Image-Based Situation Awareness Audit 28.2.2018

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 algorithm, special case for hidden objects 🟡

To do:

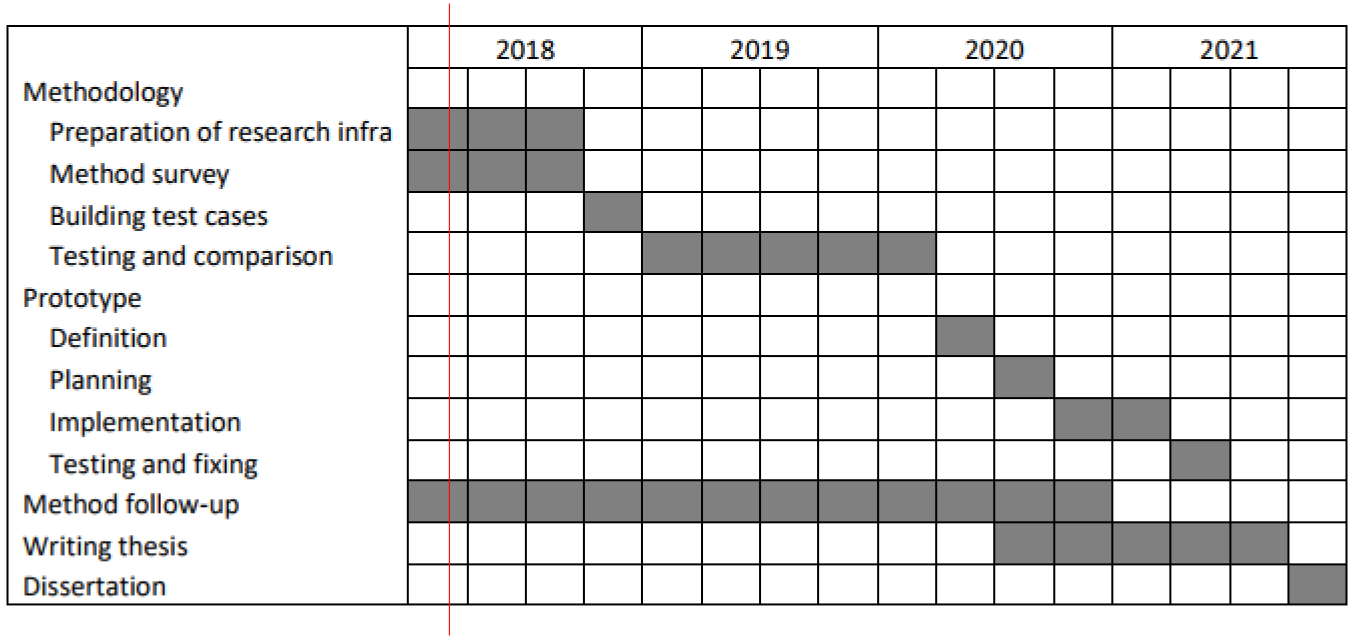
- Close open questions 🟡
- Image object status 🟢
- Image object velocity estimation 🟢
- Probabilistic approach for matching detected and image objects 🟡
- 2d -> 3d transformation 🟢
- World object state estimation 🟢

Other:

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

Project Plan

Project Plan

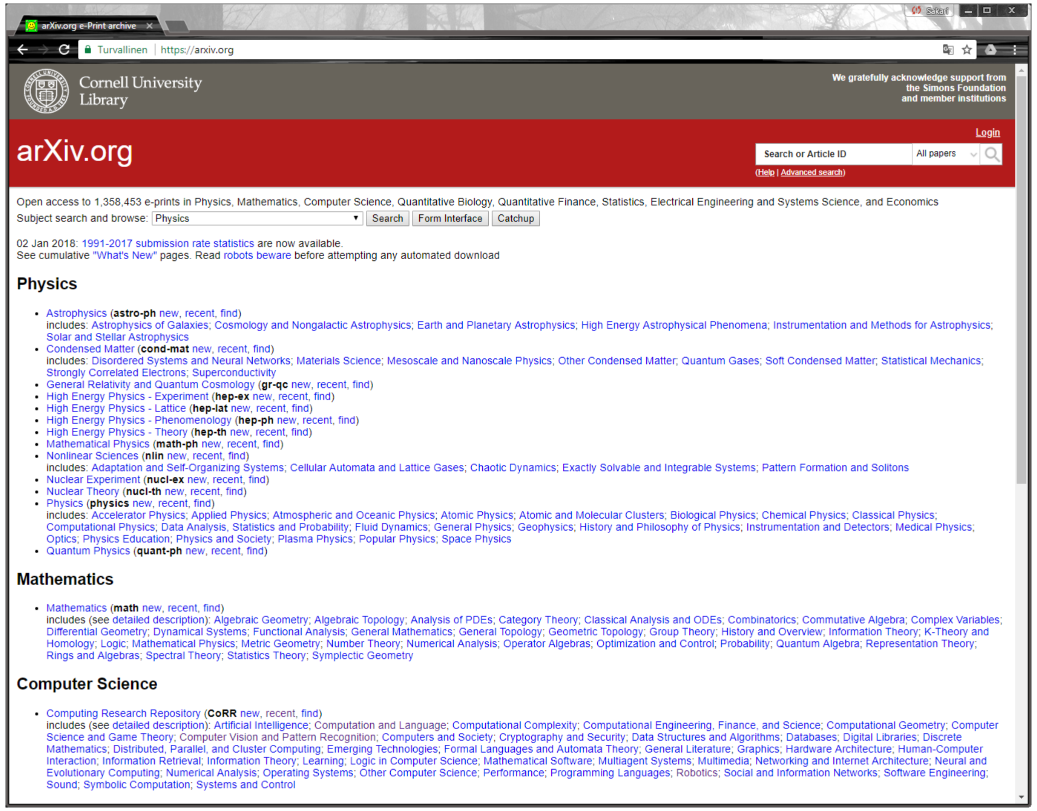


	2018				2019				2020				2021			
Methodology																
Preparation of research infra	▨	▨	▨													
Method survey	▨	▨	▨													
Building test cases				▨												
Testing and comparison					▨	▨	▨	▨	▨							
Prototype																
Definition										▨						
Planning											▨					
Implementation												▨	▨			
Testing and fixing														▨		
Method follow-up	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				
Writing thesis											▨	▨	▨	▨	▨	
Dissertation																▨

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



Cornell University Library

We gratefully acknowledge support from the Simons Foundation and member institutions

arXiv.org

Open access to 1,358,453 e-prints in Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance, Statistics, Electrical Engineering and Systems Science, and Economics

Subject search and browse: Physics

02 Jan 2018: 1991-2017 submission rate statistics are now available.
See cumulative "What's New" pages. Read robots beware before attempting any automated download

Physics

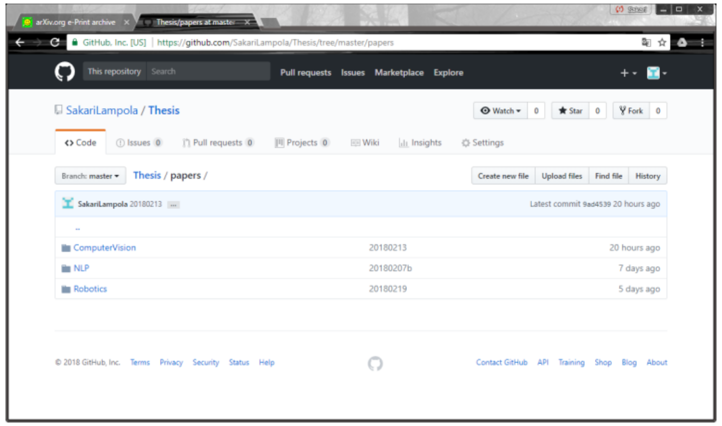
- Astrophysics (**astro-ph** new, recent, find)
  includes: Astrophysics of Galaxies; Cosmology and Nongalactic Astrophysics; Earth and Planetary Astrophysics; High Energy Astrophysical Phenomena; Instrumentation and Methods for Astrophysics; Solar and Stellar Astrophysics
- Condensed Matter (**cond-mat** new, recent, find)
  includes: Disordered Systems and Neural Networks; Materials Science; Mesoscale and Nanoscale Physics; Other Condensed Matter; Quantum Gases; Soft Condensed Matter; Statistical Mechanics; Strongly Correlated Electrons; Superconductivity
- General Relativity and Quantum Cosmology (**gr-qc** new, recent, find)
- High Energy Physics - Experiment (**hep-ex** new, recent, find)
- High Energy Physics - Lattice (**hep-lat** new, recent, find)
- High Energy Physics - Phenomenology (**hep-ph** new, recent, find)
- High Energy Physics - Theory (**hep-th** new, recent, find)
- Mathematical Physics (**math-ph** new, recent, find)
- Nonlinear Sciences (**nlin** new, recent, find)
  includes: Adaptation and Self-Organizing Systems; Cellular Automata and Lattice Gases; Chaotic Dynamics; Exactly Solvable and Integrable Systems; Pattern Formation and Solitons
- Nuclear Experiment (**nucl-ex** new, recent, find)
- Nuclear Theory (**nucl-th** new, recent, find)
- Physics (**physics** new, recent, find)
  includes: Accelerator Physics; Applied Physics; Atmospheric and Oceanic Physics; Atomic Physics; Atomic and Molecular Clusters; Biological Physics; Chemical Physics; Classical Physics; Computational Physics; Data Analysis, Statistics and Probability; Fluid Dynamics; General Physics; Geophysics; History and Philosophy of Physics; Instrumentation and Detectors; Medical Physics; Optics; Physics Education; Physics and Society; Plasma Physics; Popular Physics; Space Physics
- Quantum Physics (**quant-ph** new, recent, find)

Mathematics

- Mathematics (**math** new, recent, find)
  includes (see detailed description): Algebraic Geometry; Algebraic Topology; Analysis of PDEs; Category Theory; Classical Analysis and ODEs; Combinatorics; Commutative Algebra; Complex Variables; Differential Geometry; Dynamical Systems; Functional Analysis; General Mathematics; General Topology; Geometric Topology; Group Theory; History and Overview; Information Theory; K-Theory and Homology; Logic; Mathematical Physics; Metric Geometry; Number Theory; Numerical Analysis; Operator Algebras; Optimization and Control; Probability; Quantum Algebra; Representation Theory; Rings and Algebras; Spectral Theory; Statistics Theory; Symplectic Geometry

Computer Science

- Computing Research Repository (**CoRR** new, recent, find)
  includes (see detailed description): Artificial Intelligence; Computation and Language; Computational Complexity; Computational Engineering, Finance, and Science; Computational Geometry; Computer Science and Game Theory; Computer Vision and Pattern Recognition; Computers and Society; Cryptography and Security; Data Structures and Algorithms; Databases; Digital Libraries; Discrete Mathematics; Distributed, Parallel, and Cluster Computing; Emerging Technologies; Formal Languages and Automata Theory; General Literature; Graphics; Hardware Architecture; Human-Computer Interaction; Information Retrieval; Information Theory; Learning; Logic in Computer Science; Mathematical Software; Multiagent Systems; Multimedia; Networking and Internet Architecture; Neural and Evolutionary Computing; Numerical Analysis; Operating Systems; Other Computer Science; Performance; Programming Languages; Robotics; Social and Information Networks; Software Engineering; Sound; Symbolic Computation; Systems and Control



https://github.com/SakariLampola/Thesis/tree/master/papers

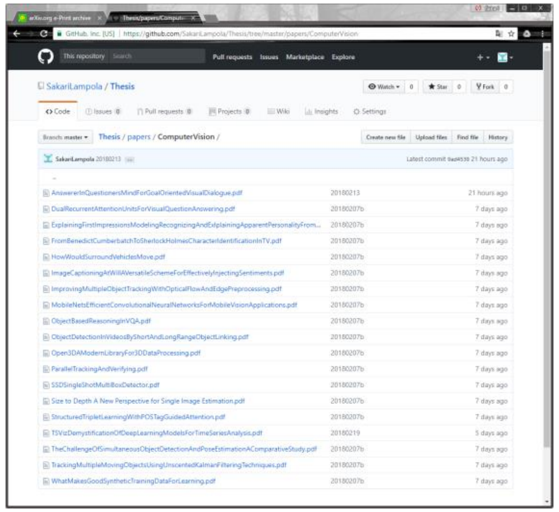
SakariLampola / Thesis

Thesis / papers /

SakariLampola 20180213 — Latest commit 9ad4539 20 hours ago

ComputerVision	20180213	20 hours ago
NLP	20180207b	7 days ago
Robotics	20180219	5 days ago

© 2018 GitHub, Inc.



https://github.com/SakariLampola/Thesis/tree/master/papers/ComputerVision

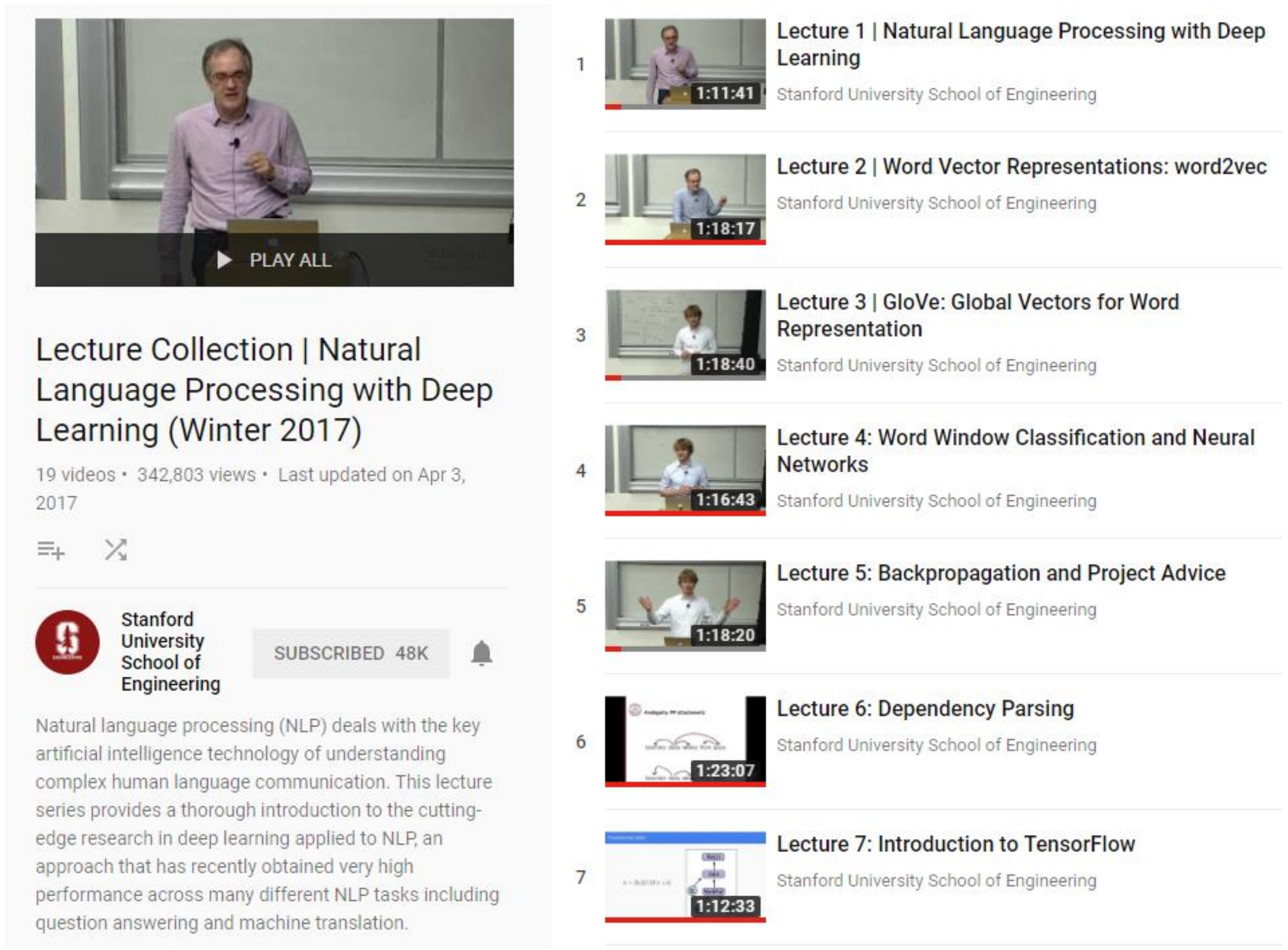
SakariLampola / Thesis

Thesis / papers / ComputerVision /

SakariLampola 20180213 — Latest commit 9ad4539 21 hours ago

AnswererInQuestionersMindForGoalOrientedVisualDialogue.pdf	20180213	21 hours ago
DualRecurrentAttentionUnitsForVisualQuestionAnswering.pdf	20180207b	7 days ago
ExplainingFirstImpressionsModelingRecognizingAndExplainingApparentPersonalityFrom...	20180207b	7 days ago
FromBenedictCumberbatchToSherlockHolmesCharacterIdentificationInTV.pdf	20180207b	7 days ago
HowWouldSurroundVehiclesMove.pdf	20180207b	7 days ago
ImageCaptioningAtWillAVersatileSchemeForEffectivelyInjectingSentiments.pdf	20180207b	7 days ago
ImprovingMultipleObjectTrackingWithOpticalFlowAndEdgePreprocessing.pdf	20180207b	7 days ago
MobileNetsEfficientConvolutionalNeuralNetworksForMobileVisionApplications.pdf	20180207b	7 days ago
ObjectBasedReasoningInVQA.pdf	20180207b	7 days ago
ObjectDetectionInVideosByShortAndLongRangeObjectLinking.pdf	20180207b	7 days ago
Open3DAModernLibraryFor3DDataProcessing.pdf	20180207b	7 days ago
ParallelTrackingAndVerifying.pdf	20180207b	7 days ago
SSDSingleShotMultiBoxDetector.pdf	20180207b	7 days ago
Size to Depth A New Perspective for Single Image Estimation.pdf	20180207b	7 days ago
StructuredTripletLearningWithPOSTagGuidedAttention.pdf	20180207b	7 days ago
TSVizDemystificationOfDeepLearningModelsForTimeSeriesAnalysis.pdf	20180219	5 days ago
TheChallengeOfSimultaneousObjectDetectionAndPoseEstimationAComparativeStudy.pdf	20180207b	7 days ago
TrackingMultipleMovingObjectsUsingUnscentedKalmanFilteringTechniques.pdf	20180207b	7 days ago
WhatMakesGoodSyntheticTrainingDataForLearning.pdf	20180207b	7 days ago

Method Survey



PLAY ALL

Lecture Collection | Natural Language Processing with Deep Learning (Winter 2017)

19 videos • 342,803 views • Last updated on Apr 3, 2017

Stanford University School of Engineering

SUBSCRIBED 48K

Natural language processing (NLP) deals with the key artificial intelligence technology of understanding complex human language communication. This lecture series provides a thorough introduction to the cutting-edge research in deep learning applied to NLP, an approach that has recently obtained very high performance across many different NLP tasks including question answering and machine translation.

1 1:11:41 Lecture 1 | Natural Language Processing with Deep Learning
Stanford University School of Engineering

2 1:18:17 Lecture 2 | Word Vector Representations: word2vec
Stanford University School of Engineering

3 1:18:40 Lecture 3 | GloVe: Global Vectors for Word Representation
Stanford University School of Engineering

4 1:16:43 Lecture 4: Word Window Classification and Neural Networks
Stanford University School of Engineering

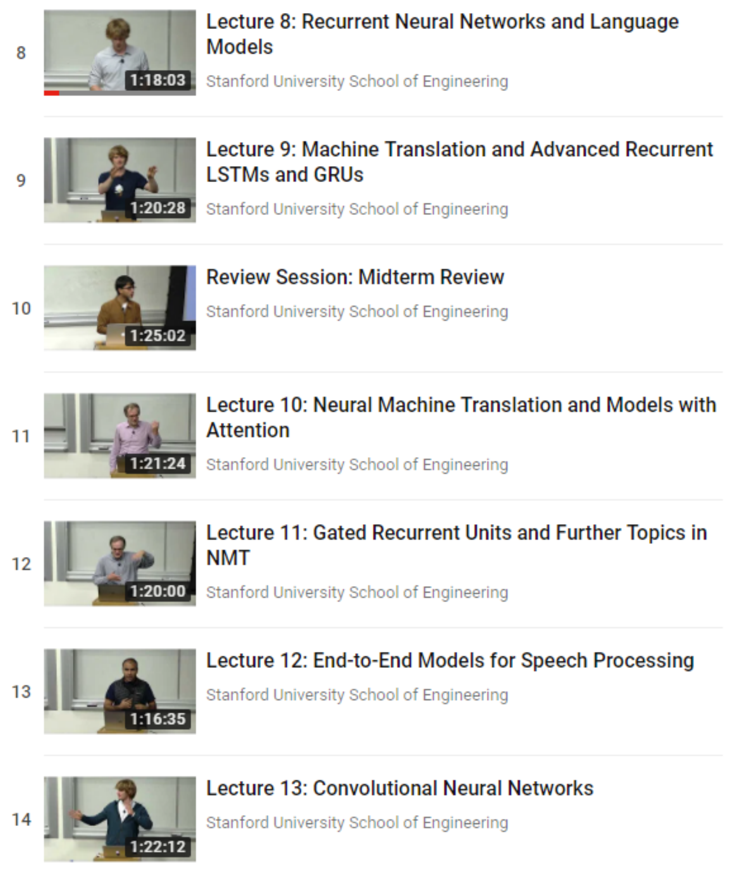
5 1:18:20 Lecture 5: Backpropagation and Project Advice
Stanford University School of Engineering

6 1:23:07 Lecture 6: Dependency Parsing
Stanford University School of Engineering

7 1:12:33 Lecture 7: Introduction to TensorFlow
Stanford University School of Engineering

Christoffer Manning & Richard Socher

Method Survey



8 1:18:03 Lecture 8: Recurrent Neural Networks and Language Models
Stanford University School of Engineering

9 1:20:28 Lecture 9: Machine Translation and Advanced Recurrent LSTMs and GRUs
Stanford University School of Engineering

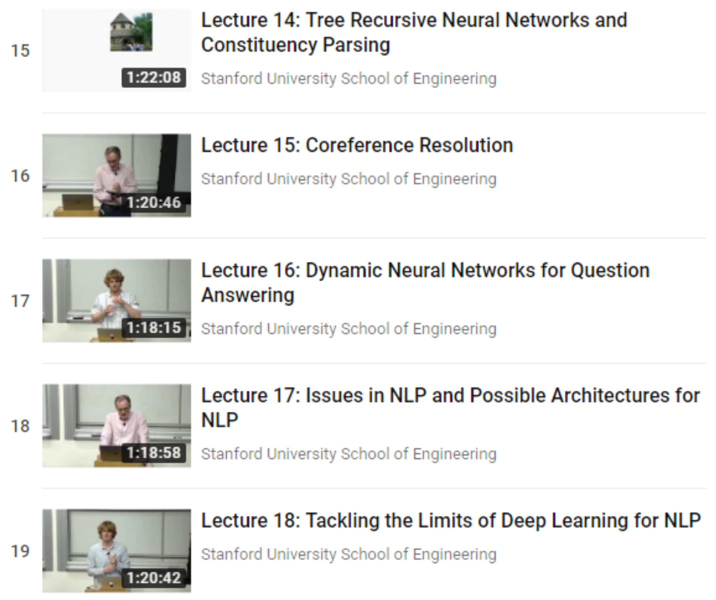
10 1:25:02 Review Session: Midterm Review
Stanford University School of Engineering

11 1:21:24 Lecture 10: Neural Machine Translation and Models with Attention
Stanford University School of Engineering

12 1:20:00 Lecture 11: Gated Recurrent Units and Further Topics in NMT
Stanford University School of Engineering

13 1:16:35 Lecture 12: End-to-End Models for Speech Processing
Stanford University School of Engineering

14 1:22:12 Lecture 13: Convolutional Neural Networks
Stanford University School of Engineering



15 1:22:08 Lecture 14: Tree Recursive Neural Networks and Constituency Parsing
Stanford University School of Engineering

16 1:20:46 Lecture 15: Coreference Resolution
Stanford University School of Engineering

17 1:18:15 Lecture 16: Dynamic Neural Networks for Question Answering
Stanford University School of Engineering

18 1:18:58 Lecture 17: Issues in NLP and Possible Architectures for NLP
Stanford University School of Engineering

19 1:20:42 Lecture 18: Tackling the Limits of Deep Learning for NLP
Stanford University School of Engineering

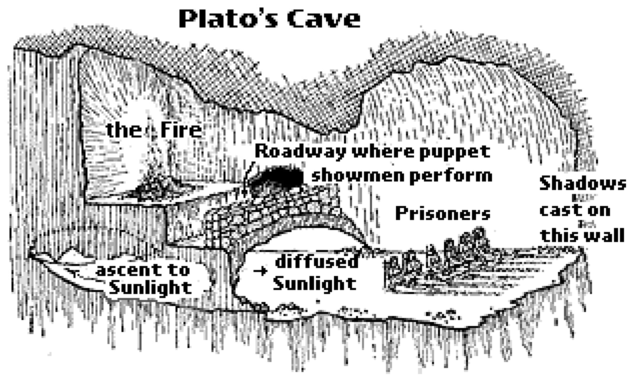
Goodfellow, Bengio, Courville: Deep Learning

Software Version 2.0

V2.0 Goal

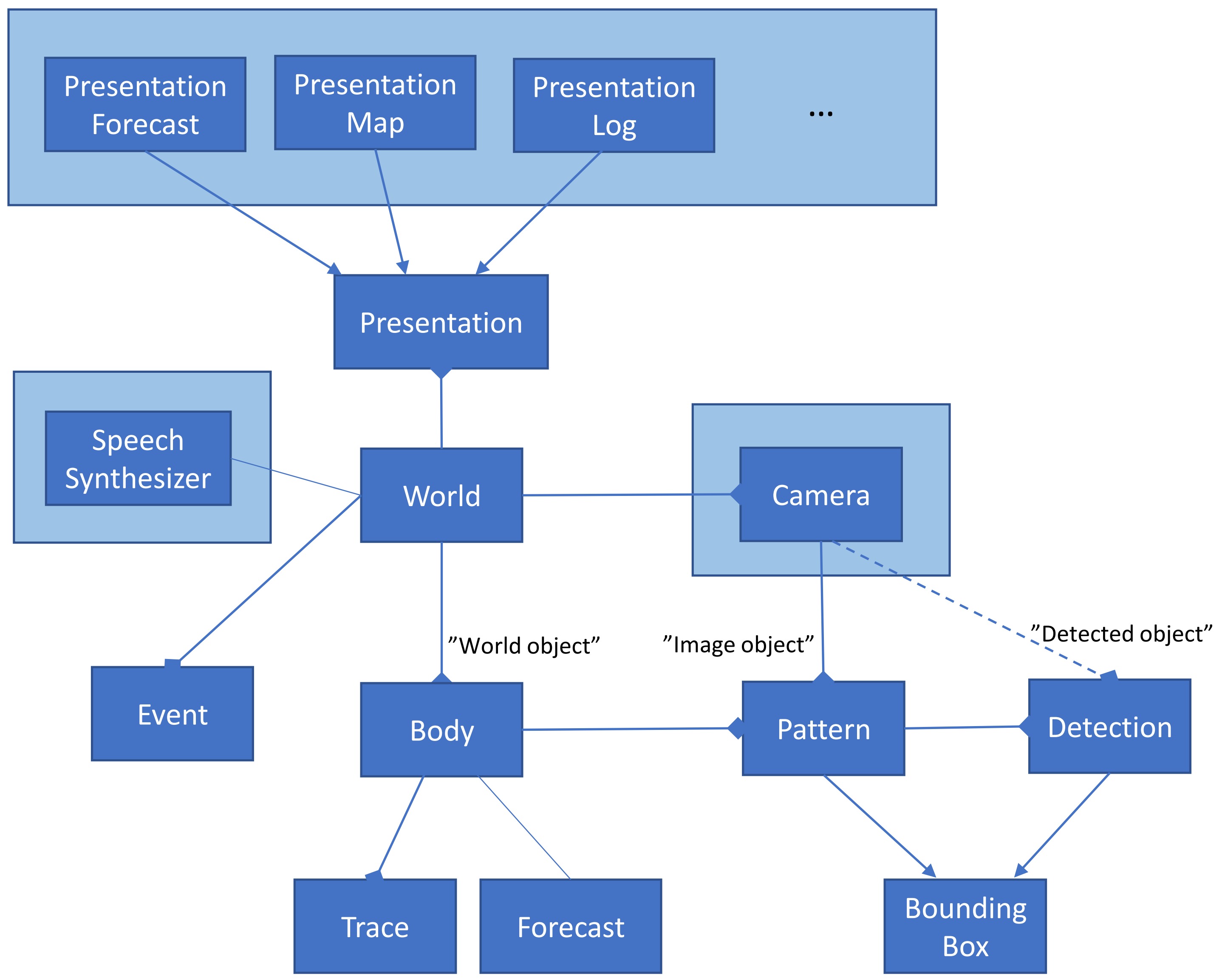
- Detected classes not hardcoded
- Object class may change
- Support for many cameras , rotations, movement
- Names less awkward, terminology fixed
- Cleaning
- Python style guide followed, excluding line length
- Code optimization
- One package
- Speech synthesizer

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

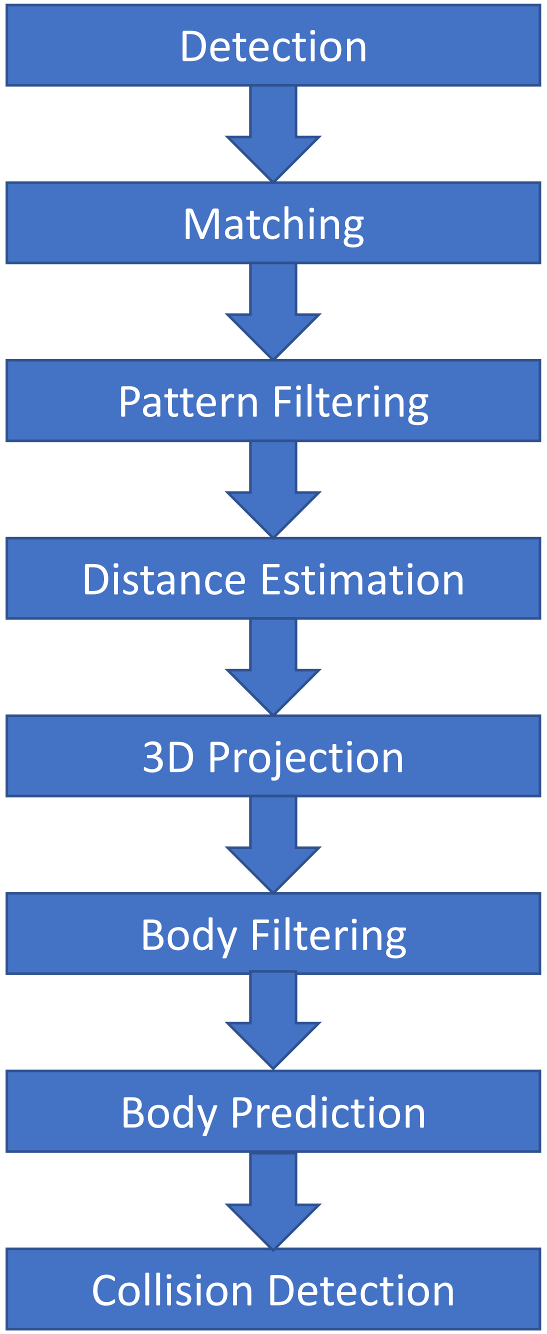


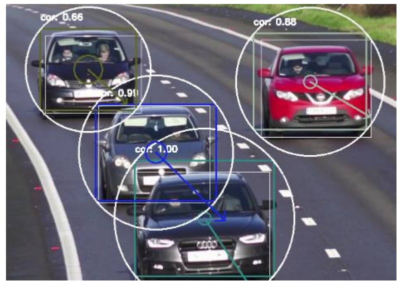
Software Version 2.0

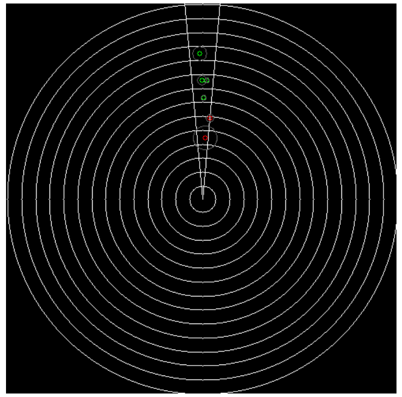
Class Diagram

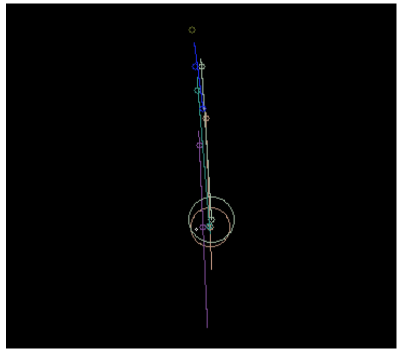


Software Version 2.0



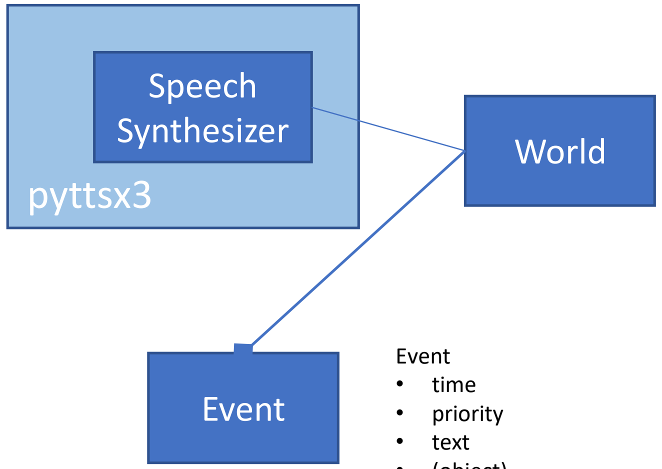






Software Version 2.0

Speech Synthesizer

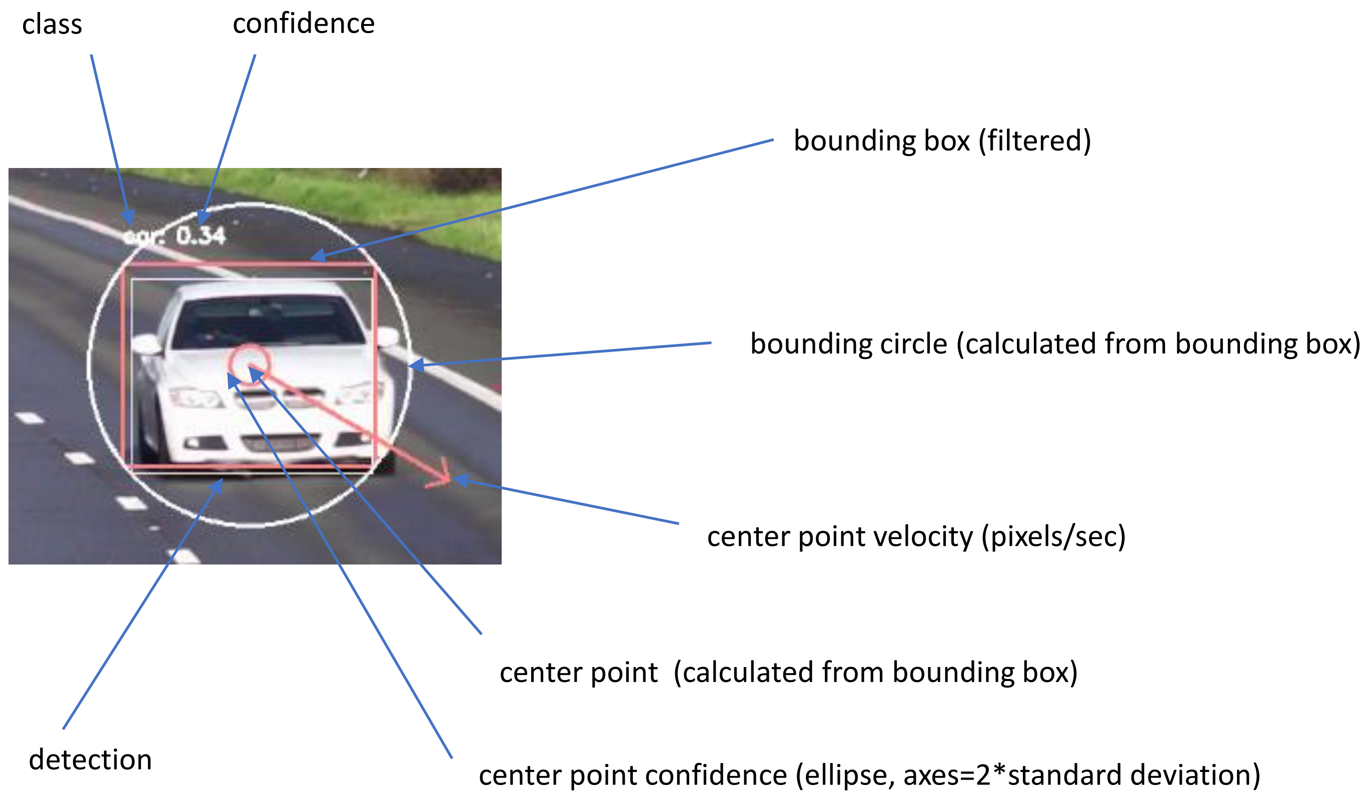


Event
- time
- priority
- text
- (object)

- Speech synthesizer based on pyttsx3 package
- Event is spelled out if priority <= 0
- Event will pause video for the duration of speech (to be changed later by using separate thread)
- Example events:
  - Body observed (1 sec after created)
  - Collision warning

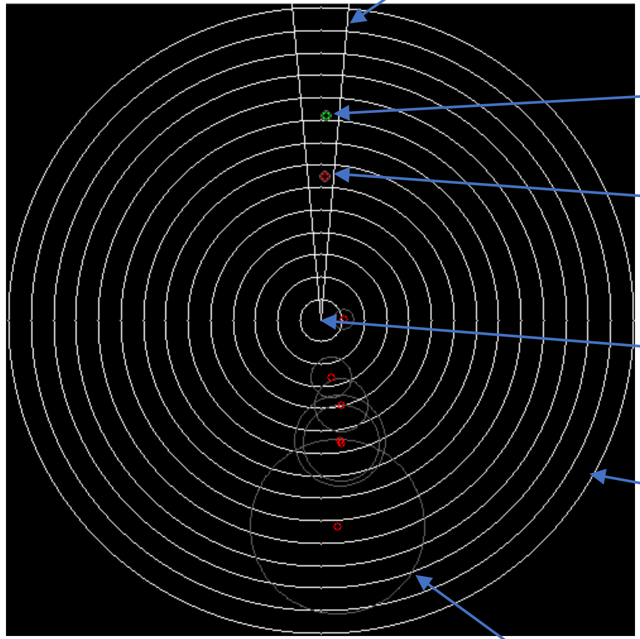
Software Version 2.0

Visual Presentation (pattern)


class
confidence
car: 0.34
bounding box (filtered)
bounding circle (calculated from bounding box)
center point velocity (pixels/sec)
center point (calculated from bounding box)
detection
center point confidence (ellipse, axes=2*standard deviation)

Software Version 2.0

Visual Presentation (body)



camera field of view

body (green, with reliable pattern)

body (red, with unreliable or lost pattern)

observer (+camera)

distance circles (every 10m)

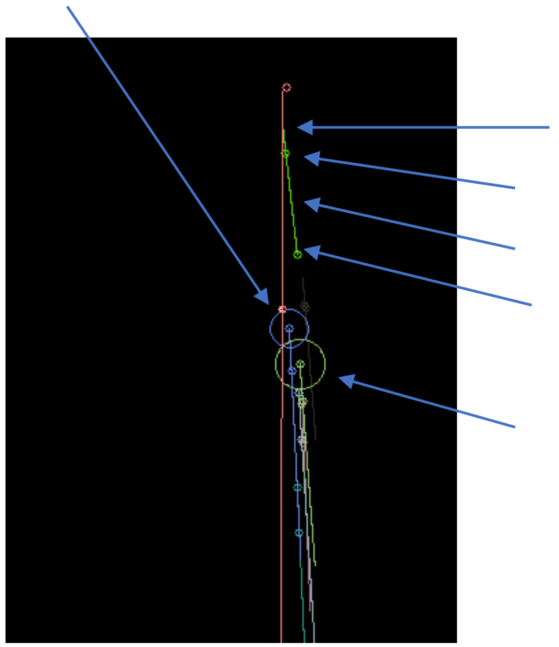
location uncertainty (ellipse, axes = standard deviation)

body size (circle, radius = class specific mean radius)

Software Version 2.0

Visual Presentation (forecast)

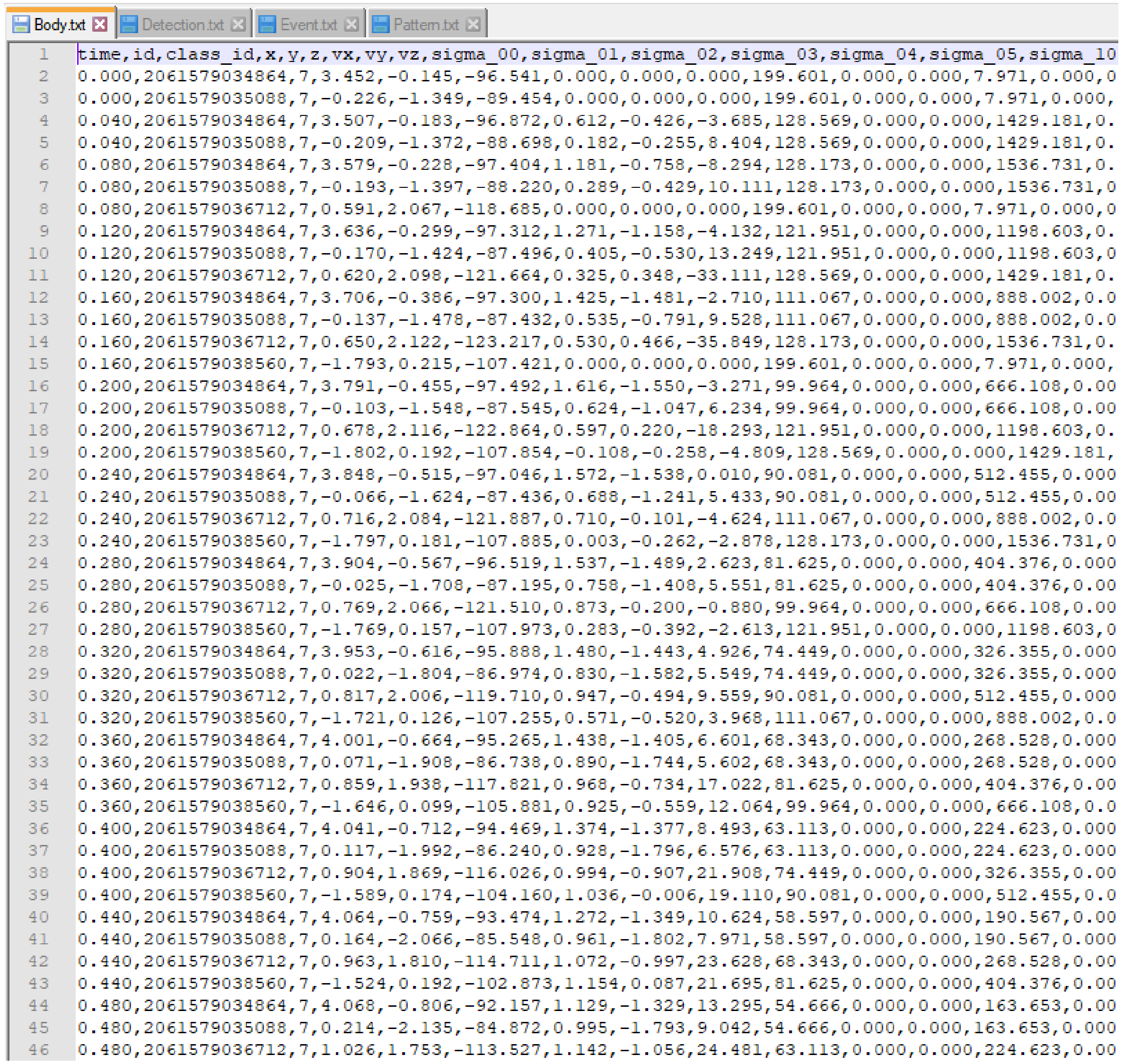
Forecast made every FORECAST_INTERVAL (=1.0) seconds


observer
body location, when forecast was made
body location, current
body location predicted path (line)
body location after FORECAST_COUNT*FORECAST_DELTA seconds
collision probability > 1

body size (circle, radius = class specific mean radius)

Software Version 2.0

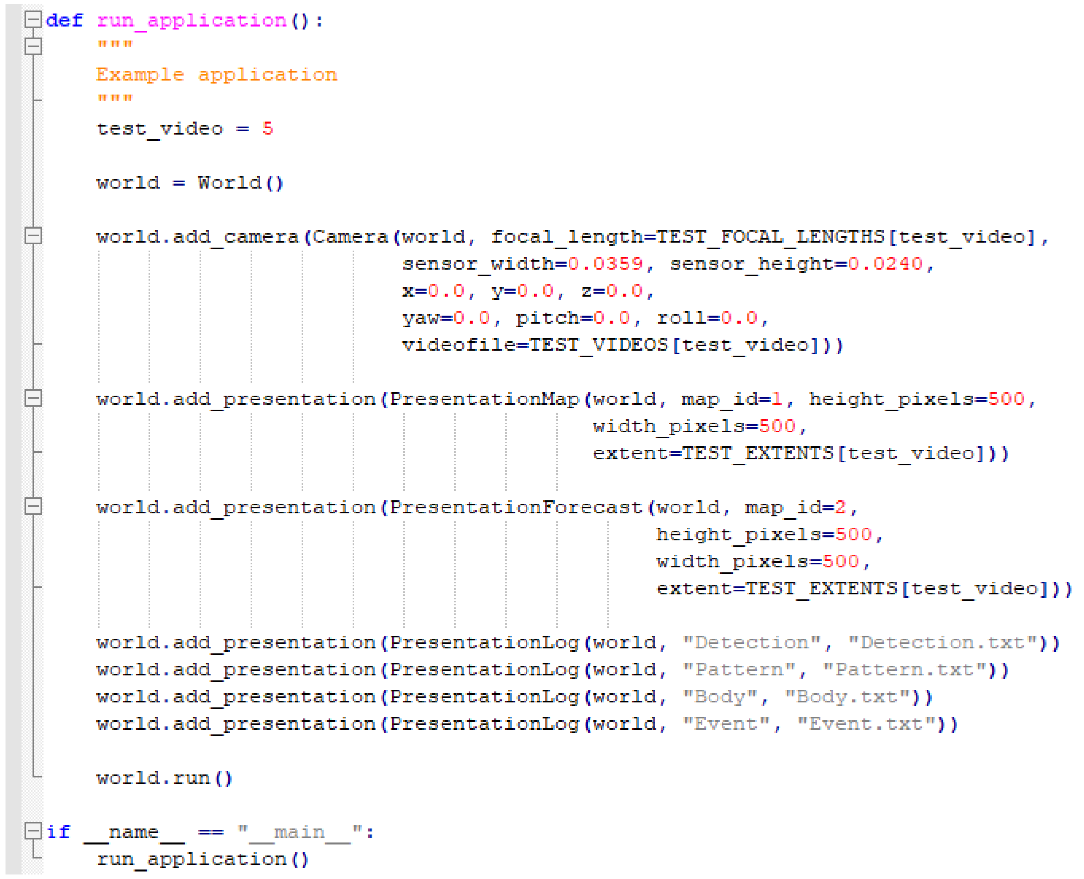
Logging



```
time,id,class_id,x,y,z,vx,vy,vz,sigma_00,sigma_01,sigma_02,sigma_03,sigma_04,sigma_05,sigma_10
0.000,2061579034864,7,3.452,-0.145,-96.541,0.000,0.000,0.000,199.601,0.000,0.000,7.971,0.000,0
0.000,2061579035088,7,-0.226,-1.349,-89.454,0.000,0.000,0.000,199.601,0.000,0.000,7.971,0.000,
0.040,2061579034864,7,3.507,-0.183,-96.872,0.612,-0.426,-3.685,128.569,0.000,0.000,1429.181,0.
0.040,2061579035088,7,-0.209,-1.372,-88.698,0.182,-0.255,8.404,128.569,0.000,0.000,1429.181,0.
0.080,2061579034864,7,3.579,-0.228,-97.404,1.181,-0.758,-8.294,128.173,0.000,0.000,1536.731,0.
0.080,2061579035088,7,-0.193,-1.397,-88.220,0.289,-0.429,10.111,128.173,0.000,0.000,1536.731,0
0.080,2061579036712,7,0.591,2.067,-118.685,0.000,0.000,0.000,199.601,0.000,0.000,7.971,0.000,0
0.120,2061579034864,7,3.636,-0.299,-97.312,1.271,-1.158,-4.132,121.951,0.000,0.000,1198.603,0.
0.120,2061579035088,7,-0.170,-1.424,-87.496,0.405,-0.530,13.249,121.951,0.000,0.000,1198.603,0
0.120,2061579036712,7,0.620,2.098,-121.664,0.325,0.348,-33.111,128.569,0.000,0.000,1429.181,0.
0.160,2061579034864,7,3.706,-0.386,-97.300,1.425,-1.481,-2.710,111.067,0.000,0.000,888.002,0.0
0.160,2061579035088,7,-0.137,-1.478,-87.432,0.535,-0.791,9.528,111.067,0.000,0.000,888.002,0.0
0.160,2061579036712,7,0.650,2.122,-123.217,0.530,0.466,-35.849,128.173,0.000,0.000,1536.731,0.
0.160,2061579038560,7,-1.793,0.215,-107.421,0.000,0.000,0.000,199.601,0.000,0.000,7.971,0.000,
0.200,2061579034864,7,3.791,-0.455,-97.492,1.616,-1.550,-3.271,99.964,0.000,0.000,666.108,0.00
0.200,2061579035088,7,-0.103,-1.548,-87.545,0.624,-1.047,6.234,99.964,0.000,0.000,666.108,0.00
0.200,2061579036712,7,0.678,2.116,-122.864,0.597,0.220,-18.293,121.951,0.000,0.000,1198.603,0.
0.200,2061579038560,7,-1.802,0.192,-107.854,-0.108,-0.258,-4.809,128.569,0.000,0.000,1429.181,
0.240,2061579034864,7,3.848,-0.515,-97.046,1.572,-1.538,0.010,90.081,0.000,0.000,512.455,0.000
0.240,2061579035088,7,-0.066,-1.624,-87.436,0.688,-1.241,5.433,90.081,0.000,0.000,512.455,0.00
0.240,2061579036712,7,0.716,2.084,-121.887,0.710,-0.101,-4.624,111.067,0.000,0.000,888.002,0.0
0.240,2061579038560,7,-1.797,0.181,-107.885,0.003,-0.262,-2.878,128.173,0.000,0.000,1536.731,0
0.280,2061579034864,7,3.904,-0.567,-96.519,1.537,-1.489,2.623,81.625,0.000,0.000,404.376,0.000
0.280,2061579035088,7,-0.025,-1.708,-87.195,0.758,-1.408,5.551,81.625,0.000,0.000,404.376,0.00
0.280,2061579036712,7,0.769,2.066,-121.510,0.873,-0.200,-0.880,99.964,0.000,0.000,666.108,0.00
0.280,2061579038560,7,-1.769,0.157,-107.973,0.283,-0.392,-2.613,121.951,0.000,0.000,1198.603,0
0.320,2061579034864,7,3.953,-0.616,-95.888,1.480,-1.443,4.926,74.449,0.000,0.000,326.355,0.000
0.320,2061579035088,7,0.022,-1.804,-86.974,0.830,-1.582,5.549,74.449,0.000,0.000,326.355,0.000
0.320,2061579036712,7,0.817,2.006,-119.710,0.947,-0.494,9.559,90.081,0.000,0.000,512.455,0.000
0.320,2061579038560,7,-1.721,0.126,-107.255,0.571,-0.520,3.968,111.067,0.000,0.000,888.002,0.0
0.360,2061579034864,7,4.001,-0.664,-95.265,1.438,-1.405,6.601,68.343,0.000,0.000,268.528,0.000
0.360,2061579035088,7,0.071,-1.908,-86.738,0.890,-1.744,5.602,68.343,0.000,0.000,268.528,0.000
0.360,2061579036712,7,0.859,1.938,-117.821,0.968,-0.734,17.022,81.625,0.000,0.000,404.376,0.00
0.360,2061579038560,7,-1.646,0.099,-105.881,0.925,-0.559,12.064,99.964,0.000,0.000,666.108,0.0
0.400,2061579034864,7,4.041,-0.712,-94.469,1.374,-1.377,8.493,63.113,0.000,0.000,224.623,0.000
0.400,2061579035088,7,0.117,-1.992,-86.240,0.928,-1.796,6.576,63.113,0.000,0.000,224.623,0.000
0.400,2061579036712,7,0.904,1.869,-116.026,0.994,-0.907,21.908,74.449,0.000,0.000,326.355,0.00
0.400,2061579038560,7,-1.589,0.174,-104.160,1.036,-0.006,19.110,90.081,0.000,0.000,512.455,0.0
0.440,2061579034864,7,4.064,-0.759,-93.474,1.272,-1.349,10.624,58.597,0.000,0.000,190.567,0.00
0.440,2061579035088,7,0.164,-2.066,-85.548,0.961,-1.802,7.971,58.597,0.000,0.000,190.567,0.000
0.440,2061579036712,7,0.963,1.810,-114.711,1.072,-0.997,23.628,68.343,0.000,0.000,268.528,0.00
0.440,2061579038560,7,-1.524,0.192,-102.873,1.154,0.087,21.695,81.625,0.000,0.000,404.376,0.00
0.480,2061579034864,7,4.068,-0.806,-92.157,1.129,-1.329,13.295,54.666,0.000,0.000,163.653,0.00
0.480,2061579035088,7,0.214,-2.135,-84.872,0.995,-1.793,9.042,54.666,0.000,0.000,163.653,0.000
0.480,2061579036712,7,1.026,1.753,-113.527,1.142,-1.056,24.481,63.113,0.000,0.000,224.623,0.00
```

Software Version 2.0

How to use?



```
def run_application():
    """
    Example application
    """
    test_video = 5

    world = World()

    world.add_camera(Camera(world, focal_length=TEST_FOCAL_LENGTHS[test_video],
                            sensor_width=0.0359, sensor_height=0.0240,
                            x=0.0, y=0.0, z=0.0,
                            yaw=0.0, pitch=0.0, roll=0.0,
                            videofile=TEST_VIDEOS[test_video]))

    world.add_presentation(PresentationMap(world, map_id=1, height_pixels=500,
                                           width_pixels=500,
                                           extent=TEST_EXTENTS[test_video]))

    world.add_presentation(PresentationForecast(world, map_id=2,
                                                height_pixels=500,
                                                width_pixels=500,
                                                extent=TEST_EXTENTS[test_video]))

    world.add_presentation(PresentationLog(world, "Detection", "Detection.txt"))
    world.add_presentation(PresentationLog(world, "Pattern", "Pattern.txt"))
    world.add_presentation(PresentationLog(world, "Body", "Body.txt"))
    world.add_presentation(PresentationLog(world, "Event", "Event.txt"))

    world.run()

if __name__ == "__main__":
    run_application()
```

Demo

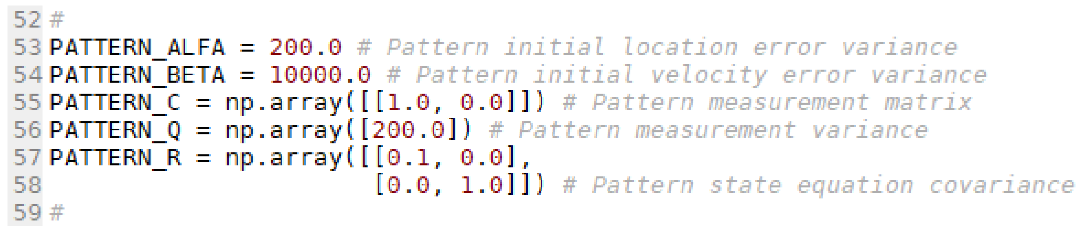
- 3-5 videos
- Short program code review

Pattern Filtering

Why pattern filtering?

- Reduces object detection noise (bounding box)
- Provides prediction for pattern location in the next frame - matching easier
- Predicts pattern location when detection is missing

Hyperparameters:

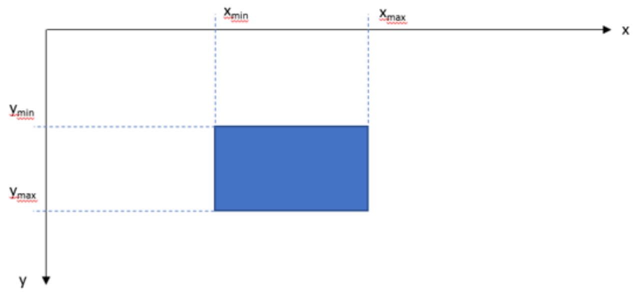


```
#
PATTERN_ALFA = 200.0 # Pattern initial location error variance
PATTERN_BETA = 10000.0 # Pattern initial velocity error variance
PATTERN_C = np.array([[1.0, 0.0]]) # Pattern measurement matrix
PATTERN_Q = np.array([200.0]) # Pattern measurement variance
PATTERN_R = np.array([[0.1, 0.0],
                      [0.0, 1.0]]) # Pattern state equation covariance
#
```

Pattern Filtering

Pattern Kalman Filtering

Bounding box edge coordinates



Pattern location (bounding box) is determined by four edge coordinates: x_{min}, x_{max}, y_{min} and y_{max}. vx_{min}, vx_{max}, vy_{min} and vy_{max} are corresponding velocities.

Each edge coordinate is filtered separately and identically. x_{min} is used here as an example.

State equation in differential form:

$$\frac{d}{dt}\begin{bmatrix} x_{min}(t) \\ vx_{min}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} * \begin{bmatrix} x_{min}(t) \\ vx_{min}(t) \end{bmatrix} + \epsilon(t)$$

State equation in difference form:

$$\begin{bmatrix} x_{min}(k+1) \\ vx_{min}(k+1) \end{bmatrix} = A * \begin{bmatrix} x_{min}(k) \\ vx_{min}(k) \end{bmatrix} + \varepsilon(k)$$

$$A = \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix}$$

where Δ is the time increment and ε Gaussian noise with covariance R:

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 1.0 \end{bmatrix}$$

Measurement equation

$$z(k) = C * \begin{bmatrix} x_{min}(k) \\ vx_{min}(k) \end{bmatrix} + \delta(k)$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

where δ is Gaussian noise with covariance matrix Q:

$$Q = [200.0]$$

Kalman filter initialization:

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

where x_{min} (0) is the first location measurement.

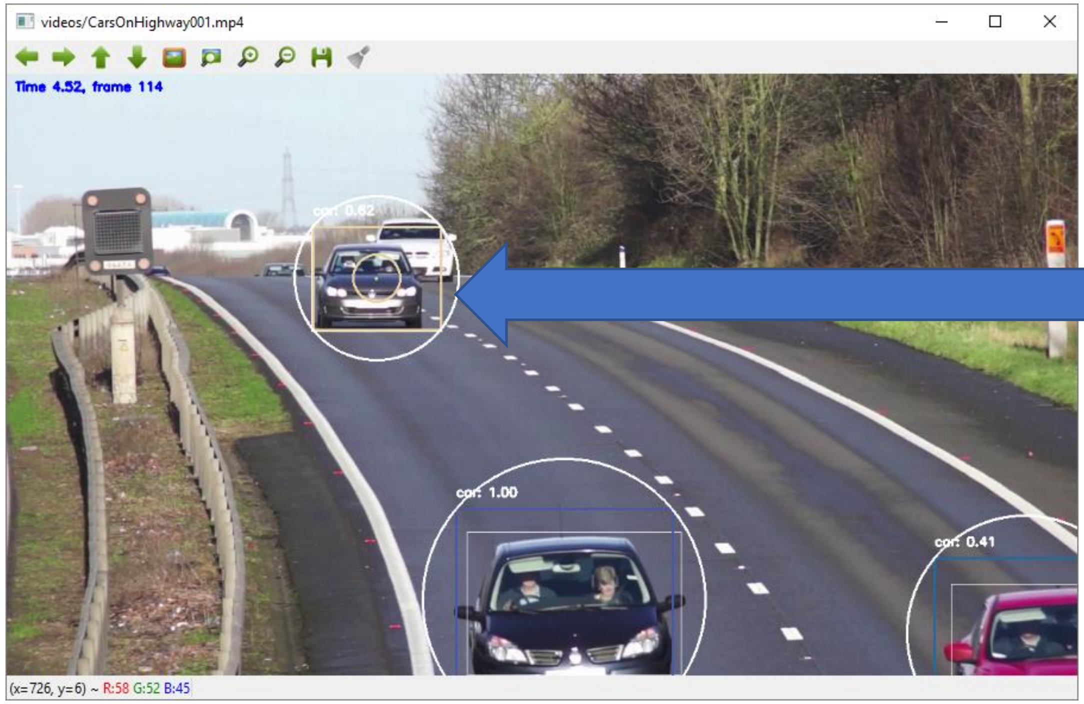
$$\Sigma(0) = \begin{bmatrix} 200.0 & 0 \\ 0 & 10\,000.0 \end{bmatrix}$$

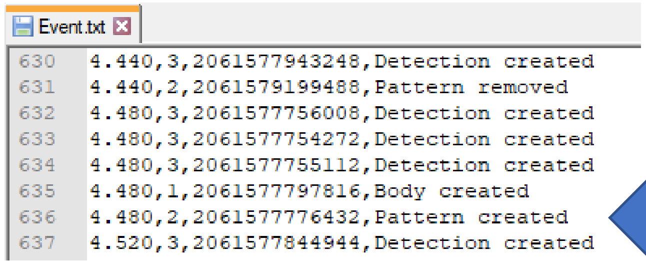
Kalman filter update:

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

Pattern Filtering

Example:



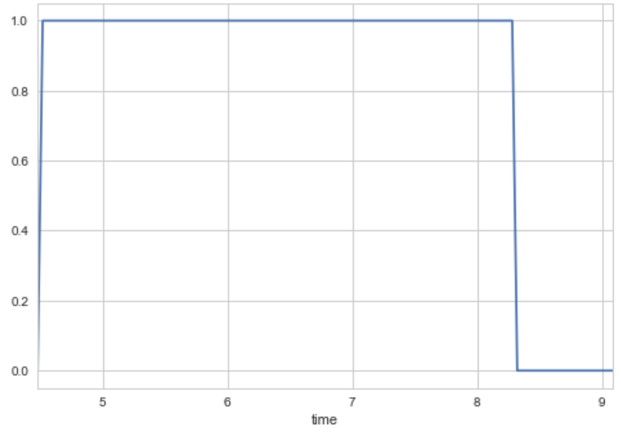

Event.txt

```
4.440,3,2061577943248,Detection created
4.440,2,2061579199488,Pattern removed
4.480,3,2061577756008,Detection created
4.480,3,2061577754272,Detection created
4.480,3,2061577755112,Detection created
4.480,1,2061577797816,Body created
4.480,2,2061577776432,Pattern created
4.520,3,2061577844944,Detection created
```

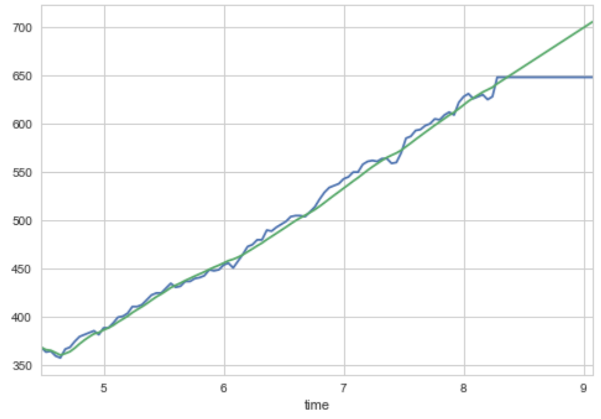
Pattern Filtering

Matched status, coordinate and velocity

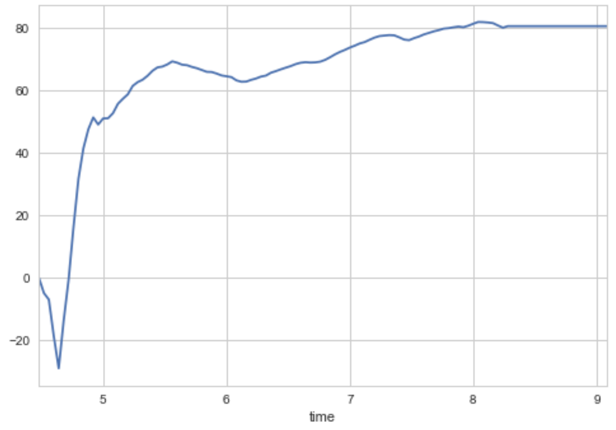
```
In [9]: data_one['matched'].plot()
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1b3759afe80>
```



```
In [10]: data_one['x_min_d'].plot() # blue, measurement
         data_one['x_min'].plot() # green, filtered + predicted
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1b376ccbe48>
```



```
In [11]: data_one['vx_min'].plot()
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1b376e50f98>
```

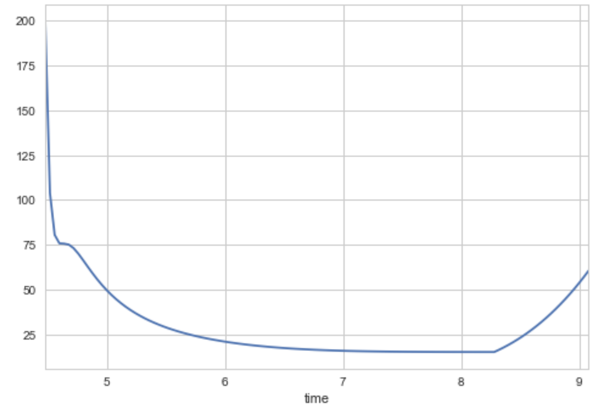


Pattern Filtering

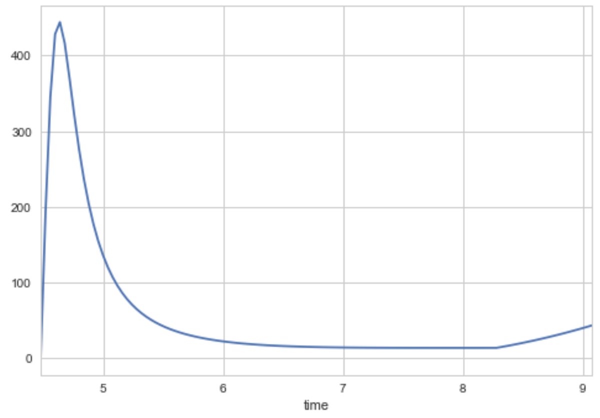
Covariance matrix

(0=location, 1=velocity)

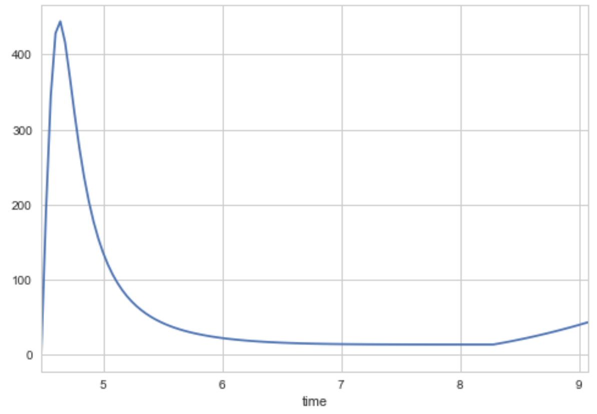
```
In [16]: data_one['sigma_x_min_00'].plot()
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1b3771859b0>
```



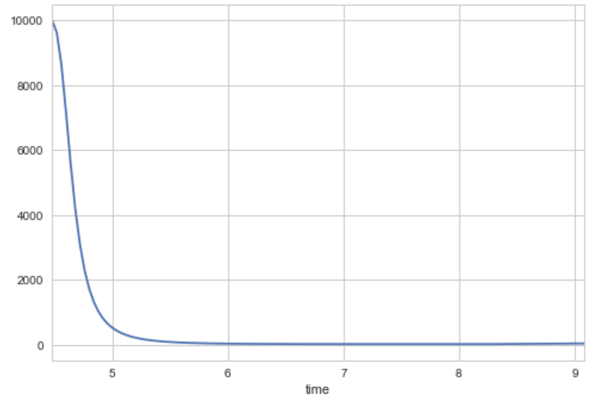
```
In [17]: data_one['sigma_x_min_01'].plot()
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x1b37726b5c0>
```



```
In [18]: data_one['sigma_x_min_10'].plot()
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1b3772b9da0>
```

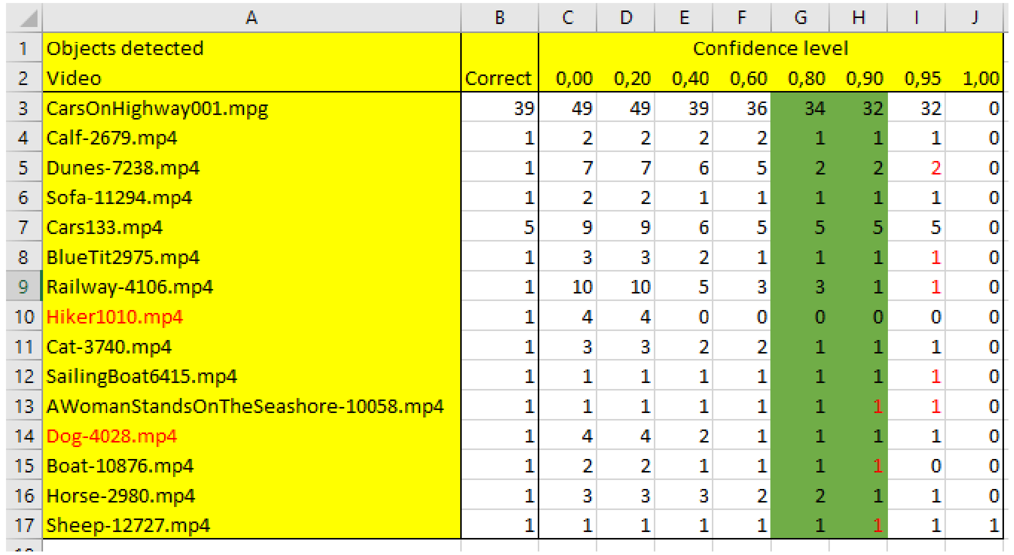


```
In [19]: data_one['sigma_x_min_11'].plot()
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1b377347358>
```



Matching / Confidence Level

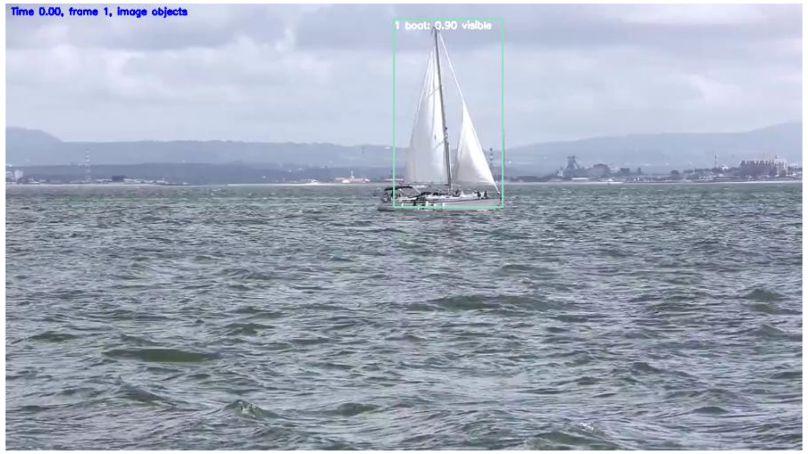
Minimum confidence level to create a pattern (and body) was varied between 0 and 1. The number of bodies created was compared to the true number of objects in the video.



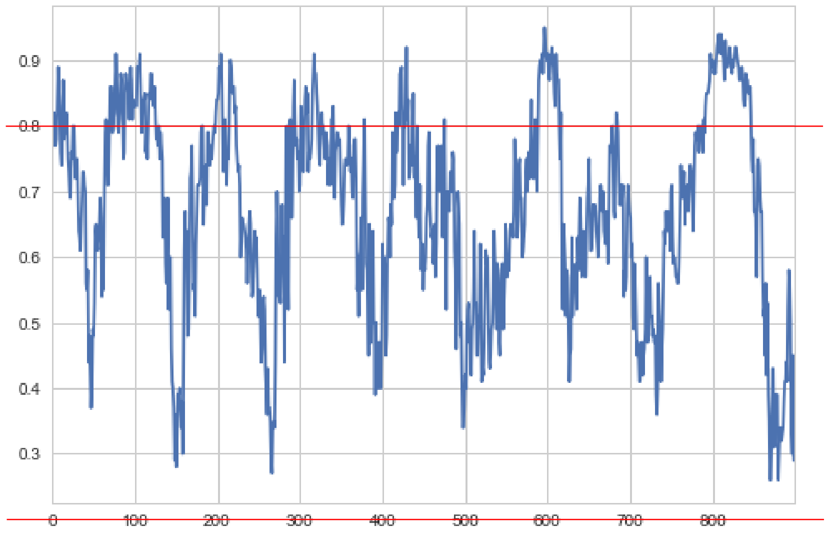
Objects detected		Confidence level							
Video	Correct	0,00	0,20	0,40	0,60	0,80	0,90	0,95	1,00
CarsOnHighway001.mpg	39	49	49	39	36	34	32	32	0
Calf-2679.mp4	1	2	2	2	2	1	1	1	0
Dunes-7238.mp4	1	7	7	6	5	2	2	2	0
Sofa-11294.mp4	1	2	2	1	1	1	1	1	0
Cars133.mp4	5	9	9	6	5	5	5	5	0
BlueTit2975.mp4	1	3	3	2	1	1	1	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 pattern is between 0.8 and 0.9.

Matching / Confidence Level



Confidence level has dynamics



create

update

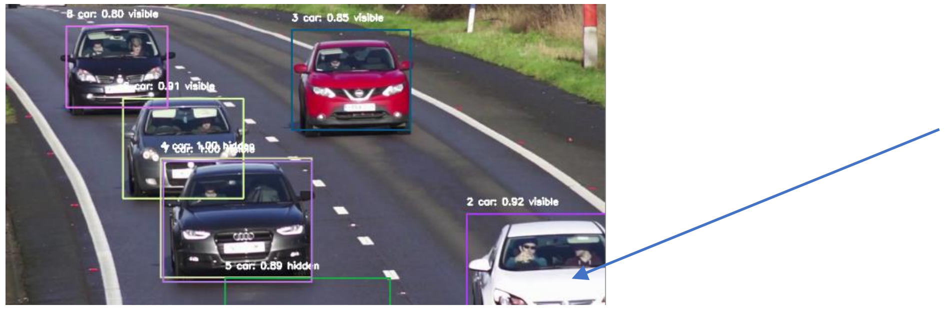
ignore

Different levels for creating and updating image object.
Hyperparameters:

- CONFIDENCE_LEVEL_CREATE (0.8)
- CONFIDENCE_LEVEL_UPDATE (0.2)

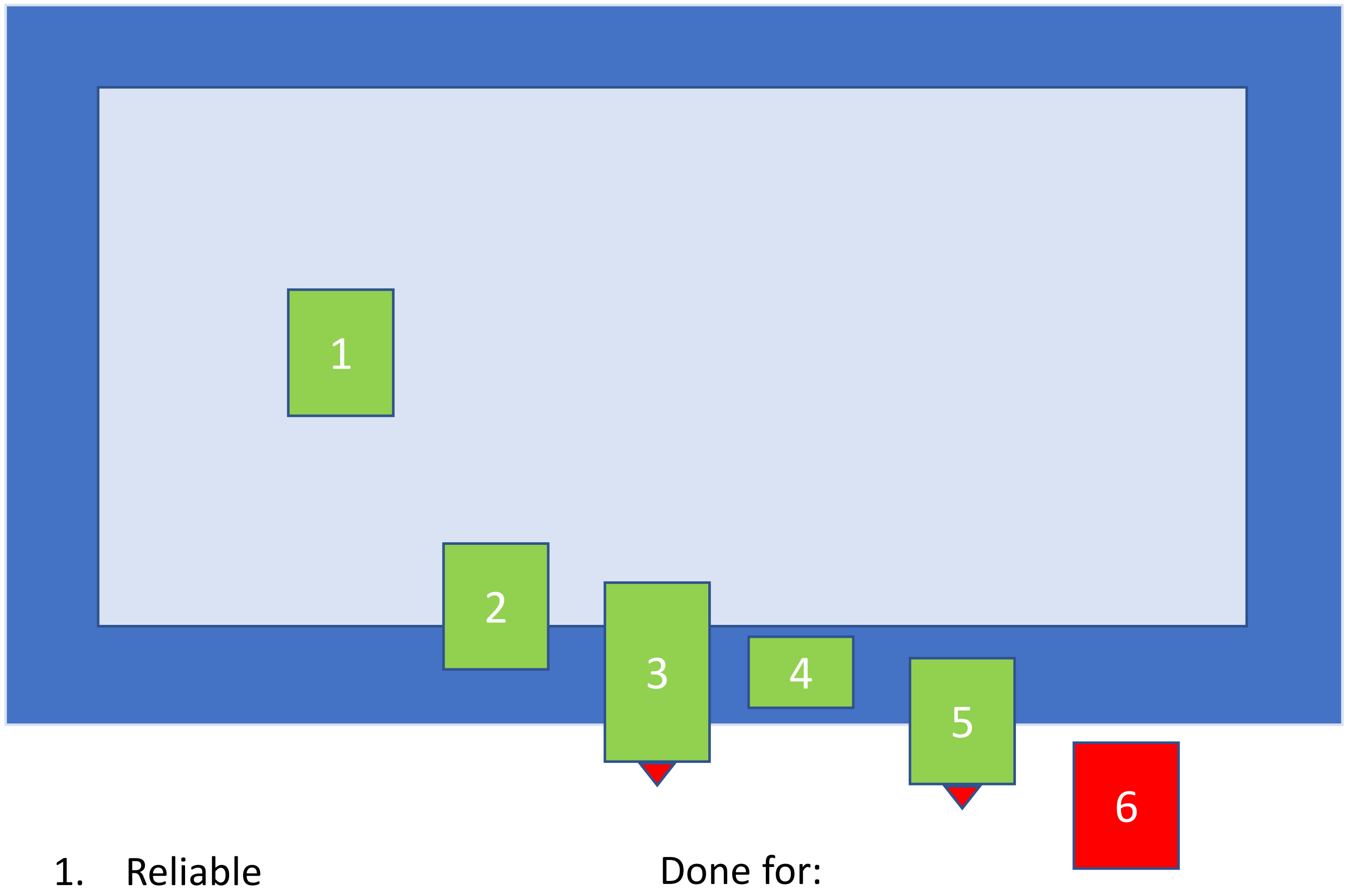
Matching / Border Behaviour

The problem:
Bounding box size and form are distorded near edges



Hyperparameter BORDER_WIDTH (30)

Matching / Border Behaviour

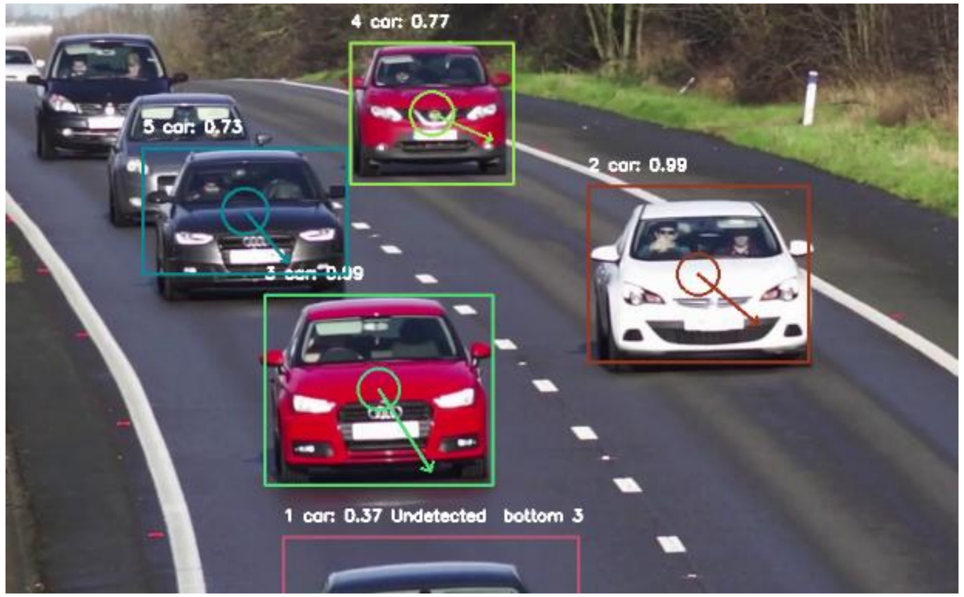


1. Reliable
2. Reliable
3. Unreliable, not created
4. Reliable, not created
5. Unreliable, not created
6. Unreliable, removed

Done for:
- left
- right
- top
- bottom

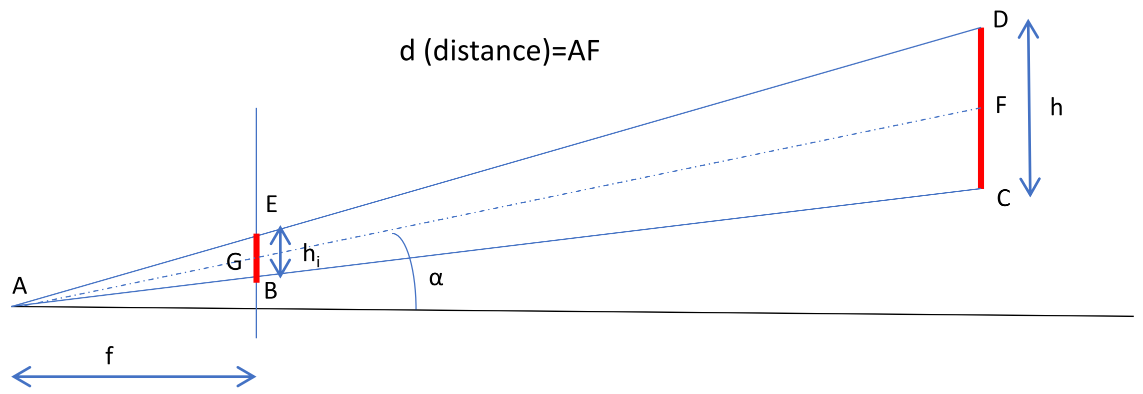
If a pattern touches 3 borders, it is removed
Reliable = body information updated

Matching / Pattern Retention



Patterns are removed if not detected in RETENTION_COUNT_MAX (30) successive frames. Body continues to live.

Distance Estimation



Similar triangles AGE and AFD:

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

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

Distance Estimation

Similar equations for horizontal direction (β=azimuth):

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

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

Assumptions:
- *equal vertical/horizontal pixel spacing*
- *optical axis in image center*

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
l_i = object length (pixels) = 100
l = object length (m) = 1.0 m
f = focal length (m) = 0.050 m
α = *altitude (rad) = 0.0*
β = *azimuth (rad) = 0.0*

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

Distance Estimation

Question: How to compare pattern and body sizes for distance estimation?

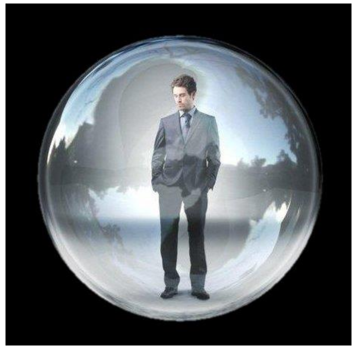
Height or width alone might be misleading.

Some form of (3D) spatial simplification is needed, like

- cube
- rectangular prism
- cylinder
- sphere (probably the easiest math)

Uncertainty should be modeled.

Solution:
pattern circle <---> body sphere

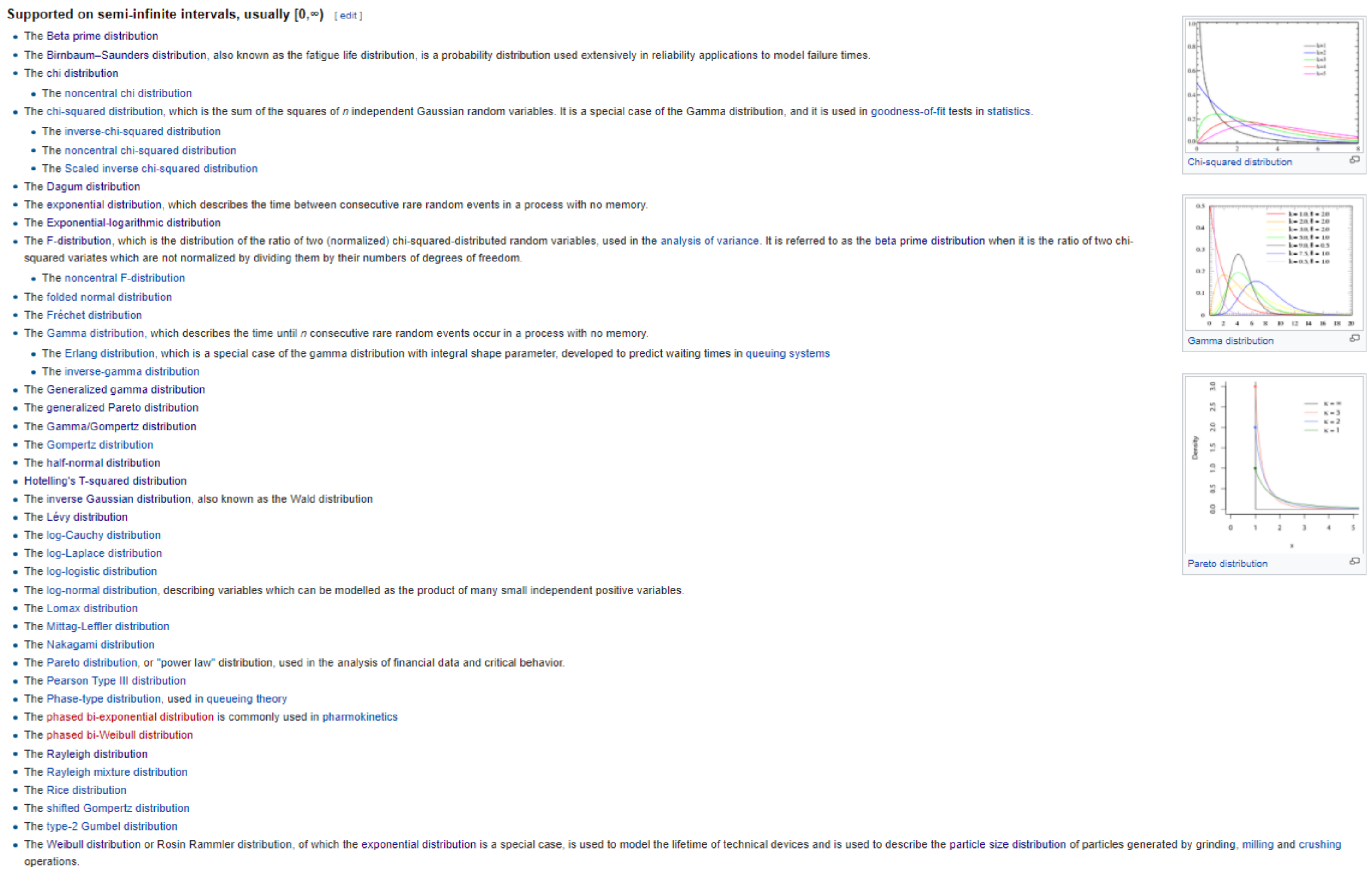


Distance Estimation

Body size distribution

Distibution should

- be defined in [0,∞]
- mode > 0
- simple
- skew controllable



Supported on semi-infinite intervals, usually [0,∞) [edit]

- The Beta prime distribution
- The Birnbaum–Saunders distribution, also known as the fatigue life distribution, is a probability distribution used extensively in reliability applications to model failure times.
- The chi distribution
  - The noncentral chi distribution
- The chi-squared distribution, which is the sum of the squares of *n* independent Gaussian random variables. It is a special case of the Gamma distribution, and it is used in goodness-of-fit tests in statistics.
  - The inverse-chi-squared distribution
  - The noncentral chi-squared distribution
  - The Scaled inverse chi-squared distribution
- The Dagum distribution
- The exponential distribution, which describes the time between consecutive rare random events in a process with no memory.
- The Exponential-logarithmic distribution
- The F-distribution, which is the distribution of the ratio of two (normalized) chi-squared-distributed random variables, used in the analysis of variance. It is referred to as the beta prime distribution when it is the ratio of two chi-squared variates which are not normalized by dividing them by their numbers of degrees of freedom.
  - The noncentral F-distribution
- The folded normal distribution
- The Fréchet distribution
- The Gamma distribution, which describes the time until *n* consecutive rare random events occur in a process with no memory.
  - The Erlang distribution, which is a special case of the gamma distribution with integral shape parameter, developed to predict waiting times in queuing systems
  - The inverse-gamma distribution
- The Generalized gamma distribution
- The generalized Pareto distribution
- The Gamma/Gompertz distribution
- The Gompertz distribution
- The half-normal distribution
- Hotelling's T-squared distribution
- The inverse Gaussian distribution, also known as the Wald distribution
- The Lévy distribution
- The log-Cauchy distribution
- The log-Laplace distribution
- The log-logistic distribution
- The log-normal distribution, describing variables which can be modelled as the product of many small independent positive variables.
- The Lomax distribution
- The Mittag-Leffler distribution
- The Nakagami distribution
- The Pareto distribution, or "power law" distribution, used in the analysis of financial data and critical behavior.
- The Pearson Type III distribution
- The Phase-type distribution, used in queueing theory
- The phased bi-exponential distribution is commonly used in pharmokinetics
- The phased bi-Weibull distribution
- The Rayleigh distribution
- The Rayleigh mixture distribution
- The Rice distribution
- The shifted Gompertz distribution
- The type-2 Gumbel distribution
- The Weibull distribution or Rosin Rammler distribution, of which the exponential distribution is a special case, is used to model the lifetime of technical devices and is used to describe the particle size distribution of particles generated by grinding, milling and crushing operations.

Chi-squared distribution

Gamma distribution

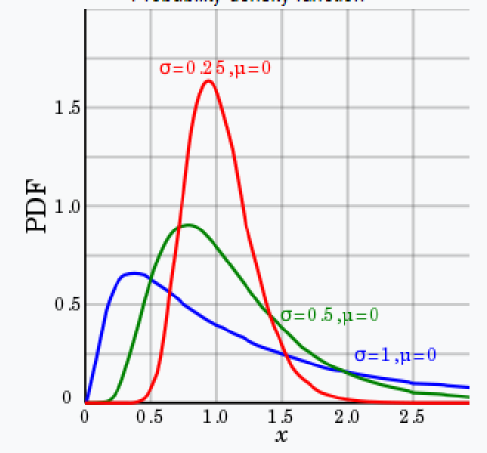
Pareto distribution

Distance Estimation

Log-normal distribution

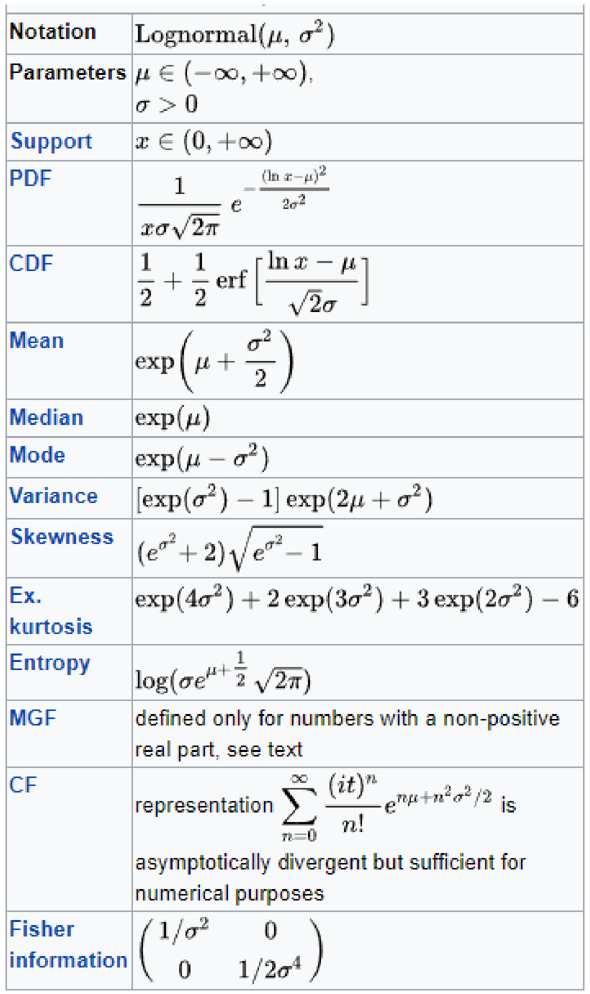
Log-normal

Probability density function



Some log-normal density functions with identical parameter μ but differing parameters σ

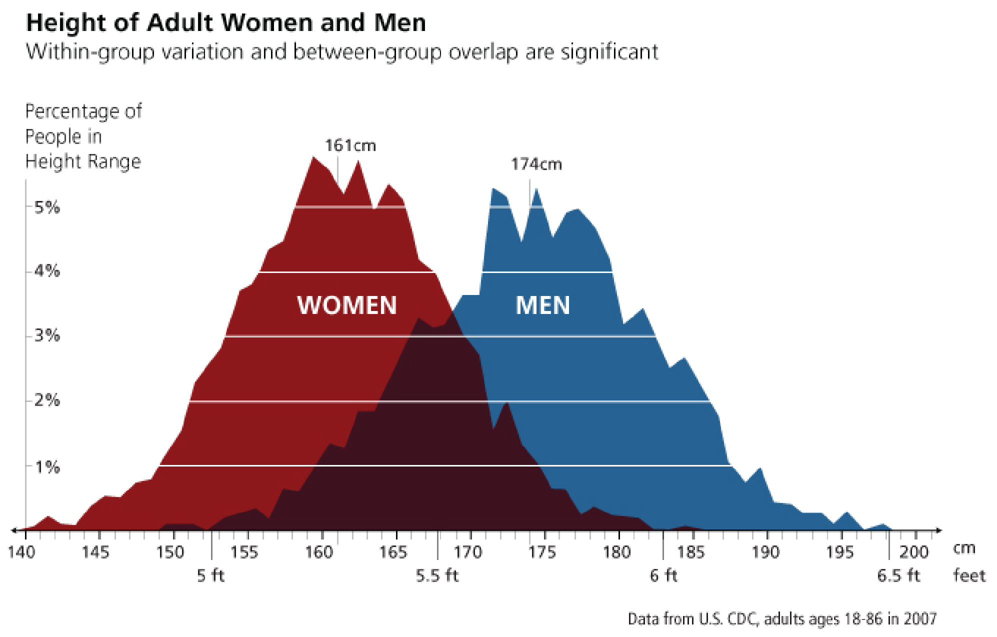
Used in the context of describing human height distribution

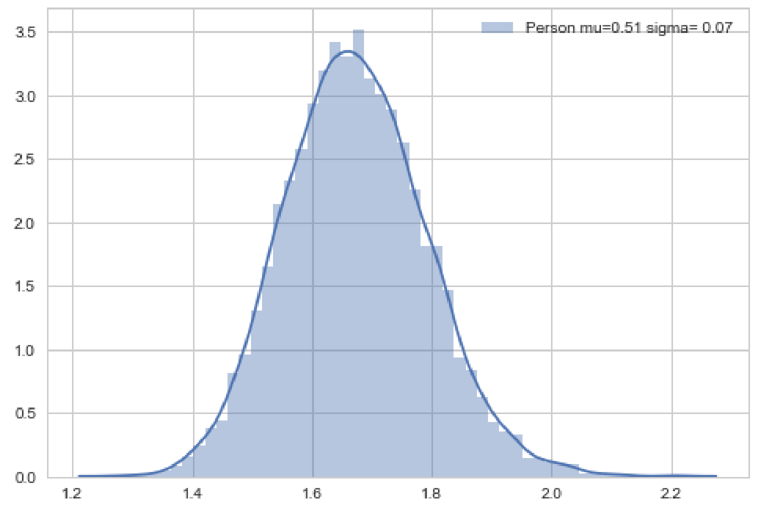


Notation	$\operatorname{Lognormal}(\mu,\, \sigma^2)$
Parameters	$\mu \in (-\infty, +\infty)$, $\sigma > 0$
Support	$x \in (0, +\infty)$
PDF	$\frac{1}{x\sigma\sqrt{2\pi}}\, e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}$
CDF	$\frac{1}{2} + \frac{1}{2}\operatorname{erf}\left[\frac{\ln x-\mu}{\sqrt{2}\sigma}\right]$
Mean	$\exp\left(\mu + \frac{\sigma^2}{2}\right)$
Median	$\exp(\mu)$
Mode	$\exp(\mu - \sigma^2)$
Variance	$[\exp(\sigma^2) - 1]\exp(2\mu + \sigma^2)$
Skewness	$(e^{\sigma^2} + 2)\sqrt{e^{\sigma^2} - 1}$
Ex. kurtosis	$\exp(4\sigma^2) + 2\exp(3\sigma^2) + 3\exp(2\sigma^2) - 6$
Entropy	$\log(\sigma e^{\mu+\frac{1}{2}}\sqrt{2\pi})$
MGF	defined only for numbers with a non-positive real part, see text
CF	representation $\sum_{n=0}^{\infty}\frac{(it)^n}{n!}e^{n\mu+n^2\sigma^2/2}$ is asymptotically divergent but sufficient for numerical purposes
Fisher information	$\begin{pmatrix} 1/\sigma^2 & 0 \\ 0 & 1/2\sigma^4 \end{pmatrix}$

Distance Estimation

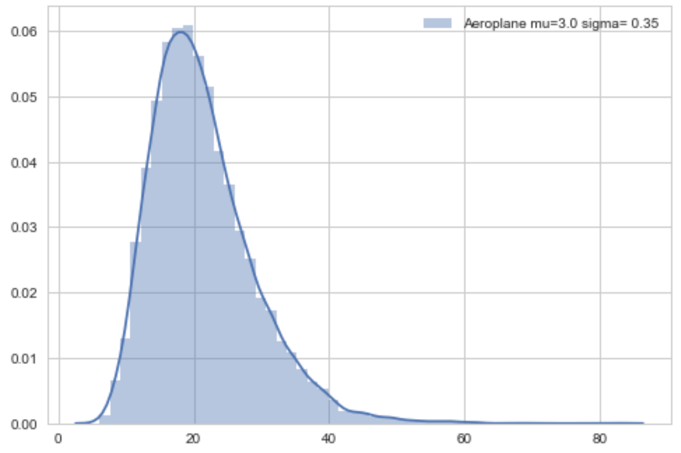
Example: Person

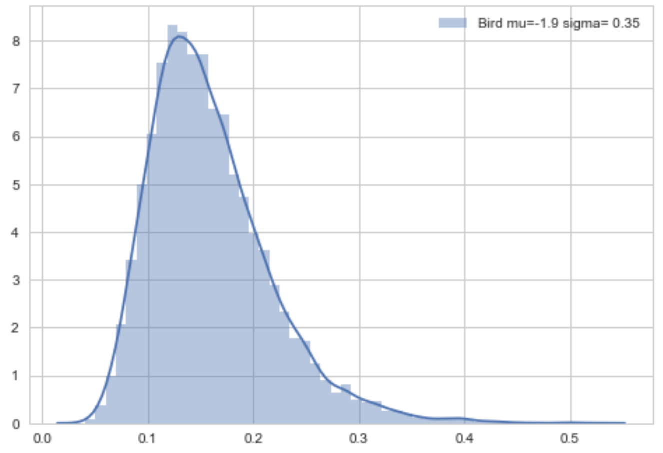


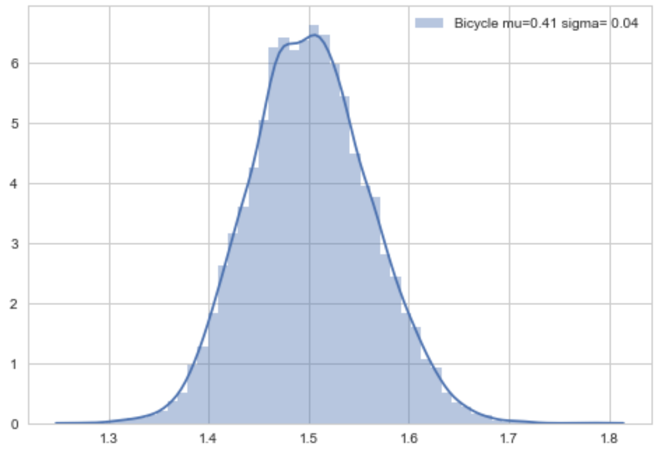


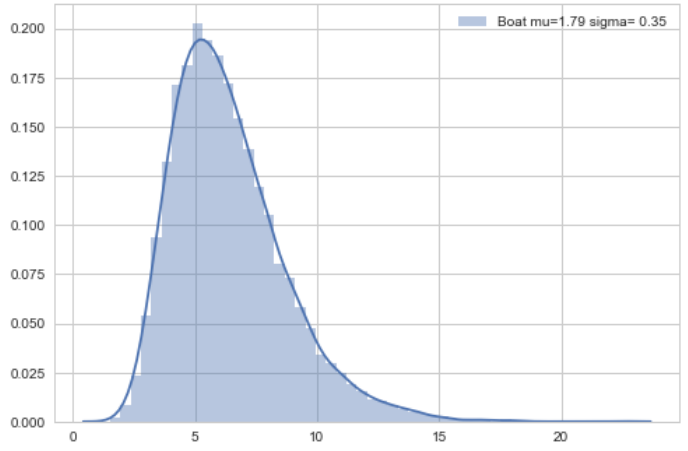
Distance Estimation

Bubble diameters (2*radius)



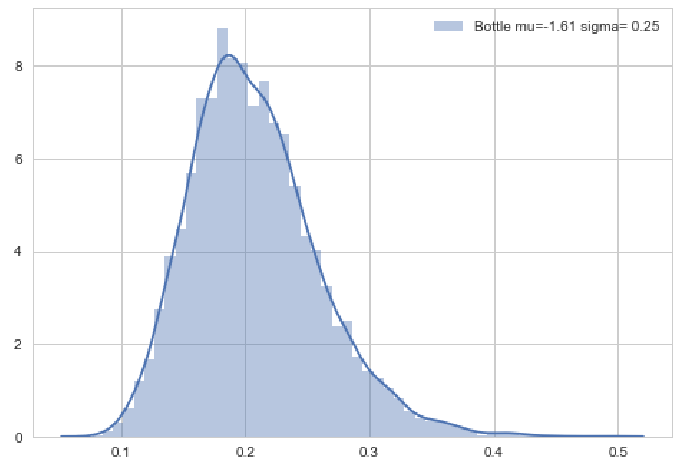


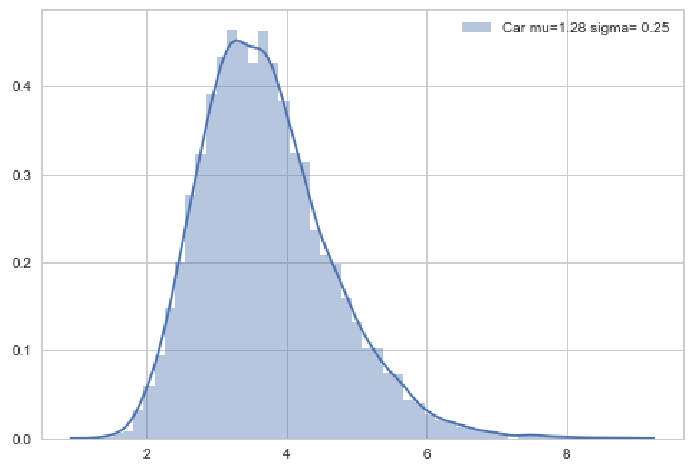


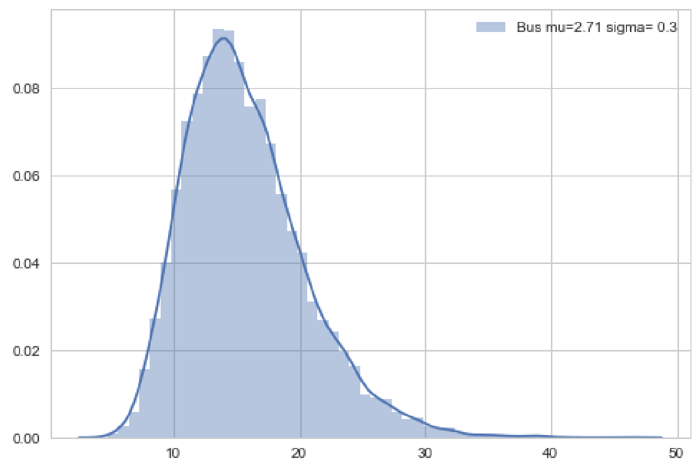


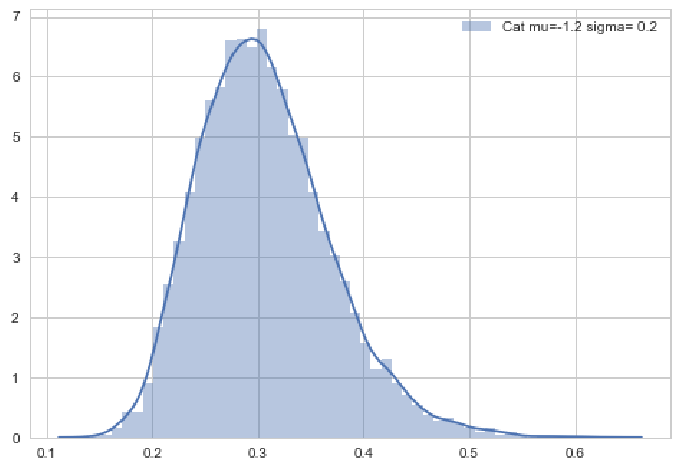
Distance Estimation

Bubble diameters (2*radius)



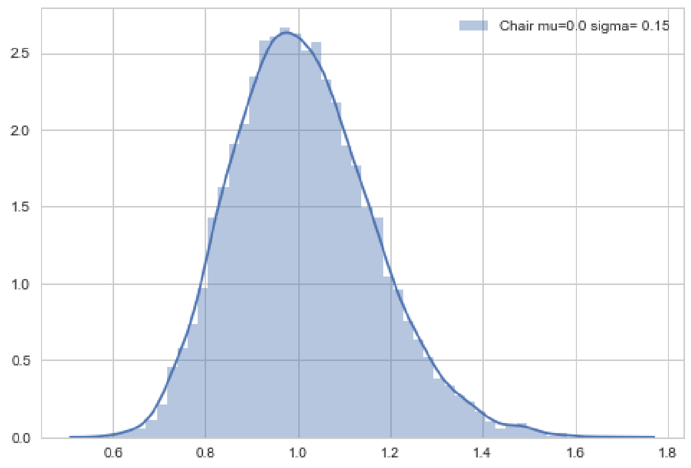


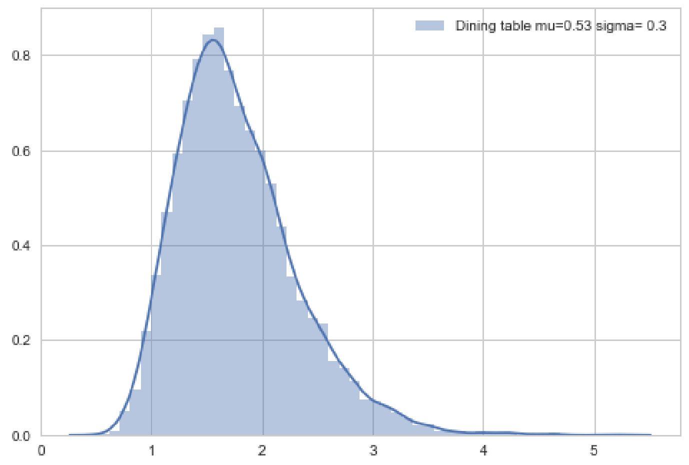


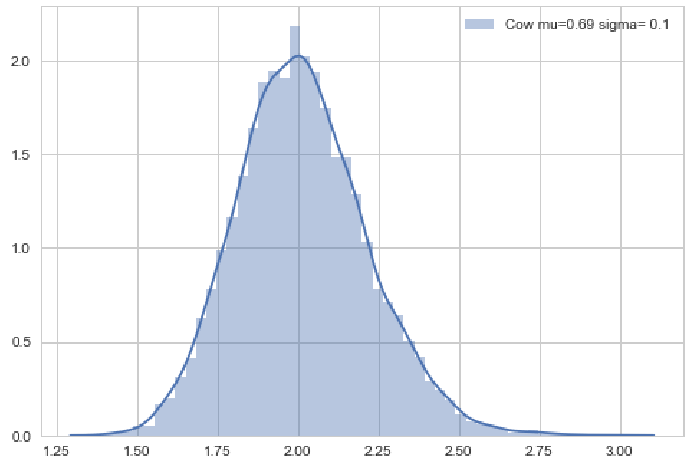


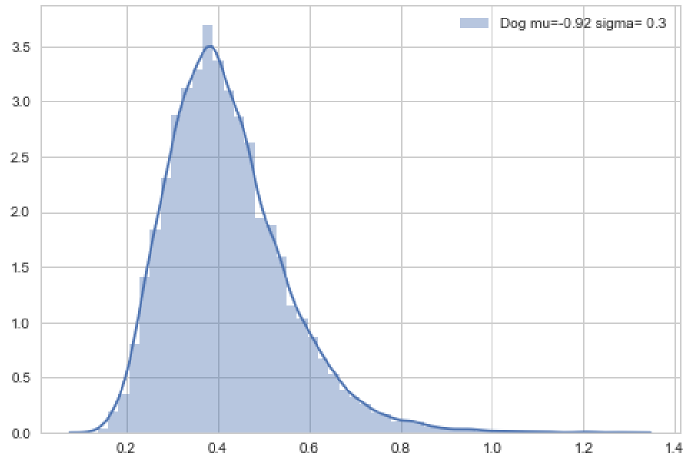
Distance Estimation

Bubble diameters (2*radius)



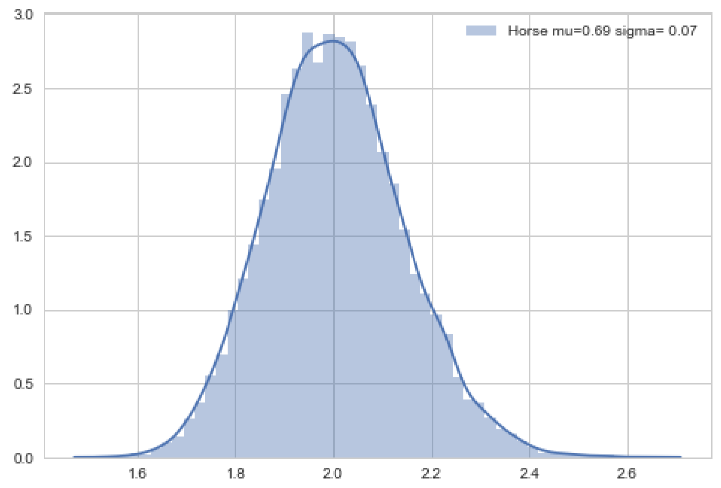


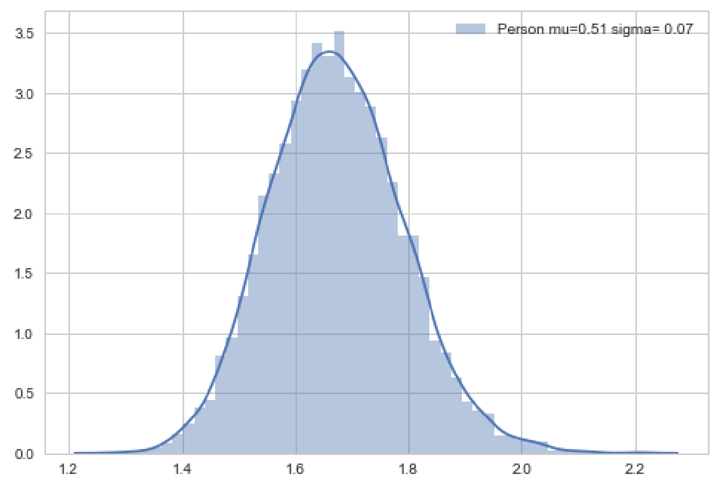


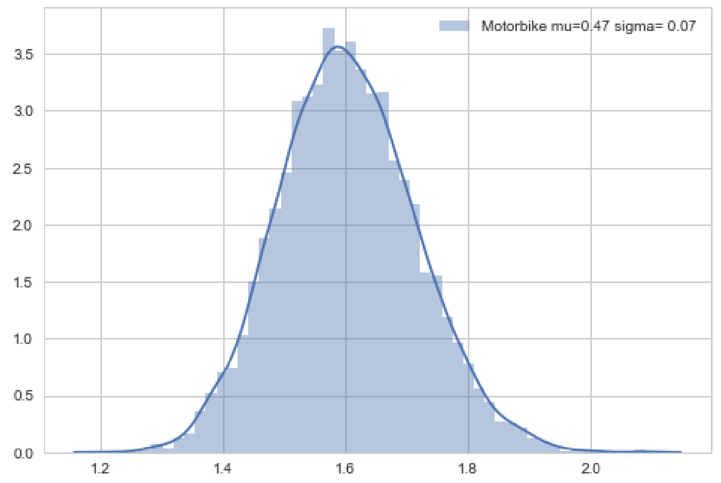


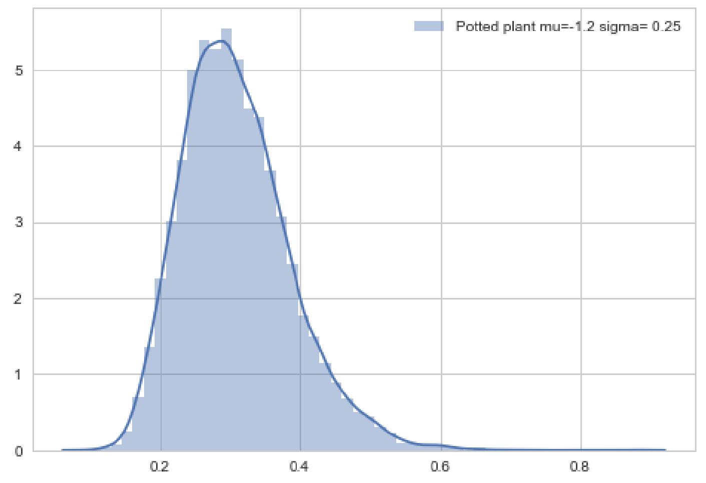
Distance Estimation

Bubble diameters (2*radius)



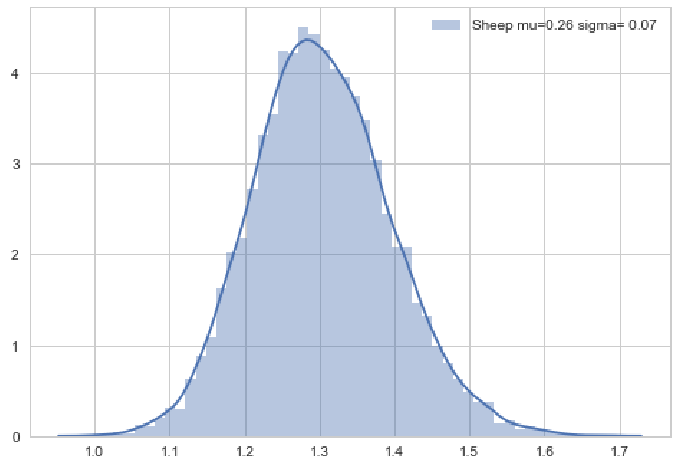


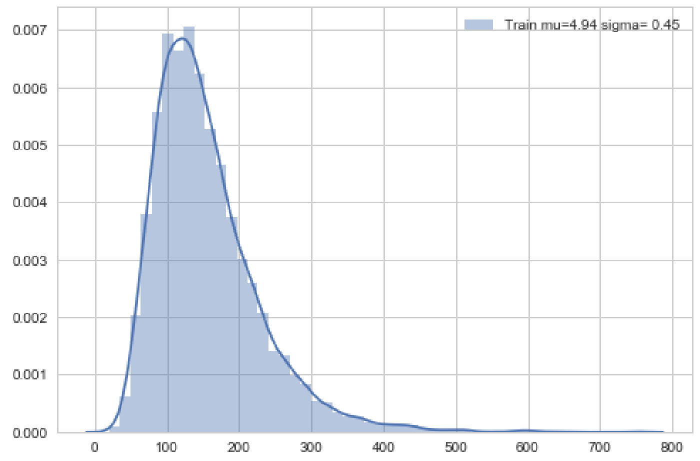


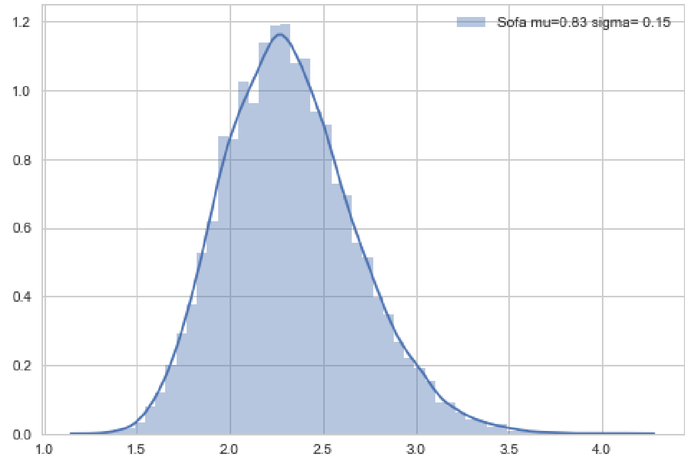


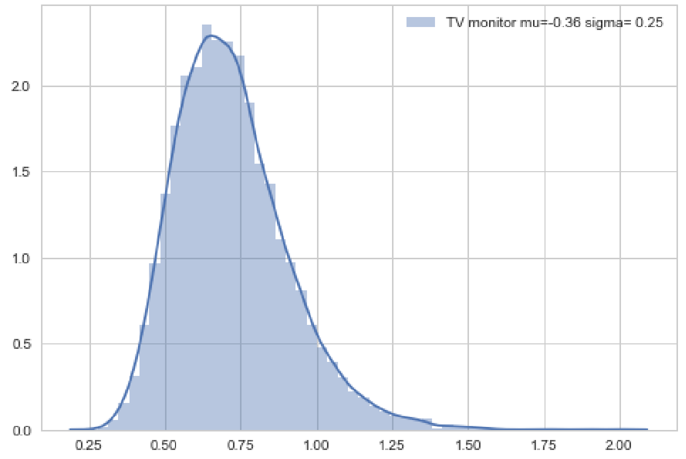
Distance Estimation

Bubble diameters (2*radius)



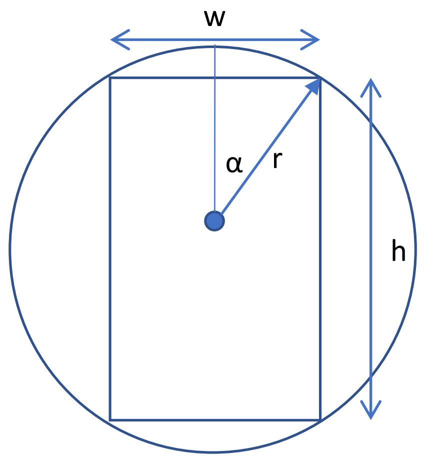






Distance Estimation

Radius of enclosing circle (pattern)



From bounding box coordinates to radius:

$$r = \sqrt{(\frac{w}{2})^2+(\frac{h}{2})^2}$$

$$h = (ymax - ymin) \quad c_y = (ymax + ymin)/2$$
$$\mathrm{w} = (xmax - xmin) \quad c_x = (xmax + xmin)/2$$

Distance Estimation

Distance estimation using pattern circle and body sphere

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

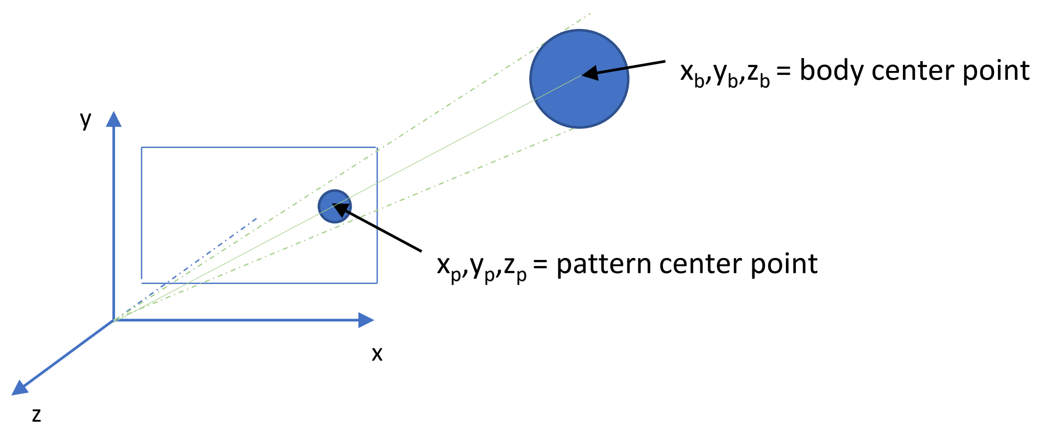
s_h = *sensor height (m)*
p_h = *image height (pixels)*
r_i = *pattern radius (pixels)*
r = *body radius (m), mean from class specific distribution*
f = *focal length (m)*
α = *altitude (rad)*
β = *azimuth (rad)*

Distance Estimation

Remarks:

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

3D Projection



x_b,y_b,z_b = body center point
y
x_p,y_p,z_p = pattern center point
x
z

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

$$(x_p, y_p, z_p) = \left(-\frac{s_w}{2} + px_p * \frac{s_w}{p_w},\ \ \frac{s_h}{2} - py_p * \frac{s_h}{p_h},\ \ -f\right)$$

Body center will be on the line:

$$(x_b, y_b, z_b) = t * (x_p, y_p, z_p)$$

Distance to the body is:

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

$$\alpha = \text{arc tan}(y_p/f)$$
$$\beta = \text{arc tan}(x_p/f)$$

s_w = *sensor width (m)*
s_h = *sensor height (m)*
p_w = *image width (pixels)*
p_h = *image height (pixels)*
r_i = *pattern radius (pixels)*
r = *body radius (m)*
f = *focal length (m)*
α = *altitude (rad)*

3D Projection

So: $t^2 * ({x_p}^2 + {y_p}^2 + {z_p}^2) = d^2$

Solving for t: $$t = \frac{d}{\sqrt{{x_p}^2 + {y_p}^2 + {z_p}^2}}$$

$(x_b, y_b, z_b) = t * (x_p, y_p, z_p)$

Where:

$$(x_p, y_p, z_p) = \left(-\frac{s_w}{2} + px_p * \frac{s_w}{p_w},\ \frac{s_h}{2} - py_p * \frac{s_h}{p_h},\ -f\right)$$

$$t = \frac{d}{\sqrt{{x_p}^2 + {y_p}^2 + {z_p}^2}}$$

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

3D Projection

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
r_i = pattern radius (pixels) = 100
r = body radius (m) = 1.0 m
f = focal length (m) = 0.050 m
x_p = 1200
y_p = 2000

$$(x_p, y_p, z_p) = \left(-\frac{s_w}{2} + xp * \frac{s_w}{p_w},\ \frac{s_h}{2} - yp * \frac{s_h}{p_h},\ -f\right)$$

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

α = arc tan(y_p/f) = 0.0445 β = arc tan(x_p/f) = -0.2374

$$d = \frac{f * h}{\cos(\alpha) * \cos(\beta) * h_i * s_h/p_h} = \frac{0.050 * 1}{\cos(0.0445) * \cos(-0.2374) * 100 * 0.0240/4912} = 105.39$$

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

$(x_b, y_b, z_b) = t * (x_p, y_p, z_p)$ = 2.0468e+03 * (-0.0121, 0.0022, -0.0500) = (-24.7593, 4.5602, -102.3389)

Body Filtering

- Enables prediction, including collision detection
- Second order model does not work, constant acceleration makes bodies bounce back or get enormous velocities
- In real world, constant acceleration for several (tens) of seconds is not common
- First order model works! (No wonder it’s popular in robotics…)
- When measurement is lost, the body is switched into constant velocity mode

Body Filtering

Body Kalman Filtering

Body center point location

State vector s:

$$s = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix}$$

where

(x, y, z) = location of the body center point

(v_x , v_y , v_z) = velocity of the body

State equation in differential form:

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

Body Filtering

State equation in difference form:

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

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where Δ is the time increment and ε Gaussian noise with covariance R:

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

Measurement equation:

$$z(k) = C * s(k) + \delta(k)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Where δ is Gaussian noise with covariance matrix Q:

$$Q = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 200 \end{bmatrix}$$

Body Filtering

Kalman filter initialization:

$$\mu(0) = \begin{bmatrix} x(0) \\ y(0) \\ z(0) \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where x(0), y(0), z(0) is the first location measurement.

$$\Sigma(0) = \begin{bmatrix} 100\,000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100\,000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100\,000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100\,000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100\,000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100\,000 \end{bmatrix}$$

Kalman filter update:

$$\mu_1(k) = A * \mu(k-1)$$

$$\Sigma_1(k) = A * \Sigma(k-1) * A^T + R$$

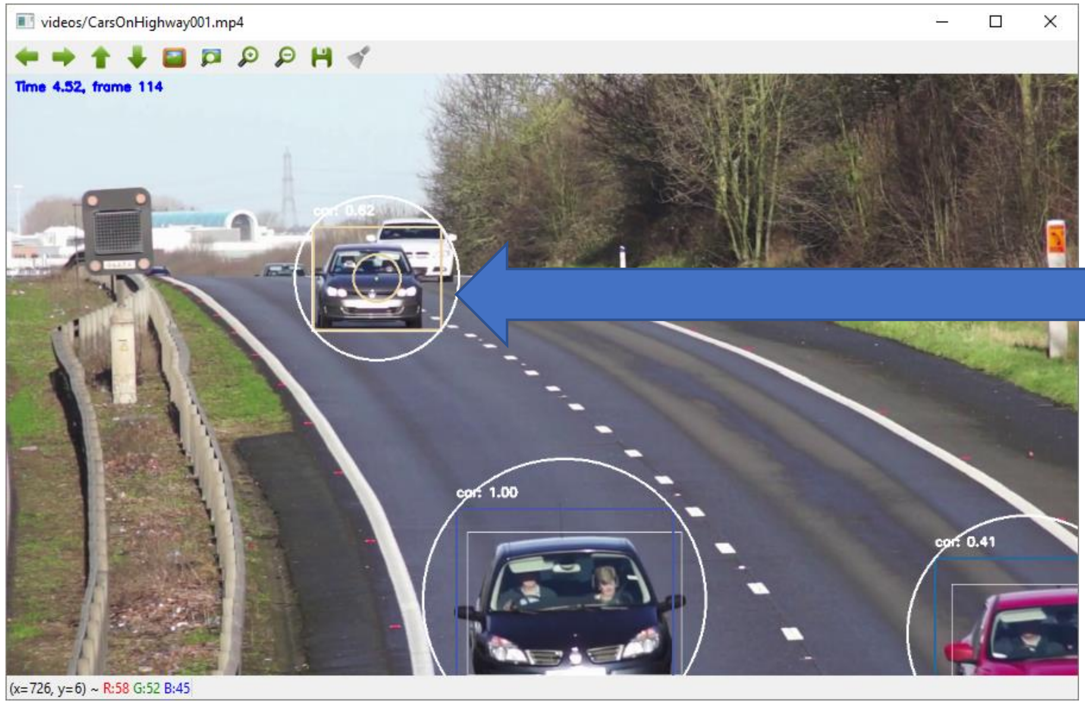
$$K(k) = \Sigma_1(k) * C^T (C * \Sigma_1(k) * C^T + Q)^{-1}$$

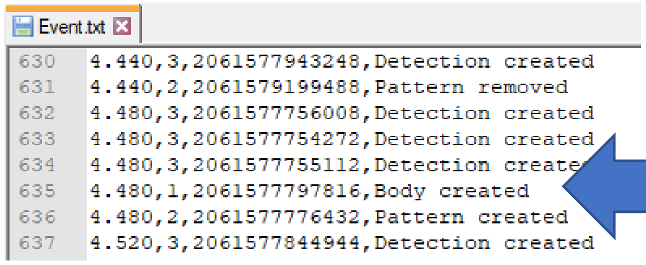
$$\mu(k) = \mu_1(k) + K(k) * (z(k) - C * \mu_1(k))$$

$$\Sigma(k) = (I - K(k) * C) * \Sigma_1(k)$$

Body Filtering

Example 1

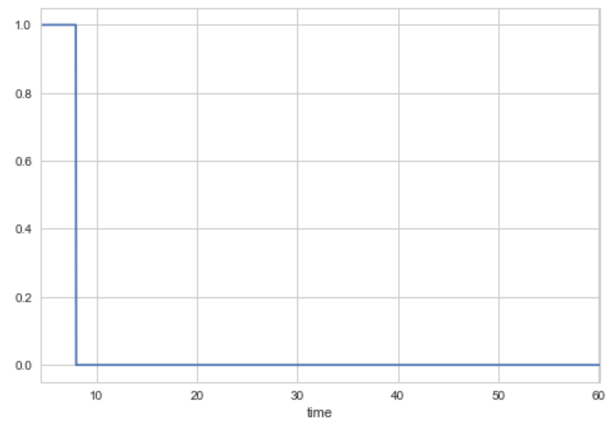



Event.txt

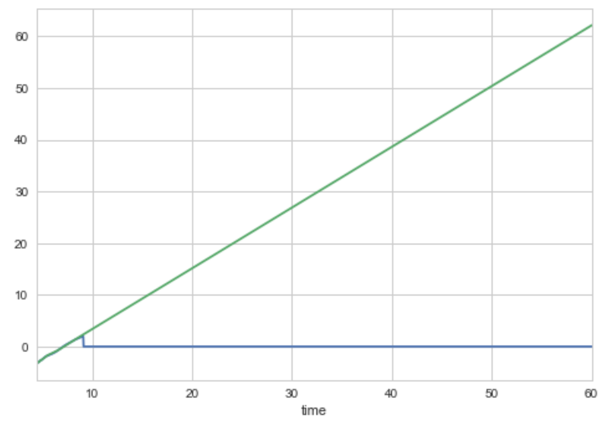
```
4.440,3,2061577943248,Detection created
4.440,2,2061579199488,Pattern removed
4.480,3,2061577756008,Detection created
4.480,3,2061577754272,Detection created
4.480,3,2061577755112,Detection create
4.480,1,2061577797816,Body created
4.480,2,2061577776432,Pattern created
4.520,3,2061577844944,Detection created
```

Body Filtering

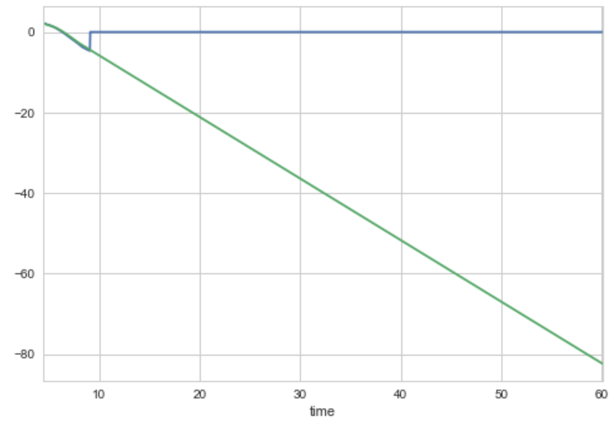
```
In [5]: data_one['status'].plot()

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x2e5b3fa4e10>
```



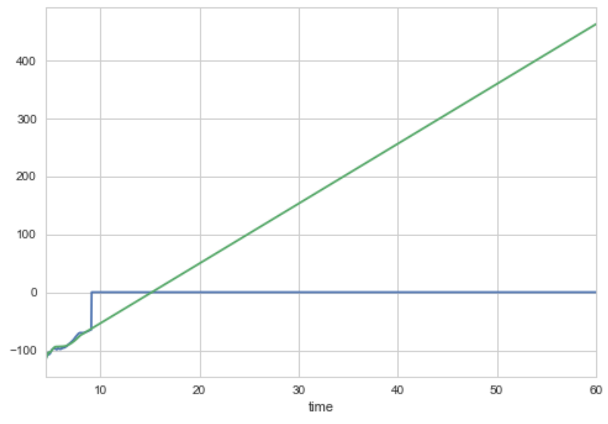
```
In [7]: data_one['x_pattern'].plot() # blue, measurement
        data_one['x'].plot() # green, filtered + predicted

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x2e5b4111c50>
```



```
In [8]: data_one['y_pattern'].plot() # blue, measurement
        data_one['y'].plot() # green, filtered + predicted

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x2e5b4125470>
```

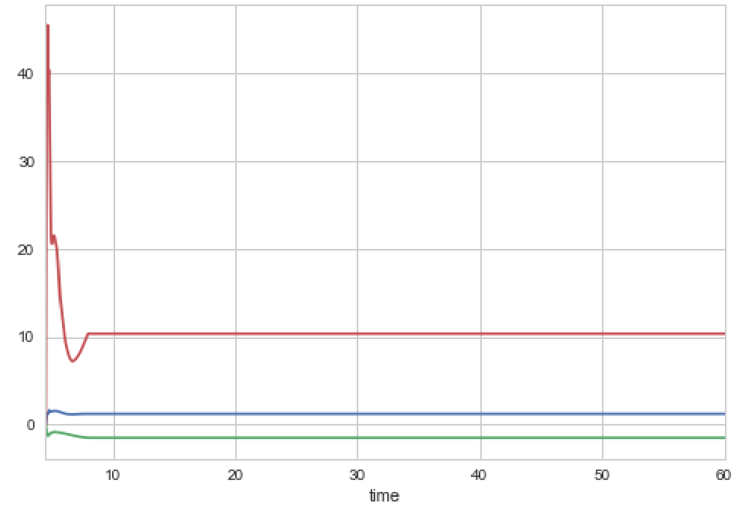


```
In [9]: data_one['z_pattern'].plot() # blue, measurement
        data_one['z'].plot() # green, filtered + predicted

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x2e5b41ab6d8>
```

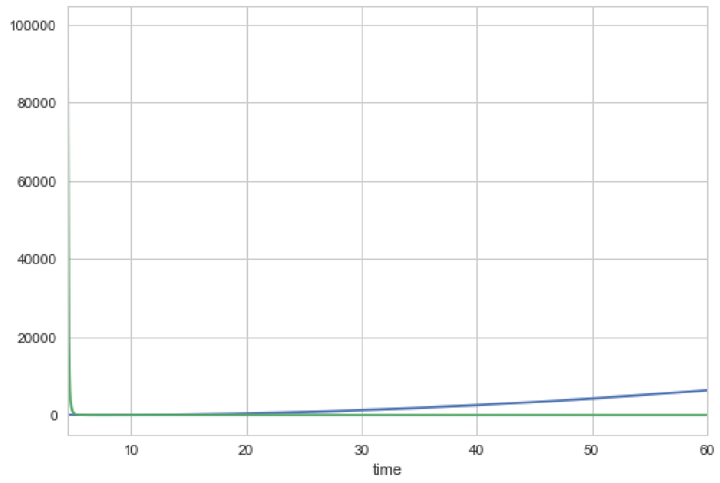


Body Filtering

```
In [10]: data_one['vx'].plot() # blue
         data_one['vy'].plot() # green
         data_one['vz'].plot() # red

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x2e5b424cac8>
```

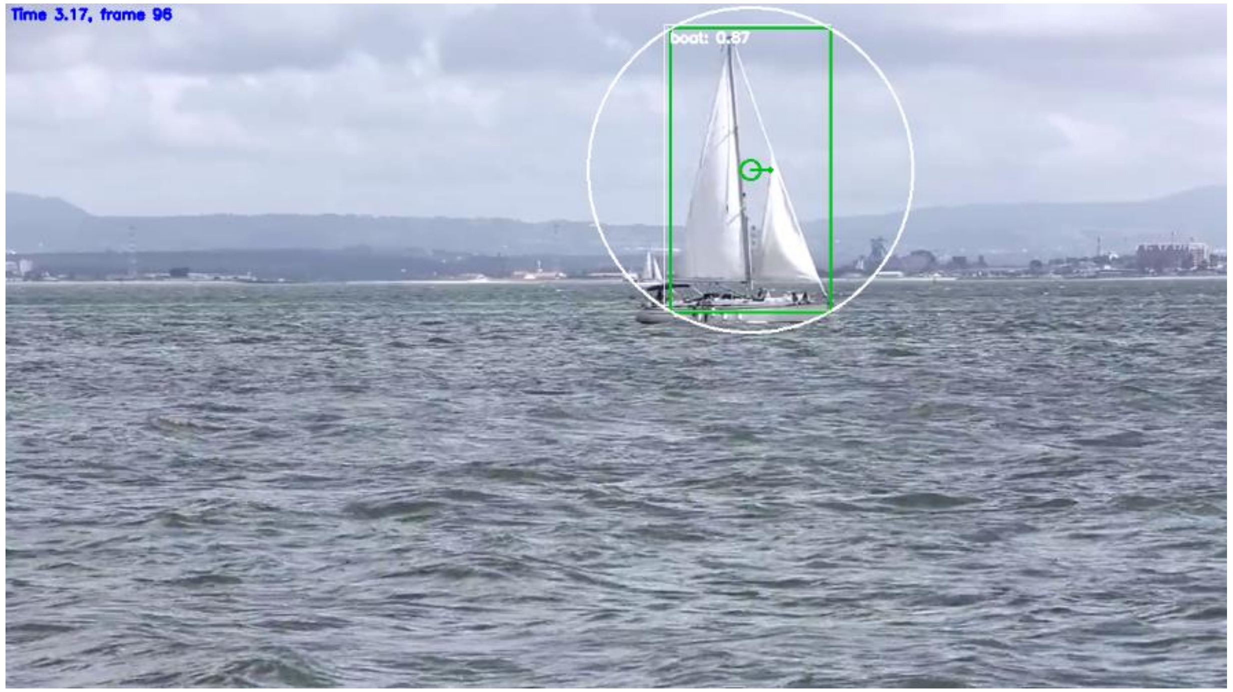


```
In [11]: data_one['sigma_00'].plot() # blue, x location variance
         data_one['sigma_33'].plot() # green, x velocity variance

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x2e5b55b6ac8>
```



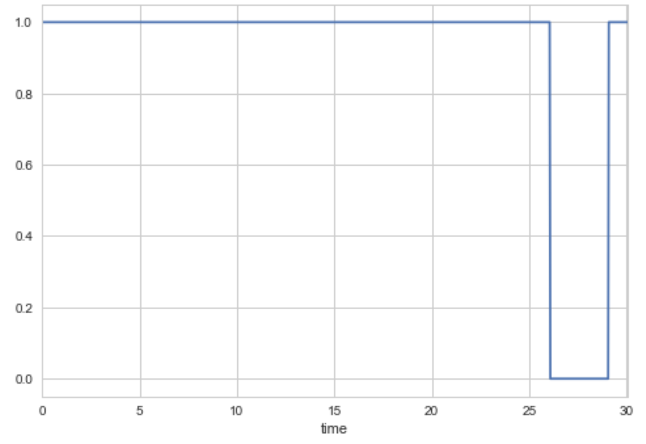
Body Filtering

Example 2

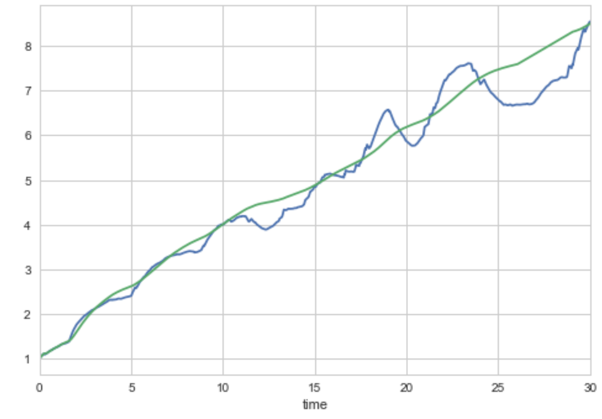


Body Filtering

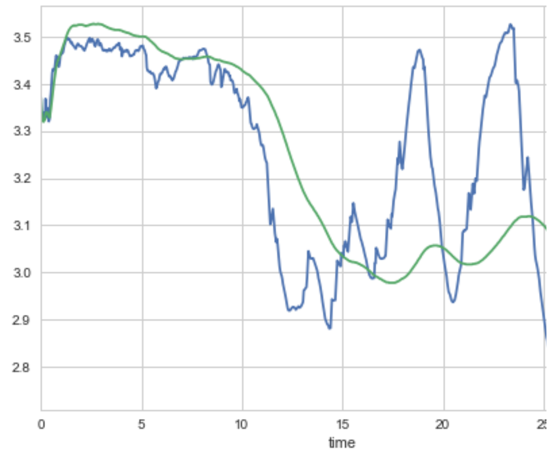
```
In [5]:  data_one['status'].plot()

Out[5]:  <matplotlib.axes._subplots.AxesSubplot at 0x1a52aca5438>
```



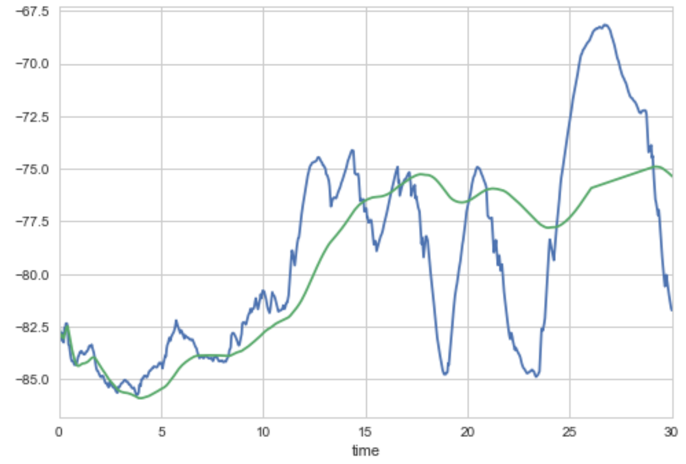
```
In [6]:  data_one['x_pattern'].plot() # blue, measurement
         data_one['x'].plot() # green, filtered + predicted

Out[6]:  <matplotlib.axes._subplots.AxesSubplot at 0x1a52bf61470>
```



```
In [7]:  data_one['y_pattern'].plot() # blue, measurement
         data_one['y'].plot() # green, filtered + predicted

Out[7]:  <matplotlib.axes._subplots.AxesSubplot at 0x1a52ad49
```

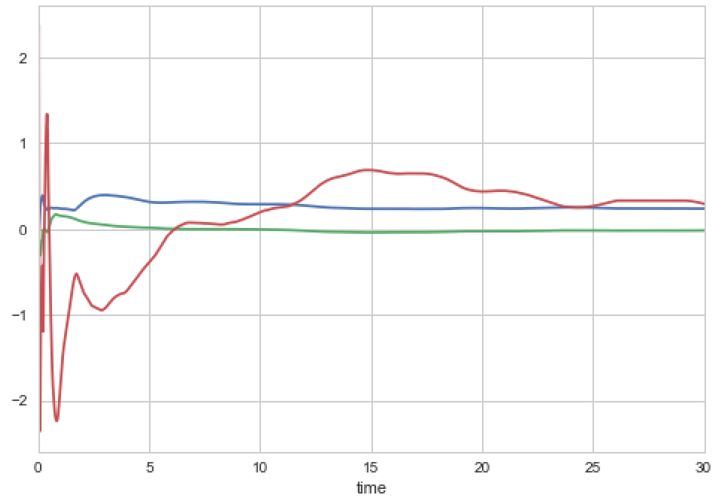


```
In [8]:  data_one['z_pattern'].plot() # blue, measurement
         data_one['z'].plot() # green, filtered + predicted

Out[8]:  <matplotlib.axes._subplots.AxesSubplot at 0x1a52c097cf8>
```

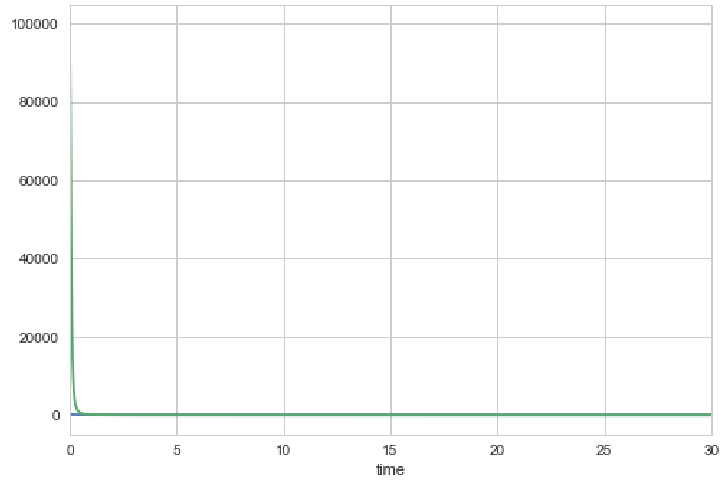


Body Filtering

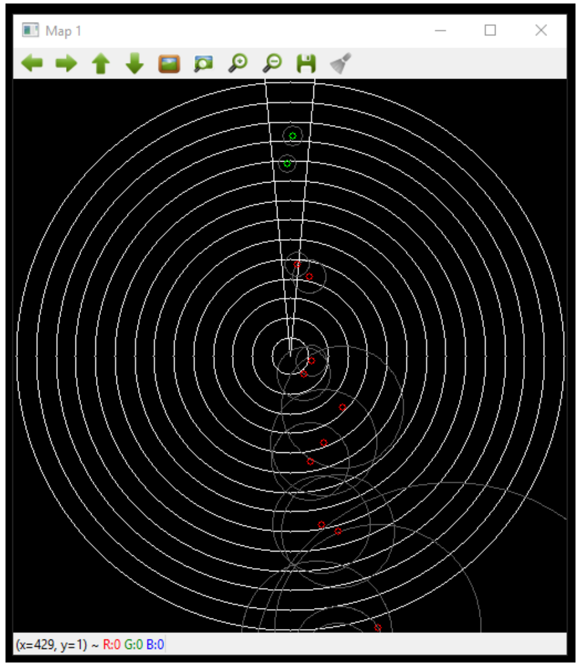
```
In [9]:  data_one['vx'].plot() # blue
         data_one['vy'].plot() # green
         data_one['vz'].plot() # red

Out[9]:  <matplotlib.axes._subplots.AxesSubplot at 0x1a52708b4a8>
```



```
In [10]:  data_one['sigma_00'].plot() # blue, x location variance
          data_one['sigma_33'].plot() # green, x velocity variance

Out[10]:  <matplotlib.axes._subplots.AxesSubplot at 0x1a52c184438>
```



Body Prediction



Kalman filter update:

$$\mu_1(k) = A * \mu(k-1)$$

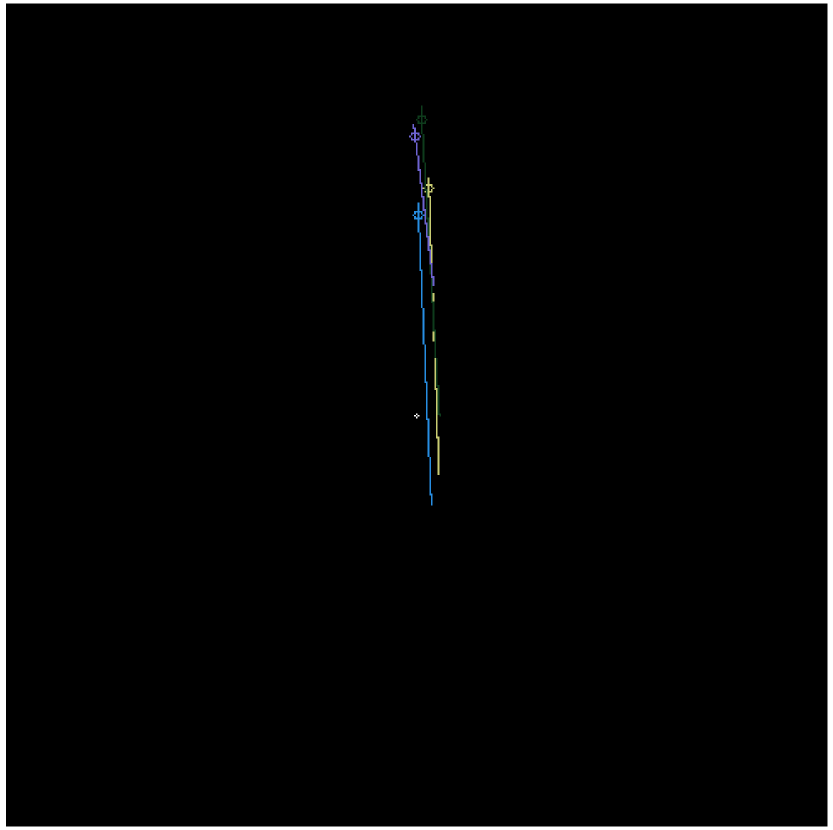
$$\Sigma_1(k) = A * \Sigma(k-1) * A^T + R$$

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mu = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} \qquad R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

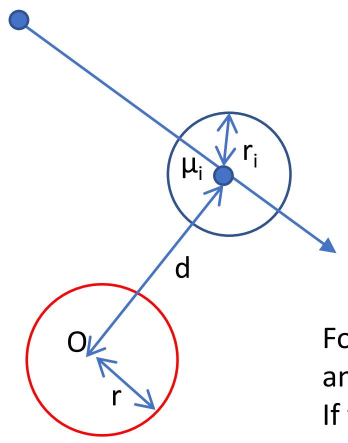
Δ has to be smaller than 1/fps. If fps=25, 1/fps = 0.04 sec. A car driving 120 km/h will proceed 1.33 meters and a collision with an observer might not be detected well enough. A value of Δ = 0.01 corresponds to the movement of 33 cm for an object moving at 120 km/h. This will generate 100 values per prediction per second predicted. Prediction is done once per second for 10 seconds horizon. This will generate 1000 values per prediction. Only the current prediction for an object is kept in memory. Objects are predicted separately.

Body Prediction


Example of predicted paths

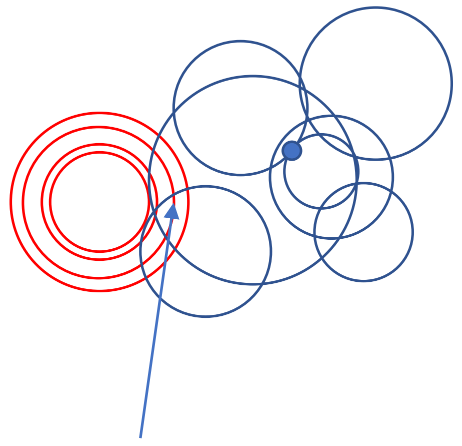
Collision Detection

Collision with the observer



nearest point μ_i on the predicted trajectory
t=t'
filter covariance estimate V_i

For each body i, a random vector t is sampled and distance to the observer calculated (center to center). If the distance is less than r_i+r, collision occurred. The proportion of collisions to all cases is the (maximum) probability estimate for the body/observer collision.



Collisions

$$t = \begin{bmatrix} x_i \\ y_i \\ z_i \\ r_i \\ r \end{bmatrix} \qquad \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \sim N(\mu_i, V_i)$$

From Kalman filter

$$r_i \sim lognormal(\mu_{r,i}, \sigma_{r,i})$$

$$r \sim lognormal(\mu_{r,o}, \sigma_{r,o})$$

Class specific

$$p_i = E\{C_i\} = \sum_{k=1}^{m} \frac{\delta(C_{i,k})}{m}$$

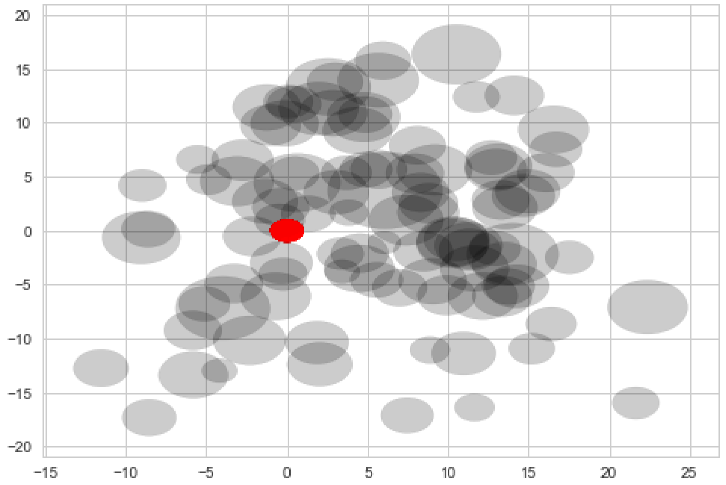
m=1000

Note! Collision might occur earlier. This algorithm estimates the **maximum** probability on the path. Should minimum time be used? Or both?

Collision Detection

Example

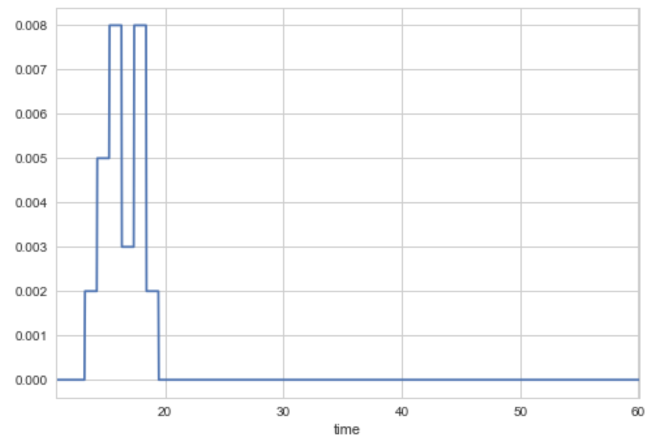
```
In [4]: data['collision_p'].argmax()

Out[4]: 2885

In [5]: data.iloc[2885].collision_p

Out[5]: 0.0080000000000000002

In [6]: mu=np.array([data.iloc[2885].c_x, data.iloc[2885].c_y, data.iloc[2885].c_z])
        mu

Out[6]: array([ 5.978, -6.696, -0.991])

In [9]: v

Out[9]: array([[ 58.632,   0.    ,   0.   ],
               [  0.   ,  58.632,   0.   ],
               [  0.   ,   0.    ,  58.632]])

In [10]: m=100

In [11]: loc_samples = np.random.multivariate_normal(mu, v, m)

In [13]: ri = np.random.lognormal(1.28, 0.25, m)/2 # car body sizes
         r = np.random.lognormal(0.51, 0.07, m)/2 # observer (person) body sizes

In [18]: circles(loc_samples[:,0],loc_samples[:,2], s=ri[:], c='black', alpha=0.2, edgecolor='none')
         circles(observer_loc[:],observer_loc[:], s=r[:], c='red', alpha=0.2, edgecolor='none')

Out[18]: <matplotlib.collections.PatchCollection at 0x20a35820ba8>
```



Collision Detection

Example

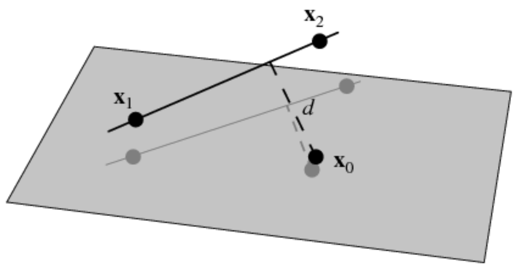
```
In [56]: data_one['collision_p'].plot()
Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x2ae866ea940>
```



Collision Detection

Open questions:

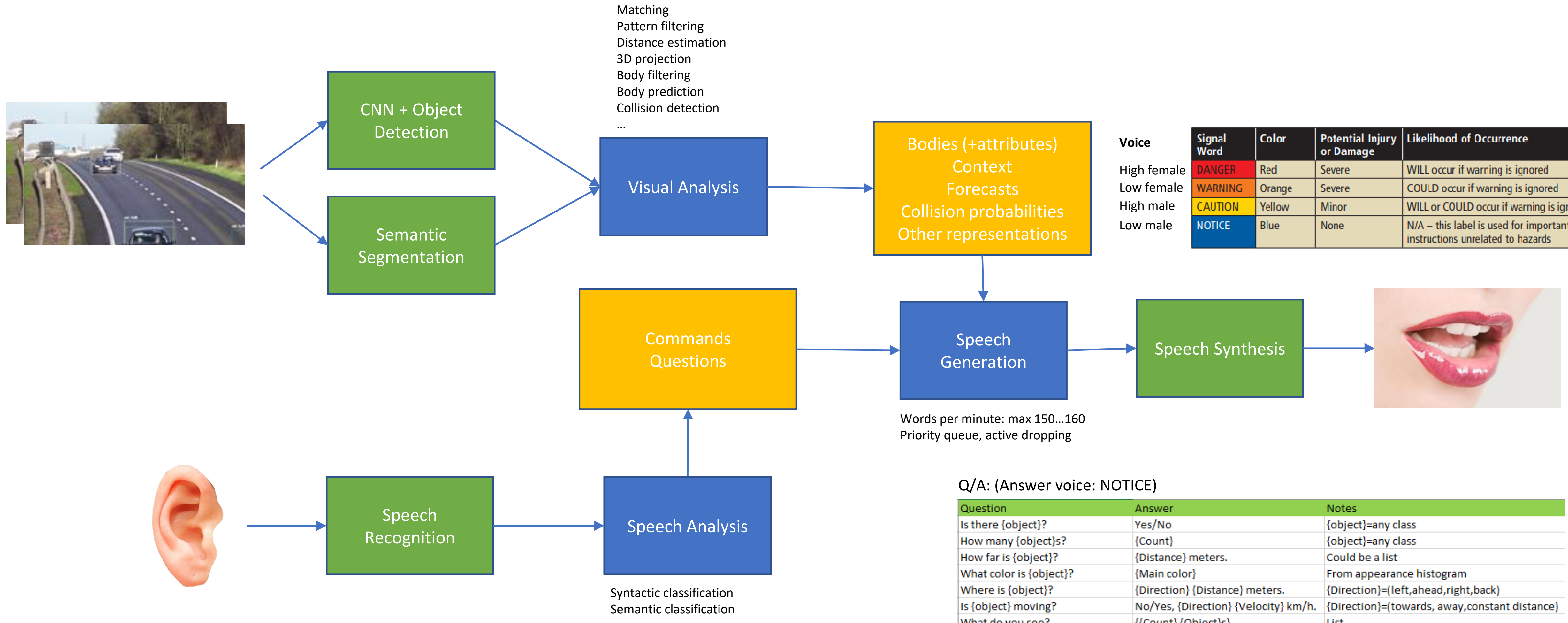
- Collisions between all bodies?
- Min time or max probability or both?
- More efficient sampling?



$$d = \frac{|(\mathbf{x}_2 - \mathbf{x}_1)\times(\mathbf{x}_1 - \mathbf{x}_0)|}{|\mathbf{x}_2 - \mathbf{x}_1|} = \frac{|(\mathbf{x}_0 - \mathbf{x}_1)\times(\mathbf{x}_0 - \mathbf{x}_2)|}{|\mathbf{x}_2 - \mathbf{x}_1|}$$

Note:
Min distance and corresponding time could be calculated without step by step prediction

The Big Picture



Voice	Signal Word	Color	Potential Injury or Damage	Likelihood of Occurrence
High female	DANGER	Red	Severe	WILL occur if warning is ignored
Low female	WARNING	Orange	Severe	COULD occur if warning is ignored
High male	CAUTION	Yellow	Minor	WILL or COULD occur if warning is ignored
Low male	NOTICE	Blue	None	N/A – this label is used for important instructions unrelated to hazards

Q/A: (Answer voice: NOTICE)

Question	Answer	Notes
Is there {object}?	Yes/No	{object}=any class
How many {object}s?	{Count}	{object}=any class
How far is {object}?	{Distance} meters.	Could be a list
What color is {object}?	{Main color}	From appearance histogram
Where is {object}?	{Direction} {Distance} meters.	{Direction}=(left,ahead,right,back)
Is {object} moving?	No/Yes, {Direction} {Velocity} km/h.	{Direction}=(towards, away,constant distance)
What do you see?	{{Count} {Object}s}	List
What is to {Direction} of {Object}?	{Object}	{Direction}=(left,right)
Is {Object} {Direction} {Object}?	Yes/No	{Direction}=(left,right,above,under)
Is {Object} free?	Yes/No	For example chair with/without other objects

Generated based on situation:

Signal word	Sentence	Notes
DANGER	{Object} will collide, move {Direction}.	{Object}=(bicycle,boat,bus,car,cow,horse,motorbike,person,train) {Direction}=(left,right,forward,backward)
WARNING	{Object} might collide, move {Direction}.	{Object}=(bicycle,boat,bus,car,cow,horse,motorbike,person,train) {Direction}=(left,right,forward,backward)
CAUTION	{Object} might collide, move {Direction}.	{Object}=(bird,cat,dog) {Direction}=(left,right,forward,backward)
WARNING	{Object} ahead, turn. Distance {Distance} meters.	{Object}=(chair,dining table,sofa)
NOTICE	{Object} is approaching. Distance {Distance} meters.	{Object}=(bicycle,bird,boat,bus,car,cat,cow,dog,horse,motorbike,person,train)
NOTICE	{Object} is leaving. Distance {Distance} meters.	{Object}=(bicycle,bird,boat,bus,car,cat,cow,dog,horse,motorbike,person,train)
NOTICE	{Caption}	
NOTICE	{Answer}	

Commands:

Command	Notes
Repeat answer	Answer to previous question is generated until stopped
Stop repeating	Stop answering
Be quiet	Output speech is off
Speak to me	Output speech is on

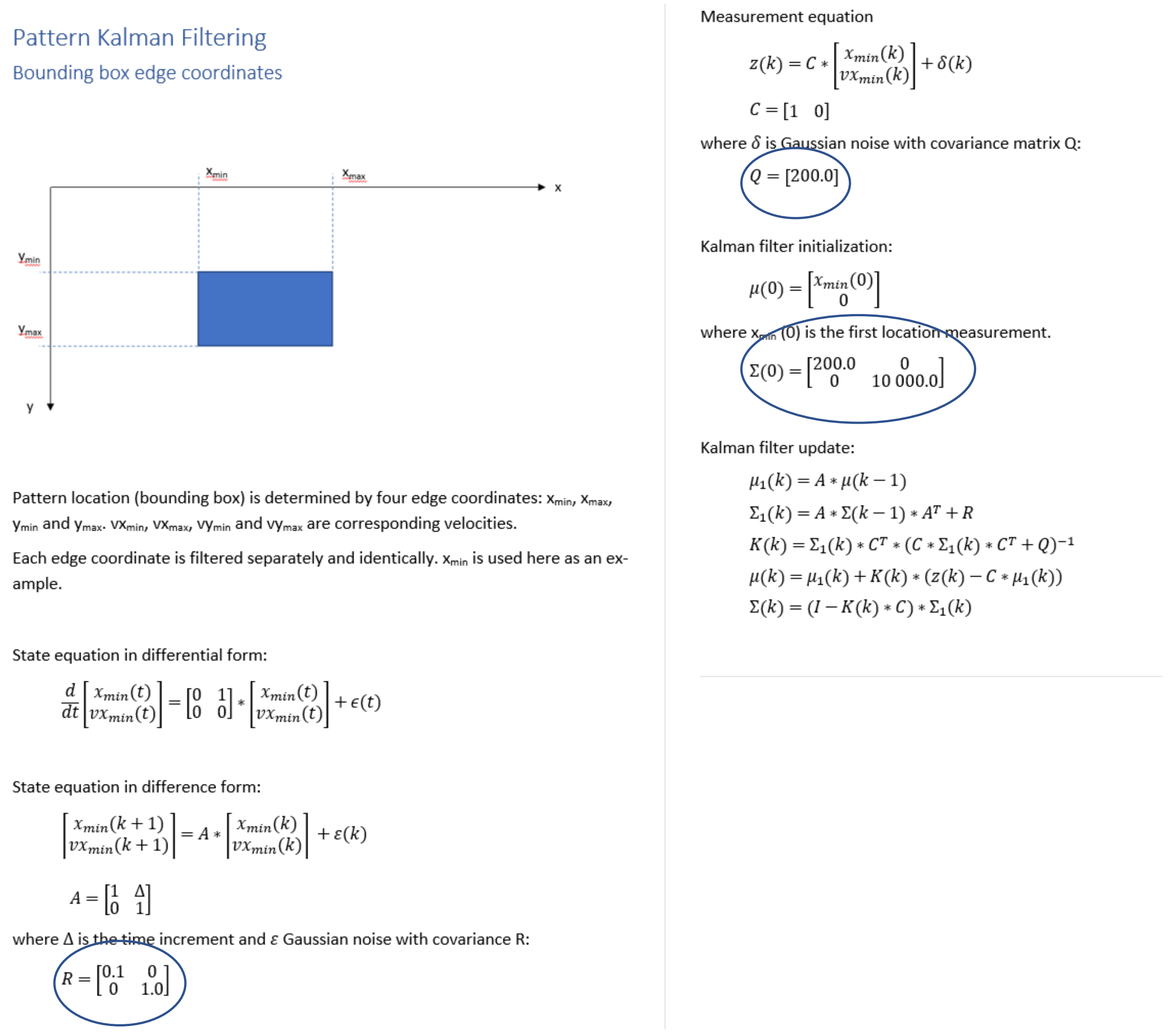
Next Steps

Next steps

- Kalman filter parameter adjustments (Q1)
- Dataset selection (Q1)
- Stereo vision (Q2)
- Camera yaw, pitch, roll estimation (Q2)
- Speech recognition (Q2)
- Semantic segmentation (Q2)
- Experiments in the wild (Q2)
- Paper (Q3)
- Speech analysis (Q3)
- Speech generation (Q3)
- Use cases (Q4)

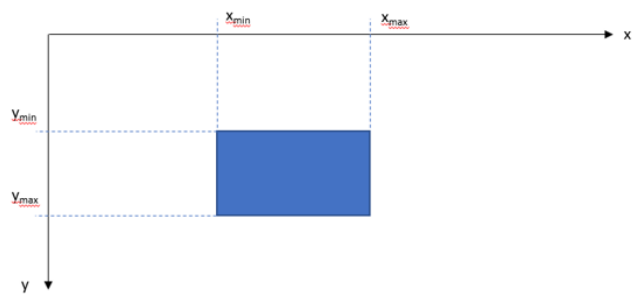
	2018				2019				2020				2021			
Methodology																
Preparation of research infra	▨	▨	▨													
Method survey	▨	▨	▨													
Building test cases				▨												
Testing and comparison					▨	▨	▨	▨	▨							
Prototype																
Definition										▨						
Planning											▨					
Implementation												▨	▨			
Testing and fixing														▨		
Method follow-up	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				
Writing thesis											▨	▨	▨	▨	▨	
Dissertation																▨

Kalman Filter Parameter Adjustments



Pattern Kalman Filtering

Bounding box edge coordinates



Pattern location (bounding box) is determined by four edge coordinates: x_{min}, x_{max}, y_{min} and y_{max}. vx_{min}, vx_{max}, vy_{min} and vy_{max} are corresponding velocities.

Each edge coordinate is filtered separately and identically. x_{min} is used here as an example.

State equation in differential form:

$$\frac{d}{dt}\begin{bmatrix} x_{min}(t) \\ vx_{min}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} * \begin{bmatrix} x_{min}(t) \\ vx_{min}(t) \end{bmatrix} + \epsilon(t)$$

State equation in difference form:

$$\begin{bmatrix} x_{min}(k+1) \\ vx_{min}(k+1) \end{bmatrix} = A * \begin{bmatrix} x_{min}(k) \\ vx_{min}(k) \end{bmatrix} + \varepsilon(k)$$

$$A = \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix}$$

where Δ is the time increment and ε Gaussian noise with covariance R:

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 1.0 \end{bmatrix}$$

Measurement equation

$$z(k) = C * \begin{bmatrix} x_{min}(k) \\ vx_{min}(k) \end{bmatrix} + \delta(k)$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

where δ is Gaussian noise with covariance matrix Q:

$$Q = [200.0]$$

Kalman filter initialization:

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

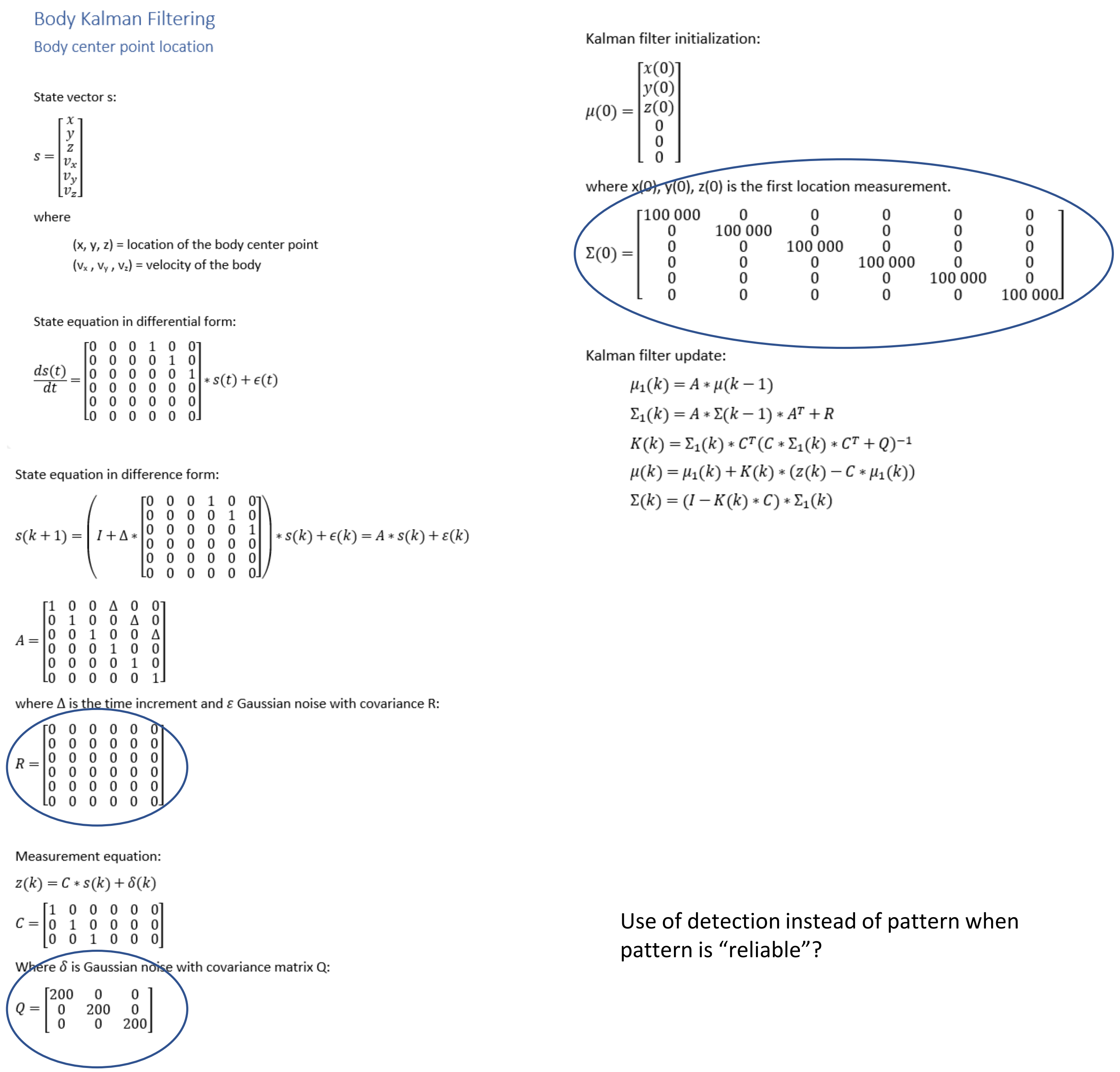
where $x_{min}(0)$ is the first location measurement.

$$\Sigma(0) = \begin{bmatrix} 200.0 & 0 \\ 0 & 10\,000.0 \end{bmatrix}$$

Kalman filter update:

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

Kalman Filter Parameter Adjustments



Body Kalman Filtering

Body center point location

State vector s:

$$s = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix}$$

where

(x, y, z) = location of the body center point

(v_x , v_y , v_z) = velocity of the body

State equation in differential form:

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

State equation in difference form:

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

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where Δ is the time increment and ε Gaussian noise with covariance R:

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

Measurement equation:

$$z(k) = C * s(k) + \delta(k)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Where δ is Gaussian noise with covariance matrix Q:

$$Q = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 200 \end{bmatrix}$$

Kalman filter initialization:

$$\mu(0) = \begin{bmatrix} x(0) \\ y(0) \\ z(0) \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where x(0), y(0), z(0) is the first location measurement.

$$\Sigma(0) = \begin{bmatrix} 100\,000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100\,000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100\,000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100\,000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100\,000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100\,000 \end{bmatrix}$$

Kalman filter update:

$$\mu_1(k) = A * \mu(k-1)$$

$$\Sigma_1(k) = A * \Sigma(k-1) * A^T + R$$

$$K(k) = \Sigma_1(k) * C^T (C * \Sigma_1(k) * C^T + Q)^{-1}$$

$$\mu(k) = \mu_1(k) + K(k) * (z(k) - C * \mu_1(k))$$

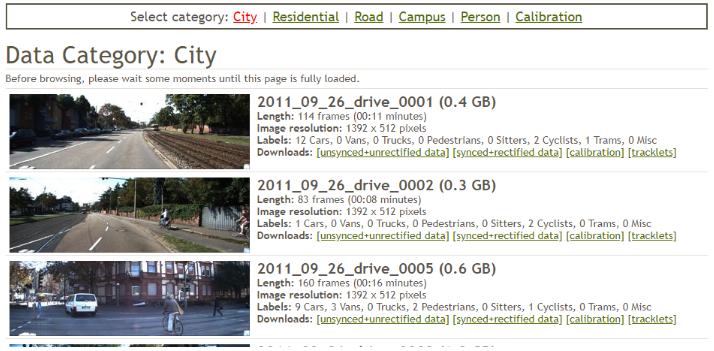
$$\Sigma(k) = (I - K(k) * C) * \Sigma_1(k)$$

Use of detection instead of pattern when pattern is “reliable”?

Dataset Selection

Specification:

- Video
- Stereo
- Distance information
- Outdoor + indoor
- Odometry



Select category: City | Residential | Road | Campus | Person | Calibration

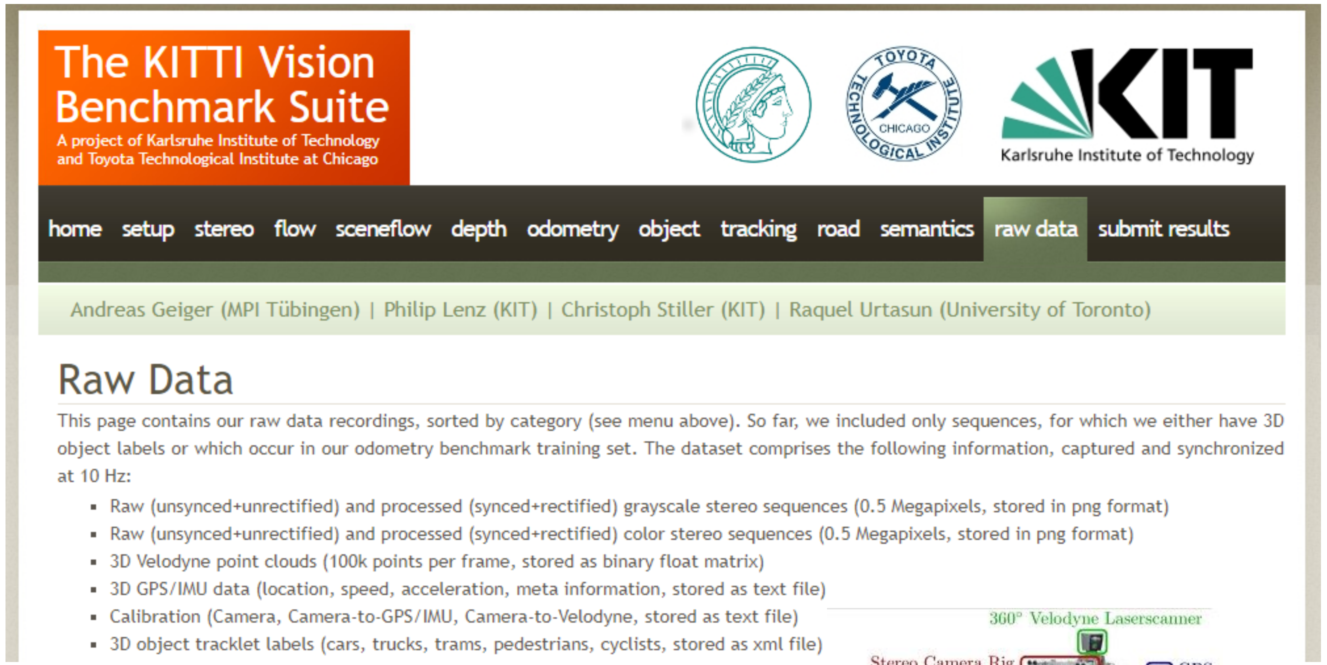
Data Category: City

Before browsing, please wait some moments until this page is fully loaded.

2011_09_26_drive_0001 (0.4 GB)
Length: 114 frames (00:11 minutes)
Image resolution: 1392 x 512 pixels
Labels: 12 Cars, 0 Vans, 0 Trucks, 0 Pedestrians, 0 Sitters, 2 Cyclists, 1 Trams, 0 Misc
Downloads: [unsynced+unrectified data] [synced+rectified data] [calibration] [tracklets]

2011_09_26_drive_0002 (0.3 GB)
Length: 83 frames (00:08 minutes)
Image resolution: 1392 x 512 pixels
Labels: 1 Cars, 0 Vans, 0 Trucks, 0 Pedestrians, 0 Sitters, 2 Cyclists, 0 Trams, 0 Misc
Downloads: [unsynced+unrectified data] [synced+rectified data] [calibration] [tracklets]

2011_09_26_drive_0005 (0.6 GB)
Length: 160 frames (00:16 minutes)
Image resolution: 1392 x 512 pixels
Labels: 9 Cars, 3 Vans, 0 Trucks, 2 Pedestrians, 0 Sitters, 1 Cyclists, 0 Trams, 0 Misc
Downloads: [unsynced+unrectified data] [synced+rectified data] [calibration] [tracklets]



The KITTI Vision Benchmark Suite

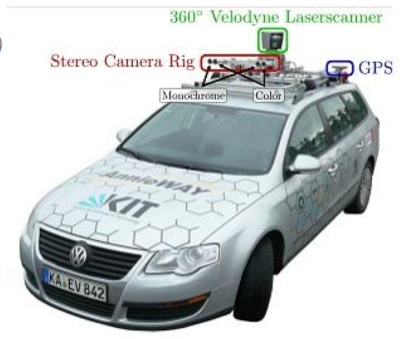
A project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago

Andreas Geiger (MPI Tübingen) | Philip Lenz (KIT) | Christoph Stiller (KIT) | Raquel Urtasun (University of Toronto)

Raw Data

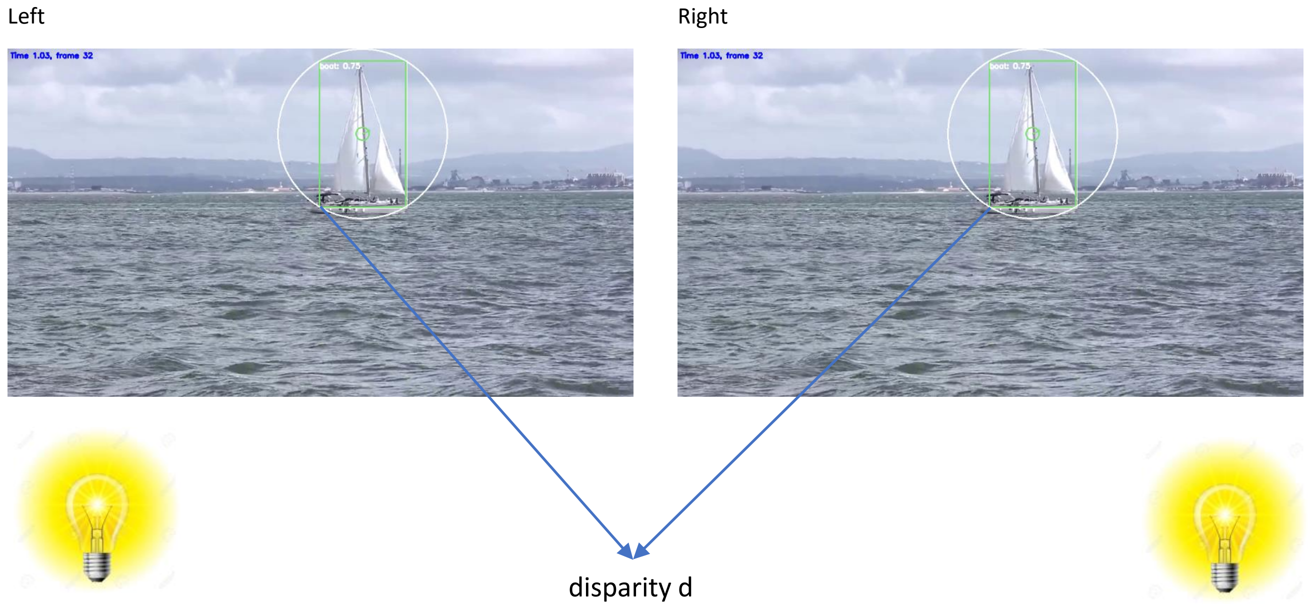
This page contains our raw data recordings, sorted by category (see menu above). So far, we included only sequences, for which we either have 3D object labels or which occur in our odometry benchmark training set. The dataset comprises the following information, captured and synchronized at 10 Hz:

- Raw (unsynced+unrectified) and processed (synced+rectified) grayscale stereo sequences (0.5 Megapixels, stored in png format)
- Raw (unsynced+unrectified) and processed (synced+rectified) color stereo sequences (0.5 Megapixels, stored in png format)
- 3D Velodyne point clouds (100k points per frame, stored as binary float matrix)
- 3D GPS/IMU data (location, speed, acceleration, meta information, stored as text file)
- Calibration (Camera, Camera-to-GPS/IMU, Camera-to-Velodyne, stored as text file)
- 3D object tracklet labels (cars, trucks, trams, pedestrians, cyclists, stored as xml file)

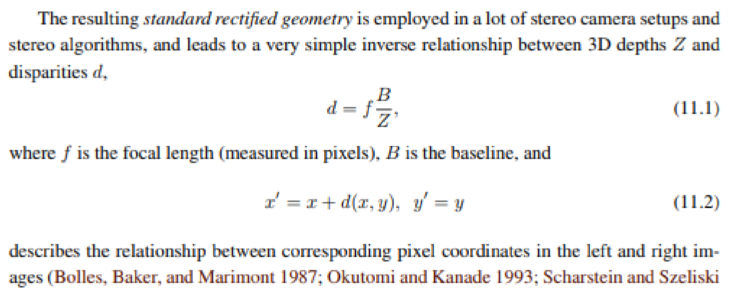


Open question: Indoor? Self generated?

Stereo Vision



Disparity for “free” using left and right bounding boxes!!!!!!!!!!



The resulting *standard rectified geometry* is employed in a lot of stereo camera setups and stereo algorithms, and leads to a very simple inverse relationship between 3D depths Z and disparities d,

$$d = f\frac{B}{Z}, \qquad (11.1)$$

where f is the focal length (measured in pixels), B is the baseline, and

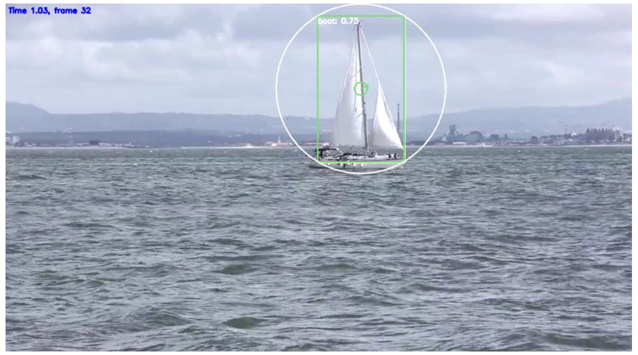
$$x' = x + d(x, y), \quad y' = y \qquad (11.2)$$

describes the relationship between corresponding pixel coordinates in the left and right images (Bolles, Baker, and Marimont 1987; Okutomi and Kanade 1993; Scharstein and Szeliski

If this works, stereo algorithm will be extremely fast and easy. Paper!!!

Rectification might still be needed?

Camera yaw, pitch, roll estimation

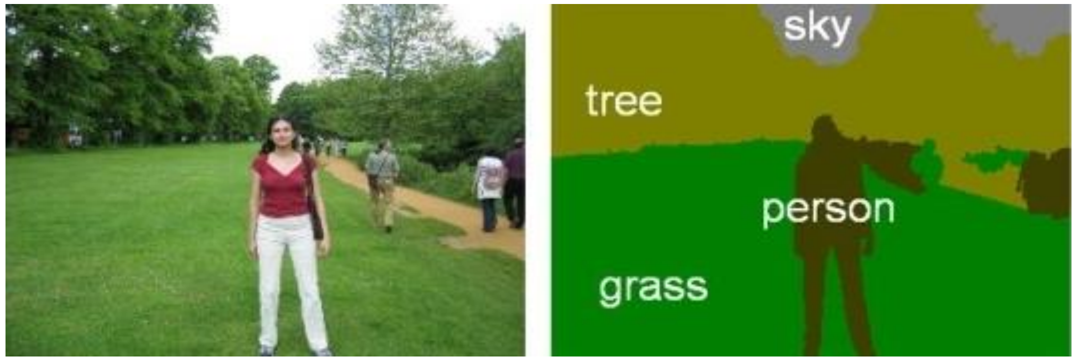


Can be estimated from background movement (average optical flow)?

Speech Recognition

Which off-the-shelf network to use? Must be light and fast.

Semantic Segmentation



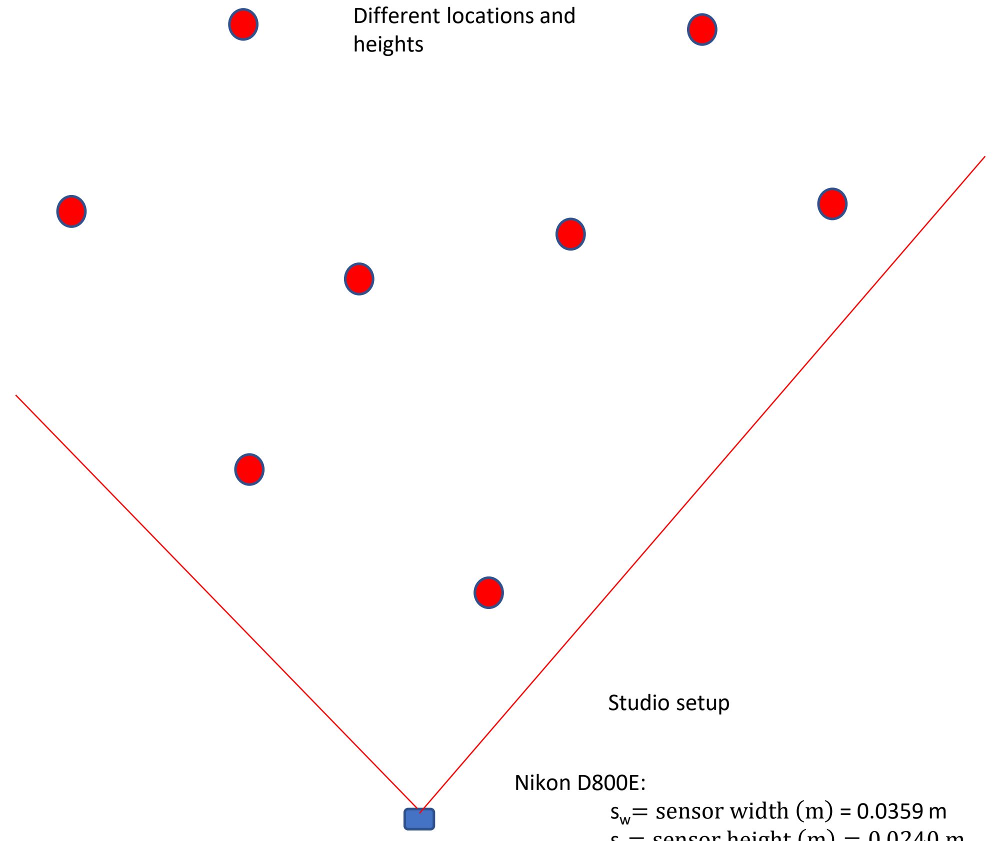
Why?

- Object context (”person on grass”)
- More exact localization for attribute extraction (”what color is person? red and white.”)

Which off-the-shelf network to use? Must be light and fast.

Experiments in the Wild

Distance estimation and 3D projection



Studio setup

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

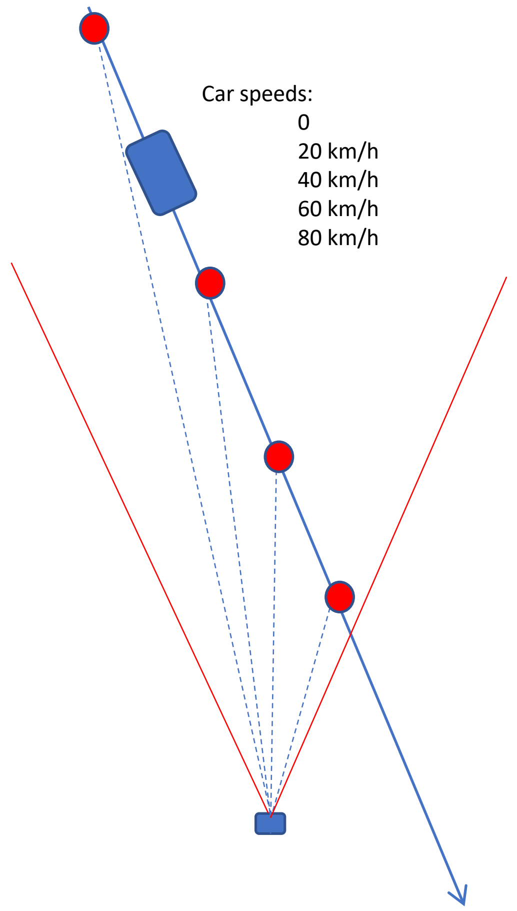
h = object height (m) = 1.5 m

f = focal length (m) = 0.020 m, varied

$\text{fov} = 2 * \text{atan}(\frac{s_w}{2*f}) = 83.81\,^{o}$

Experiments in the Wild

Body filtering



Car speeds:

- 0
- 20 km/h
- 40 km/h
- 60 km/h
- 80 km/h

Distances:

- 10 m
- 20 m
- 30 m
- 40 m
- 50 m
- 60 m
- 70 m
- 80 m
- 90 m
- 100 m

Outdoor setup

Laser distance measuring device would be nice…

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 = object height (m) = 1.5 m
- f = focal length (m) = 0.050 m
- fov = $2 * \text{atan}(\frac{s_w}{2*f})$ = 39.49 ^{o}

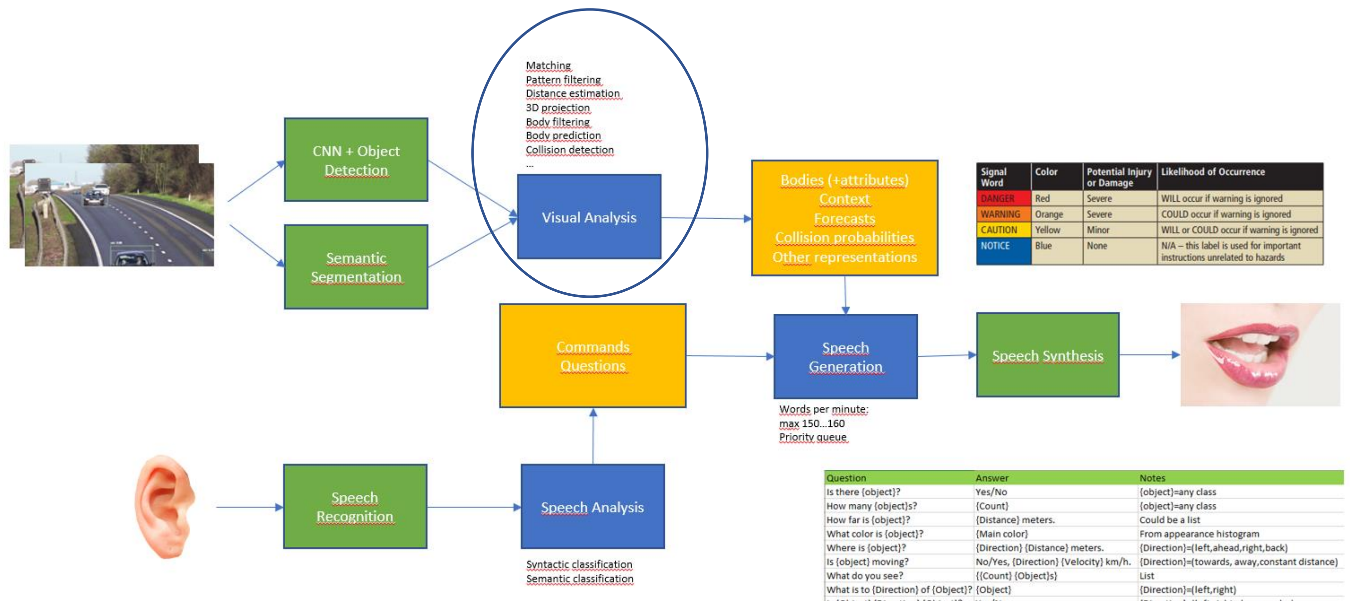
Paper

Image-Based Situation Awareness: Estimating Object Locations, Velocities and Collision Probabilities Using Object Detection

Contents:

???

Paper



Signal Word	Color	Potential Injury or Damage	Likelihood of Occurrence
DANGER	Red	Severe	WILL occur if warning is ignored
WARNING	Orange	Severe	COULD occur if warning is ignored
CAUTION	Yellow	Minor	WILL or COULD occur if warning is ignored
NOTICE	Blue	None	N/A – this label is used for important instructions unrelated to hazards

Question	Answer	Notes
Is there {object}?	Yes/No	{object}=any class
How many {object}s?	{Count}	{object}=any class
How far is {object}?	{Distance} meters.	Could be a list
What color is {object}?	{Main color}	From appearance histogram
Where is {object}?	{Direction} {Distance} meters.	{Direction}=(left,ahead,right,back)
Is {object} moving?	No/Yes, {Direction} {Velocity} km/h.	{Direction}=(towards, away,constant distance)
What do you see?	{{Count} {Object}s}	List
What is to {Direction} of {Object}?	{Object}	{Direction}=(left,right)
Is {Object} {Direction} {Object}?	Yes/No	{Direction}=(left,right,above,under)
Is {Object} free?	Yes/No	For example chair with/without other objects

Signal word	Sentence	Notes
DANGER	{Object} will collide, move {Direction}.	{Object}=(bicycle,boat,bus,car,cow,horse,motorbike,person,train) {Direction}=(left,right,forward,backward)
WARNING	{Object} might collide, move {Direction}.	{Object}=(bicycle,boat,bus,car,cow,horse,motorbike,person,train) {Direction}=(left,right,forward,backward)
CAUTION	{Object} might collide, move {Direction}.	{Object}=(bird,cat,dog) {Direction}=(left,right,forward,backward)
WARNING	{Object} ahead, turn. Distance {Distance} meters.	{Object}=(chair,dining table,sofa)
NOTICE	{Object} is approaching. Distance {Distance} meters.	{Object}=(bicycle,bird,boat,bus,car,cat,cow,dog,horse,motorbike,person,train)
NOTICE	{Object} is leaving. Distance {Distance} meters.	{Object}=(bicycle,bird,boat,bus,car,cat,cow,dog,horse,motorbike,person,train)
NOTICE	{Caption}	
NOTICE	{Answer}	

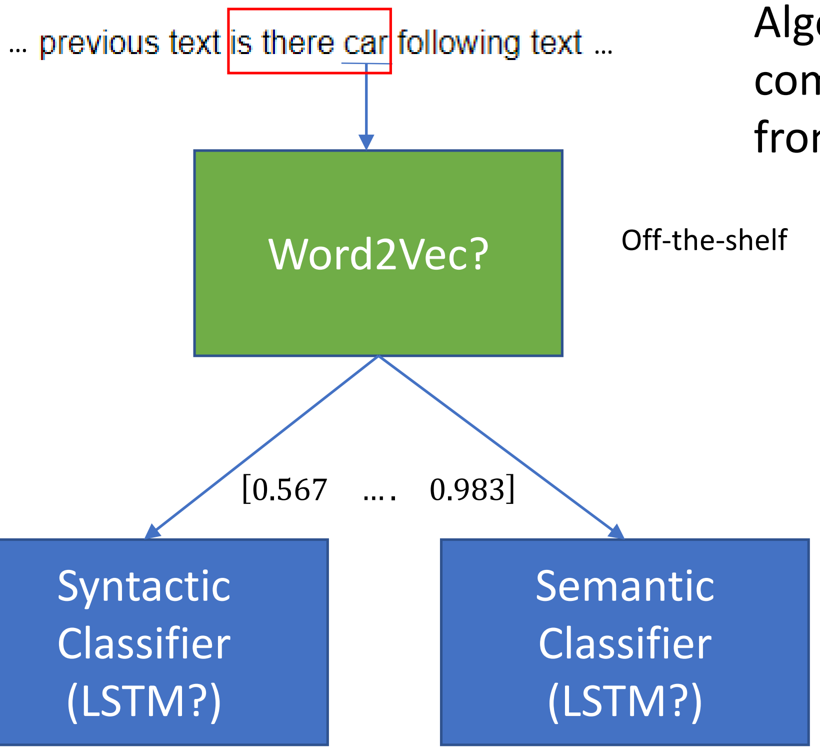
Command	Notes
Repeat answer	Answer to previous question is generated until stopped
Stop repeating	Stop answering
Be quiet	Output speech is off
Speak to me	Output speech is on

Where to publish?

- CVPR 2019 (6/2019, Long Beach, deadline 11/2018)
- ICCV 2019 (29.10.-3.11.2019, Seoul, deadline 1/2019)

Speech Analysis

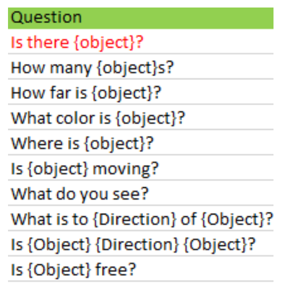
Algorithm has to pick commands and questions from word flow



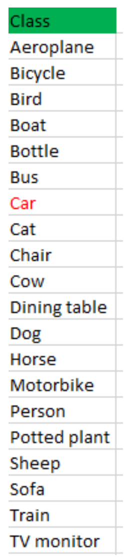
... previous text is there car following text ...
Word2Vec?
[0.567 .... 0.983]
Syntactic Classifier (LSTM?)
Semantic Classifier (LSTM?)

Off-the-shelf

These two networks need to be taught - data generation required



Question
Is there {object}?
How many {object}s?
How far is {object}?
What color is {object}?
Where is {object}?
Is {object} moving?
What do you see?
What is to {Direction} of {Object}?
Is {Object} {Direction} {Object}?
Is {Object} free?



Class
Aeroplane
Bicycle
Bird
Boat
Bottle
Bus
Car
Cat
Chair
Cow
Dining table
Dog
Horse
Motorbike
Person
Potted plant
Sheep
Sofa
Train
TV monitor

Speech Generation

?

Probably mostly rule-based. ML algorithms?

Use Cases

Example: Search chair

User	Application	Notes
speak to me	hello	
what do you see	one chair one dining table	
where is chair	right two point five meters	user turns right and steps forward
is chair free	yes	no objects detected nearby
repeat answer	ahead two point five meters	user steps forward
	ahead two meters	user steps forward
	ahead one point six meters	user steps forward
	ahead zero point seven meters	user stops and finds chair
stop repeating		
be quiet		

Example: Give way

User	Application	Notes
	danger person will collide move left	DANGER voice, person steps left
	danger person will collide move left	DANGER voice, person steps left
	warning person might collide move left	WARNING voice, person steps left
	notice no collision	NOTICE voice

Examples here are very preliminary. ☺ We need tens of real use cases. Näkövammaisten Liitto?

Final commands and question alternatives are derived from use cases.

Used also as test cases.

Discussion

Thank you!

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