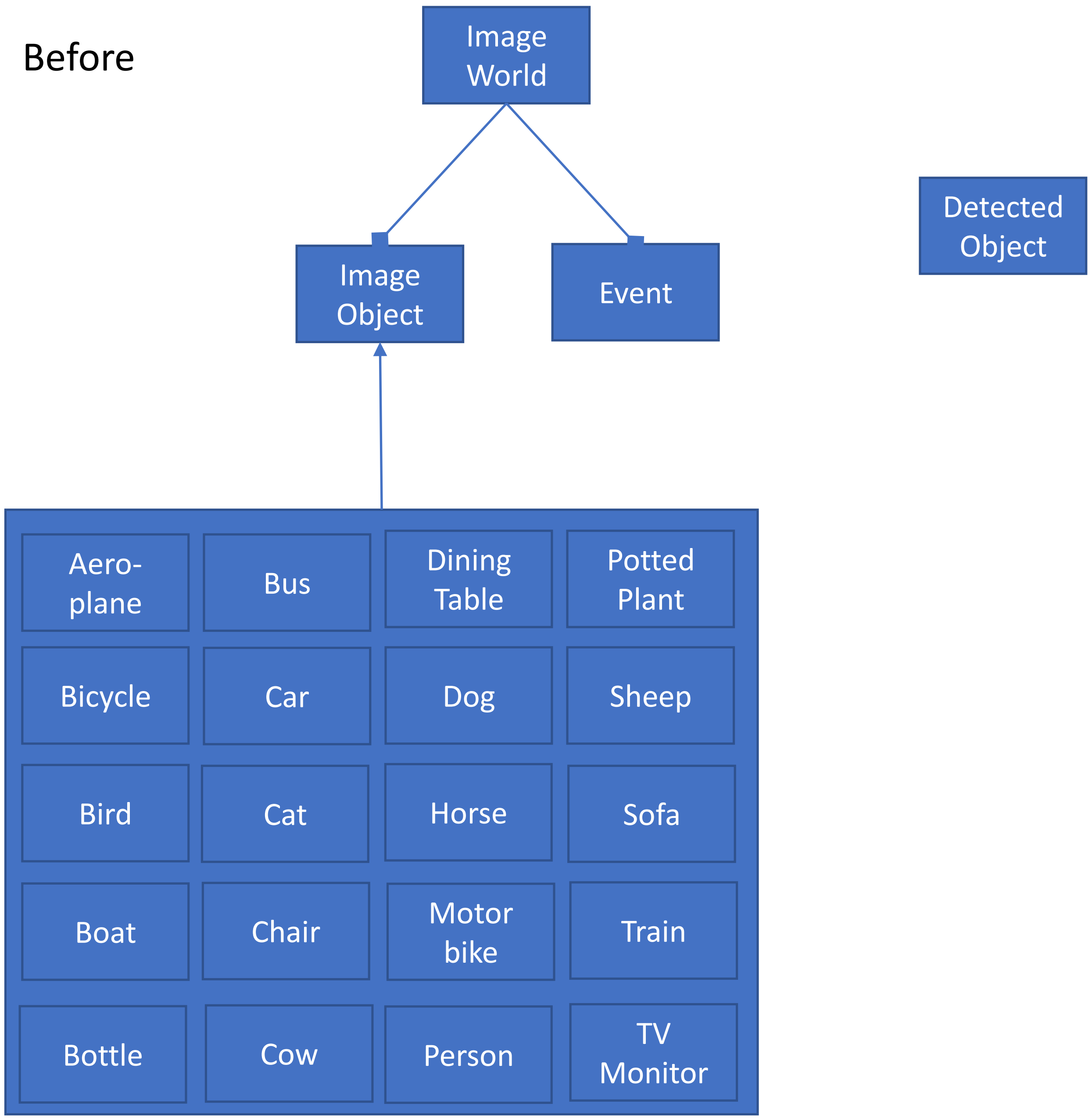
Map Presentation



$$FOV = 2 * \text{atan}(\frac{s_w}{2 * f})$$

$s_w = sensor\ width\ (m)$
f = focal length (m)

Entity Diagram



Entity Diagram

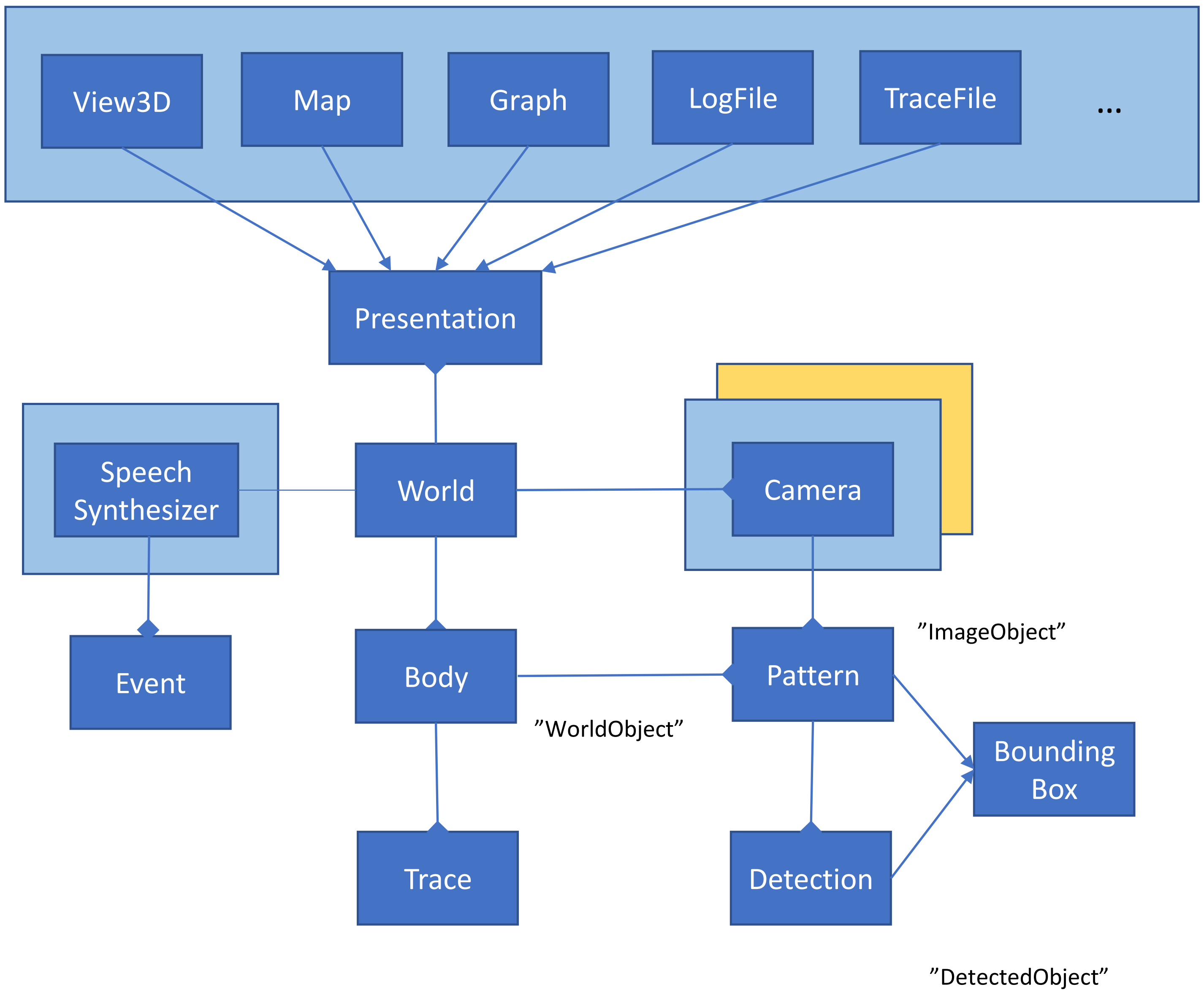
V2.0 goal

- Detected classes not hardcoded
- Object class may change
- Support for many cameras , rotations
- Names less awkward
- Cleaning
- Python style guide followed, excluding line length
- Code optimization
- One package

Name of the software package: ShadowWorld
(Plato: Allegory of the Cave)

Entity Diagram

V2.0



Entities

Camera
----
id
world
patterns
image_width
image_height
focal_length
sensor_width
sensor_height
field_of_view
x
y
z
yaw
pitch
roll

__init__

Pattern
----
id
camera
detections
class_id
x_min
x_max
y_min
y_max
vx_min
vx_max
vy_min
vy_max
sigma_x_min
sigma_x_max
sigma_y_min
sigma_x_max
appearance
confidence
bounding_box_color
border_left
border_right
border_top
border_bottom
retention_count
matched

__init__
border_count
center_point
center_point_velocity
correct
Is_center_reliable
Is_vanished
location_variance
predict
set_border_behaviour

Detection
----
id
pattern
time
class_id
x_min
x_max
y_min
y_max
appearance
confidence
matched

__init__
pattern_distance
pattern_distance_with_class

Entities

World
----
bodies
cameras
speech_synthesizer

__init__
update

SpeechSynthesizer
----
engine

__init__
say

Body
----
id
world
events
traces
x
y
z
vx
vy
vz
ax
ay
az
height_min
height_mean
height_max
width_min
width_mean
width_max
length_min
length_mean
length_max
velocity_max
acceleration_max

__init__
border_count
center_point
center_point_velocity
correct
Is_center_reliable
Is_vanished
location_variance
predict
set_border_behaviour

Event
----
body1
body2
time
category
priority

__init__

Trace
----
body
time
x
Y
z

__init__

Work in Progress

Perception

"The first step in achieving SA is to perceive the status, attributes, and dynamics of relevant elements in the environment. Thus, Level 1 SA, the most basic level of SA, involves the processes of monitoring, cue detection, and simple recognition, which lead to an awareness of multiple situational elements (objects, events, people, systems, environmental factors) and their current states (locations, conditions, modes, actions)."

Next Steps

Next steps

Comprehension:

1. Closing the open questions
2. 2d -> 3d transformation
3. World object state estimation

To Be Discussed

To Be Discussed

- Activity recognition?
- Emotion recognition?
- Turning camera, estimation by background movement?

Thank you!

lampola@student.tut.fi
https://github.com/SakariLampola/Thesis