université de BORDEAUX

Model Free Target Tracking

TRDP Part II

Supervisors: Juan Carlos San Miguel(Proponent), Aurélie Bugeau, Miklós Koller

GROUP 7

Eduardo Daniel Bravo Solis

Frances Ryan

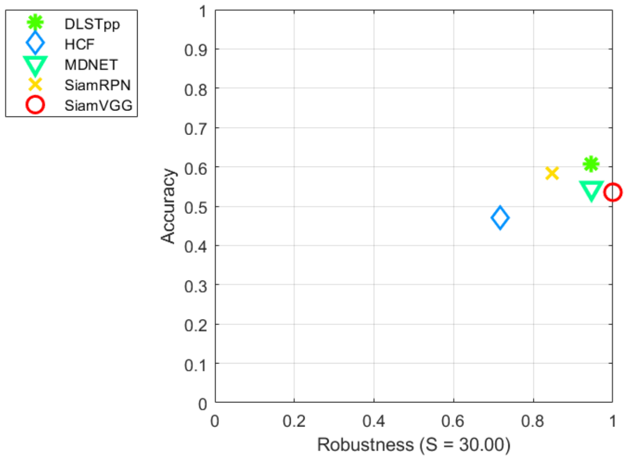
Image Processing AND Computer Vision

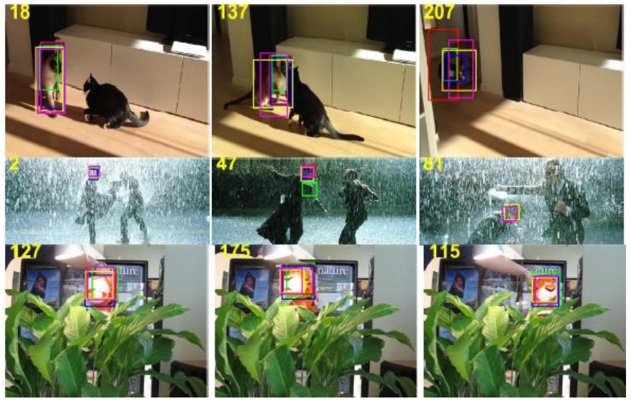
Co-funded by the
Erasmus+ Programme
of the European Union

Previous work - TRDP semester 1

1. Understanding of the basic principles of CNNs
2. Detailed review of two tracking algorithms using CNNs
   1. Hierarchical convolutional features for visual tracking(HCF)
   2. Learning Multi-Domain Convolutional Neural Networks for Visual Tracking
3. Code was adapted for integration to the VOT2018 evaluation framework

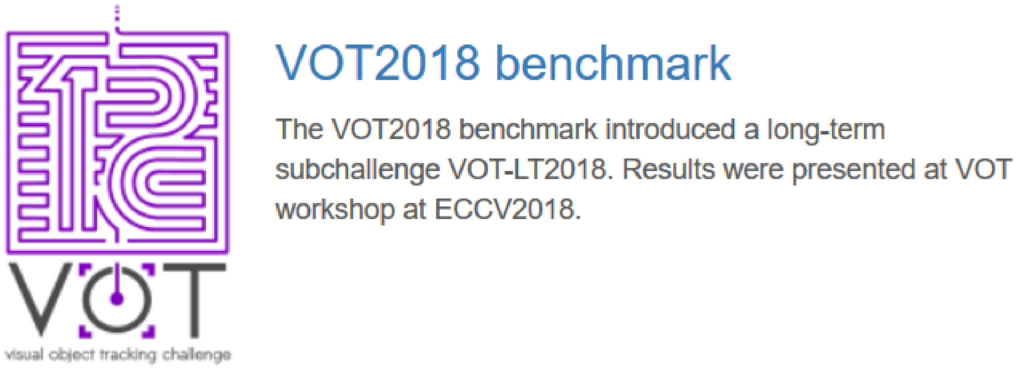
Task	Feb	March	April	May
Basics of CNNs	🔍			
Review Papers of Two Trackers		🔍	🔍	
Setup and Review Code of Trackers			🔍	
Setup and evaluation with VOT benchmark				🔍
Write-up				🔍





Problem to solve

- VOT benchmark requires particular **software**, long **configurations** and certain **hardware** to work properly.



VOT2018 benchmark

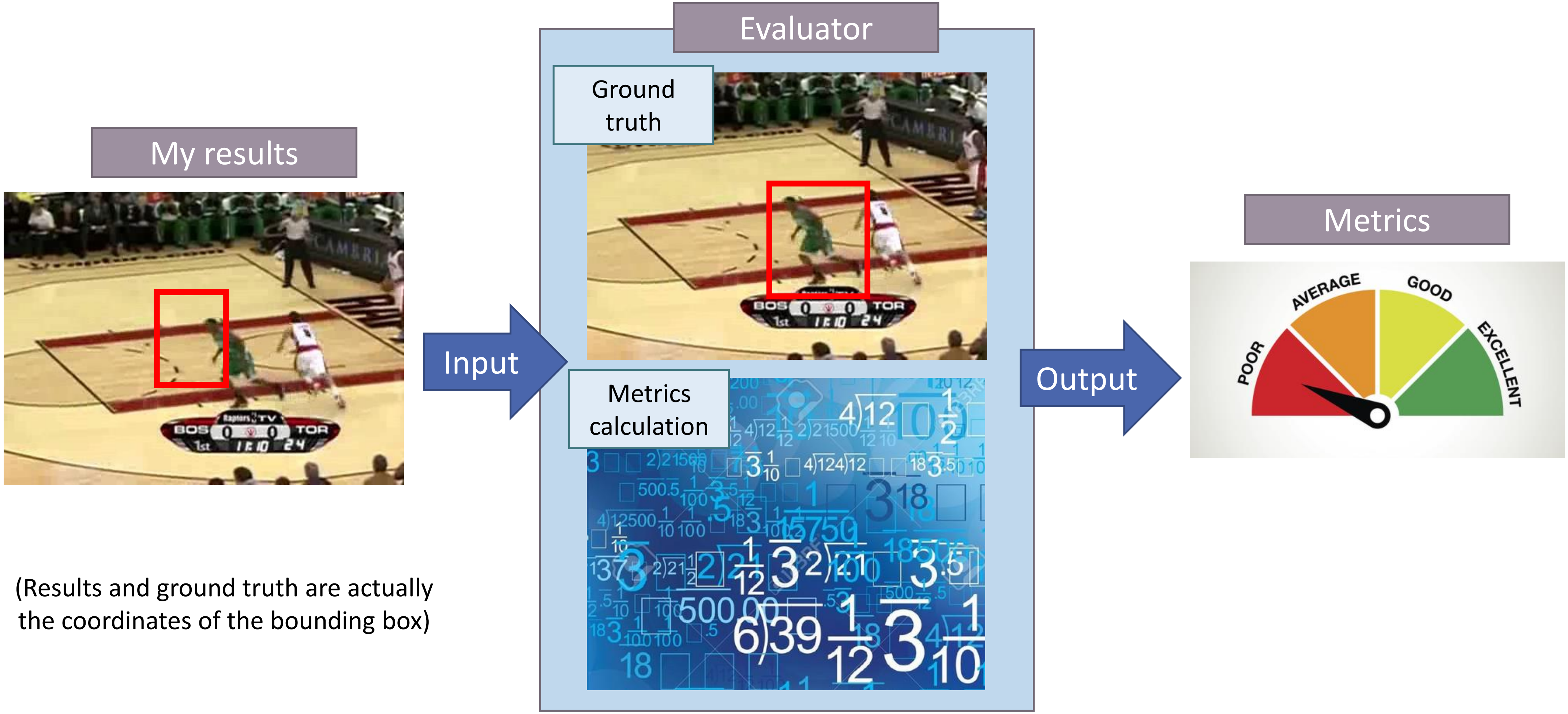
The VOT2018 benchmark introduced a long-term subchallenge VOT-LT2018. Results were presented at VOT workshop at ECCV2018.

- Other options are not intuitive and does not offer user-friendly features.

http://www.votchallenge.net/

How good is my video tracking algorithm?

- Run results through an evaluator that contains the ground truth.



My results

Input

Evaluator

Ground truth

Metrics calculation

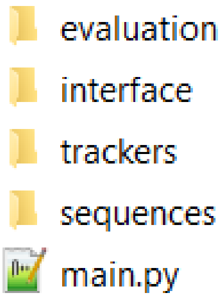
Output

Metrics

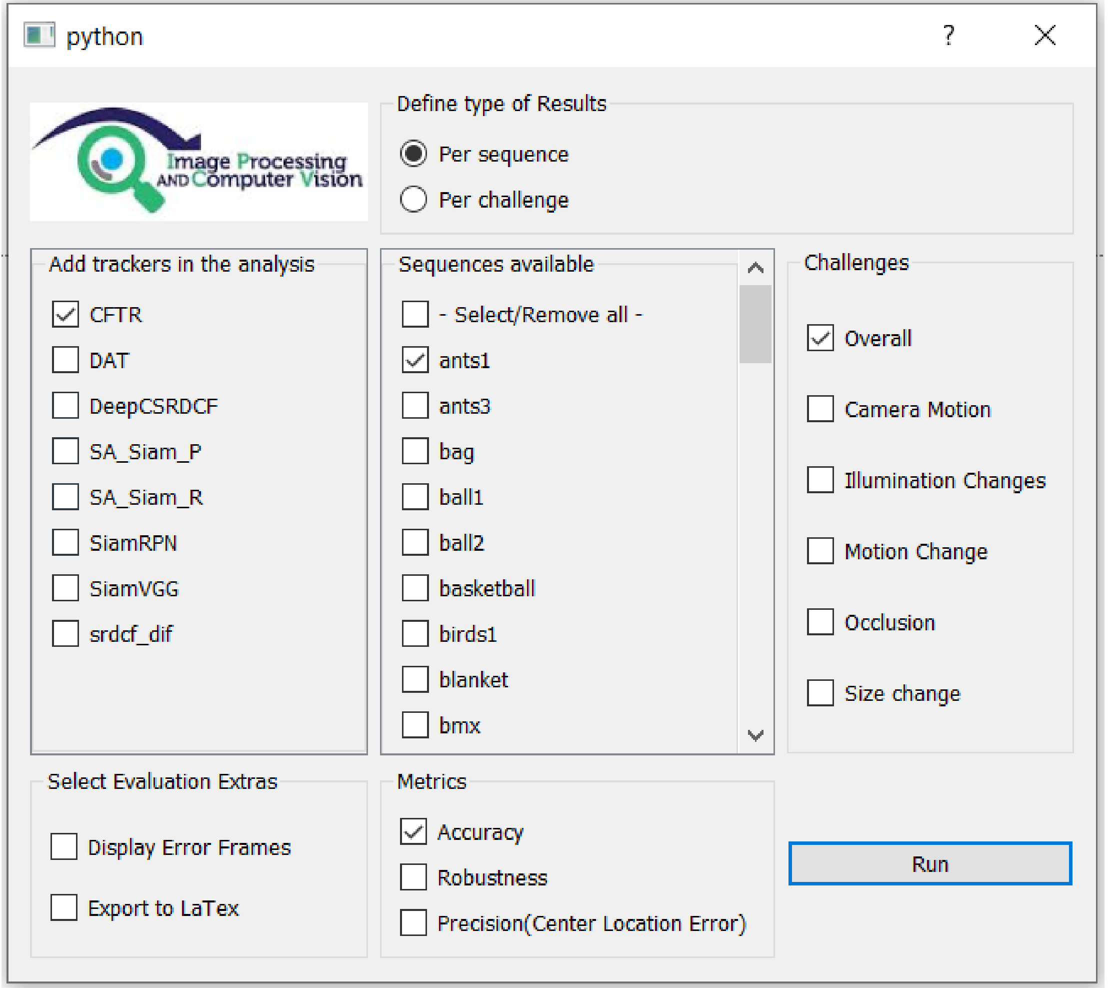
(Results and ground truth are actually the coordinates of the bounding box)

Interface

- User add to the corresponding folder the trackers and sequences to add in the analysis
- Select the analysis configuration
- Run



evaluation
interface
trackers
sequences
main.py



Define type of Results
● Per sequence
○ Per challenge

Add trackers in the analysis
☑ CFTR
□ DAT
□ DeepCSRDCF
□ SA_Siam_P
□ SA_Siam_R
□ SiamRPN
□ SiamVGG
□ srdcf_dif

Sequences available
□ - Select/Remove all -
☑ ants1
□ ants3
□ bag
□ ball1
□ ball2
□ basketball
□ birds1
□ blanket
□ bmx

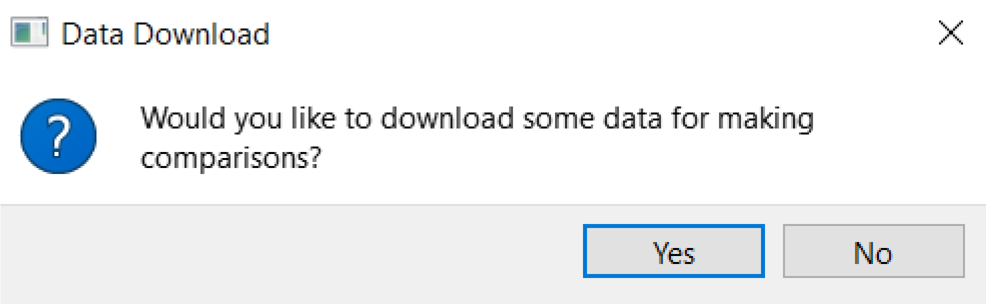
Challenges
☑ Overall
□ Camera Motion
□ Illumination Changes
□ Motion Change
□ Occlusion
□ Size change

Select Evaluation Extras
□ Display Error Frames
□ Export to LaTex

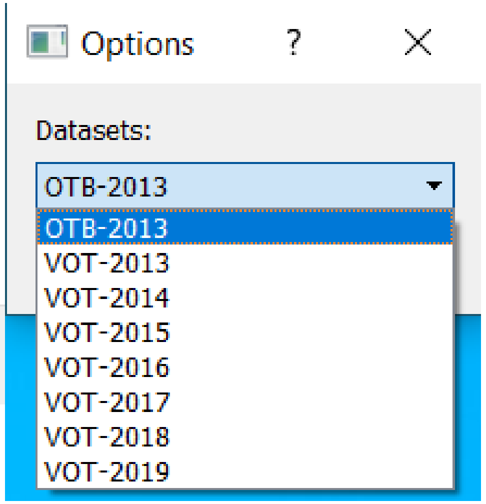
Metrics
☑ Accuracy
□ Robustness
□ Precision(Center Location Error)

Datasets

- Evaluator was first tested tracker results available from the VOT 2018 challenge
- Evaluator now works with data from OTB 2013 and older VOT datasets and tracker results also
- Functionality was added to allow automated download of tracker results from OTB or VOT results and sequences from 2013-2019

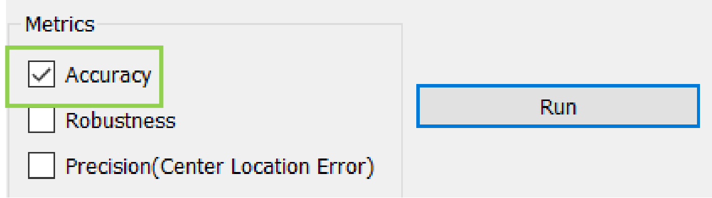

Data Download

Would you like to download some data for making comparisons?


Options

Datasets: OTB-2013

Evaluation

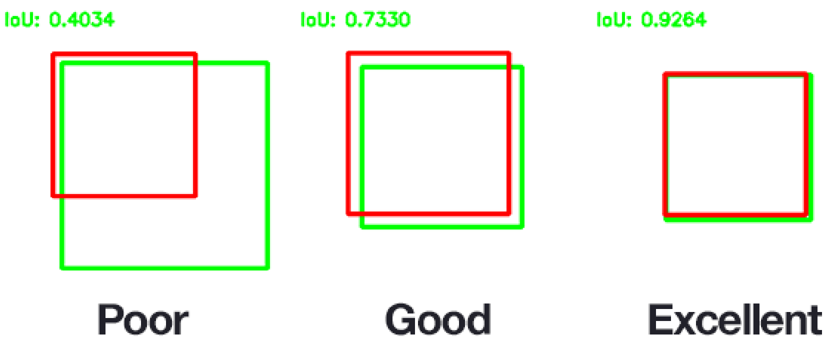


Metrics

☑ Accuracy

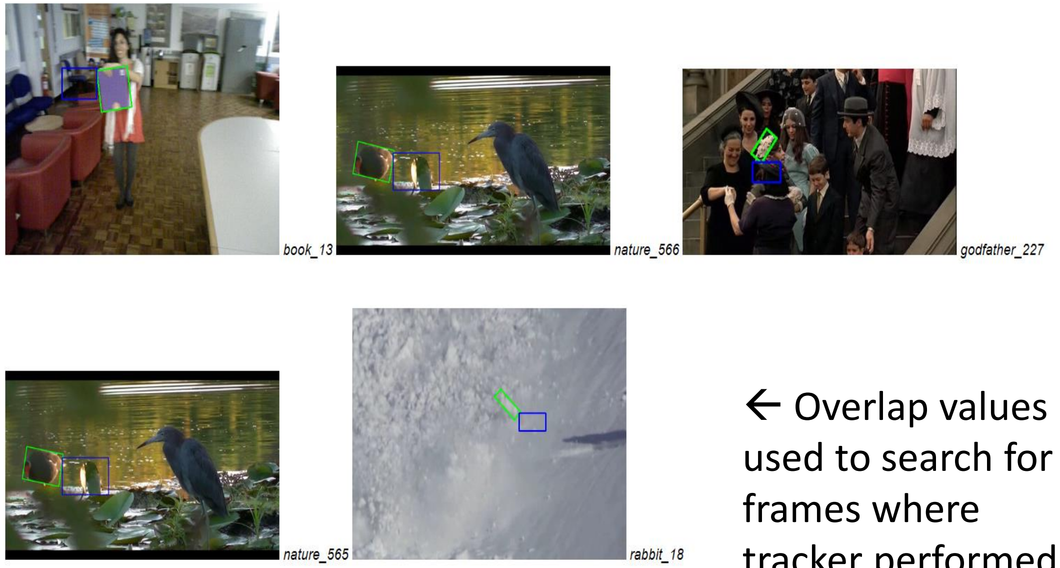
□ Robustness

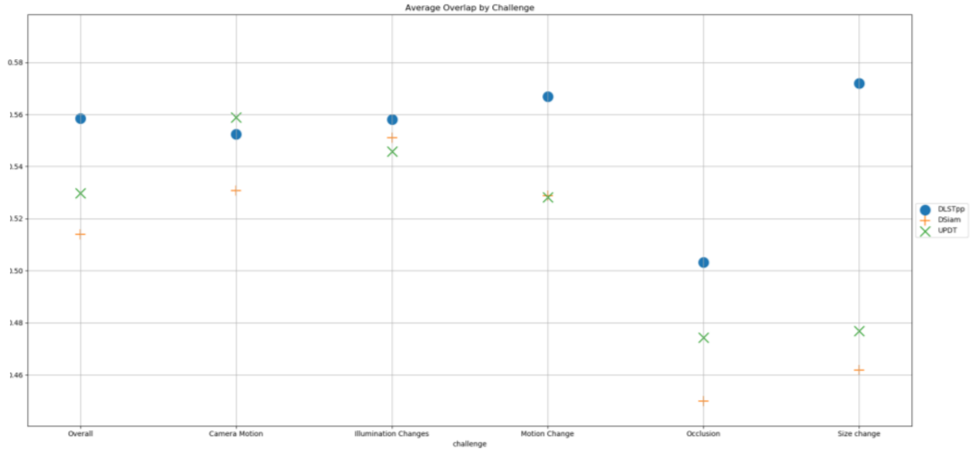
□ Precision(Center Location Error)



→ $\frac{Intersection}{Union}$

Low Frames for tracker: SiamFC challenge: Motion Change

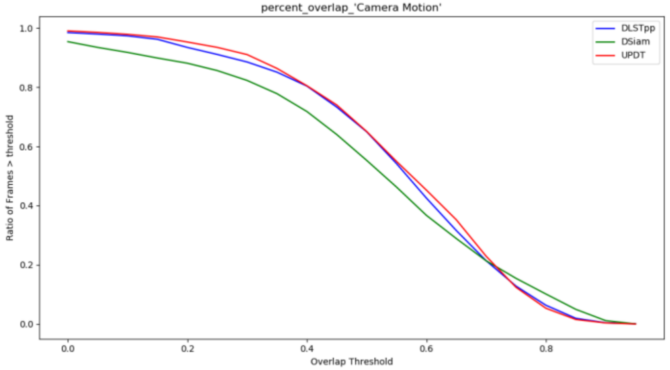




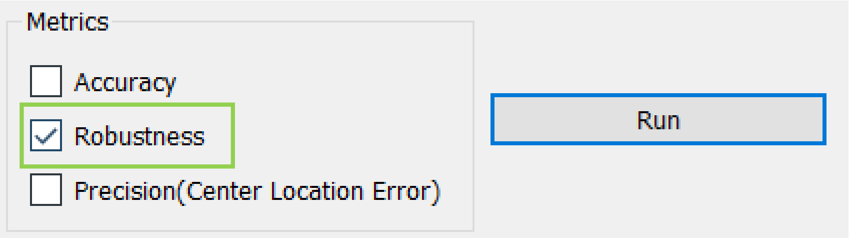
← Average overlap across sequences and challenges

← Overlap values used to search for frames where tracker performed poorly

→ Overlap from each frame used to show fraction of frames which are greater than threshold



Evaluation

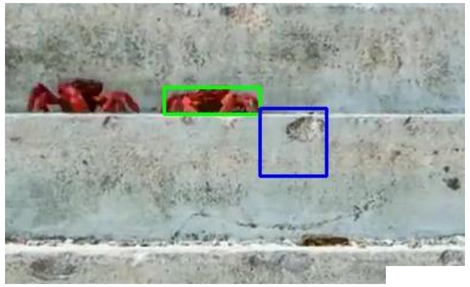


Metrics

□ Accuracy

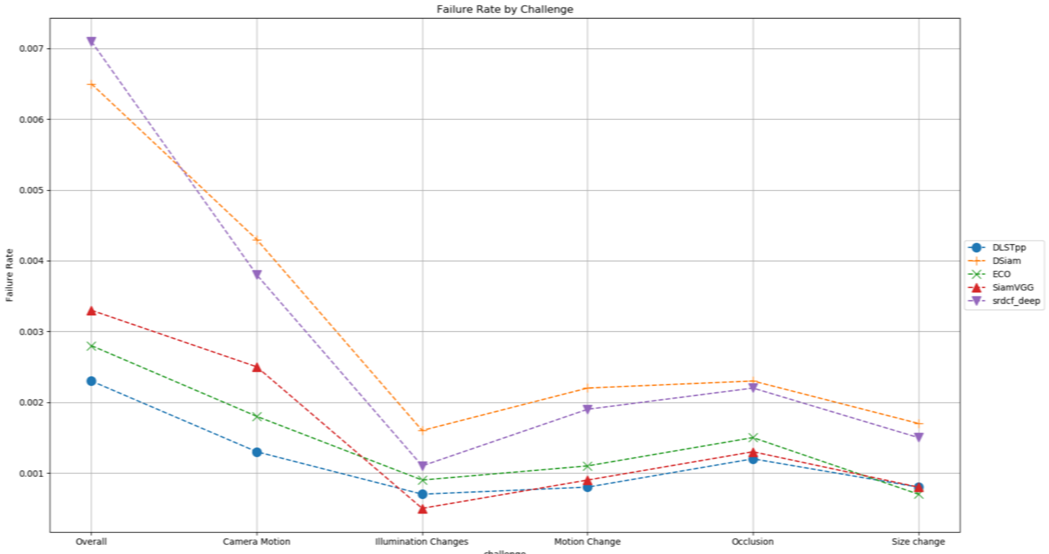
☑ Robustness

□ Precision(Center Location Error)

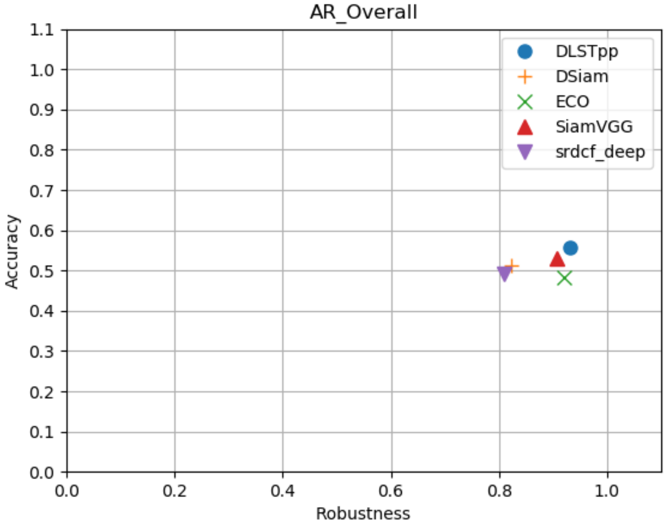


→ $$\frac{Failure\ Count}{Total\ Number\ of\ Frames\ Assessed}$$

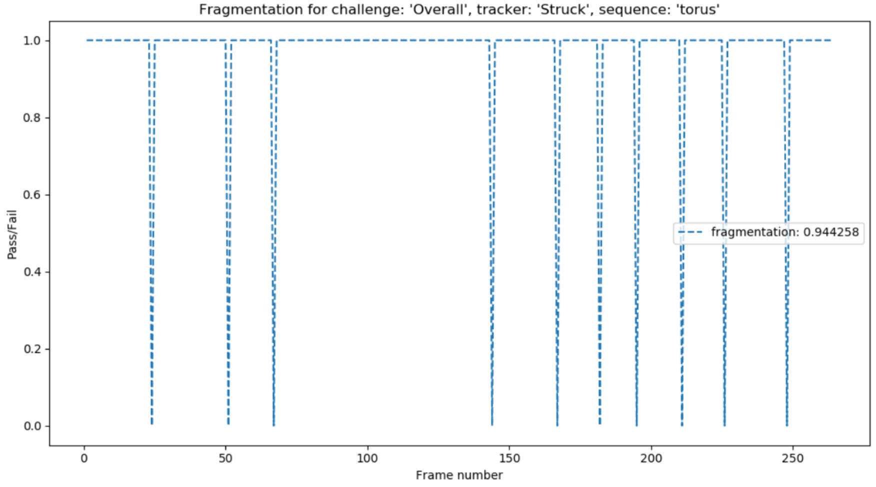
Expressed as a rate of failures per challenge/sequence →



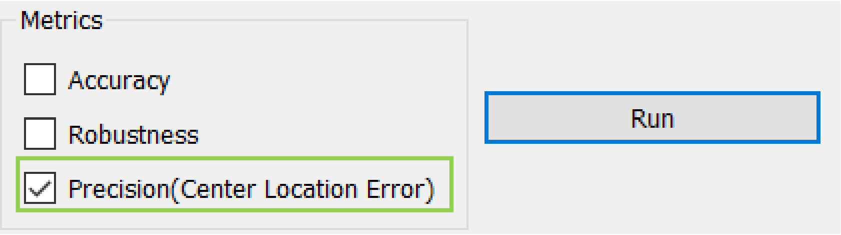
If failures of some sequence exceed 1, fragmentation plot created showing location of failures ↓



When selected with accuracy, robustness converted to 'reliability' for AR plot ↑



Evaluation

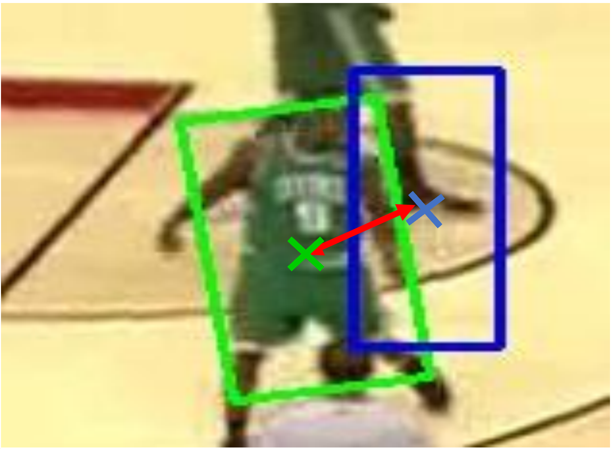


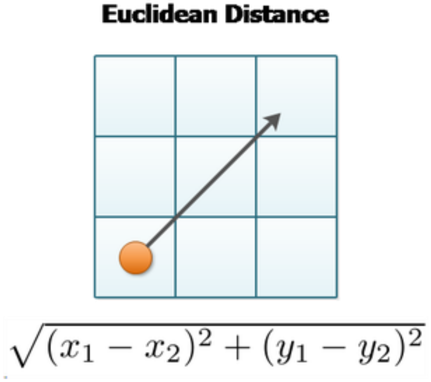
Metrics

□ Accuracy

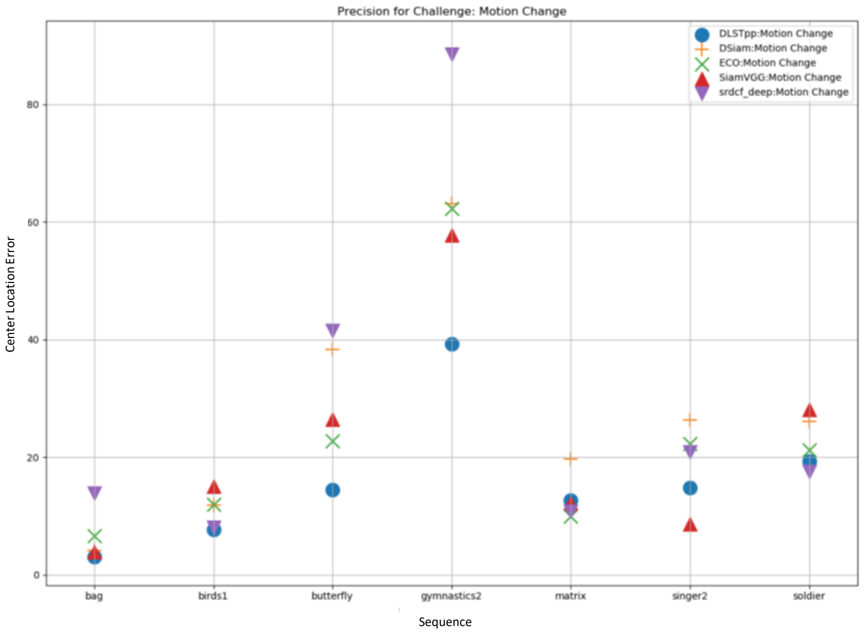
□ Robustness

☑ Precision(Center Location Error)



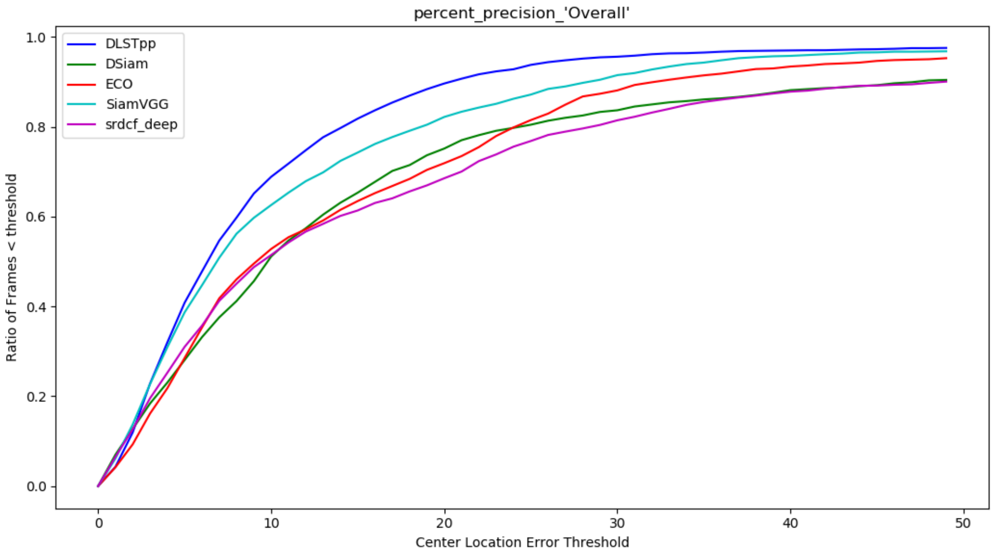


$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



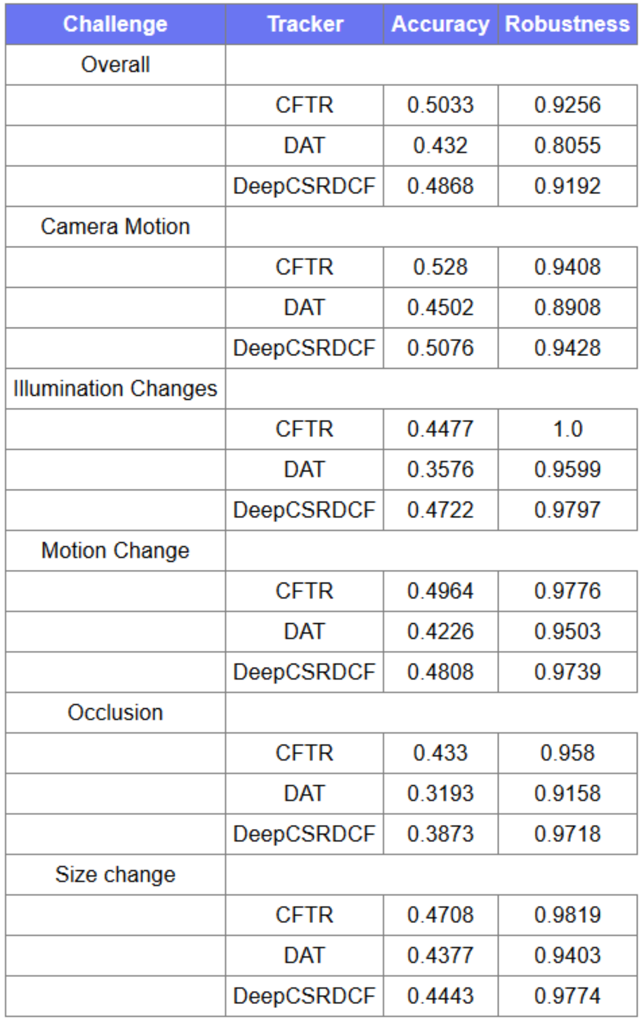
←Mean CLE plotted by challenge or by sequence for each challenge

Fraction of frames assessed less than some error threshold→

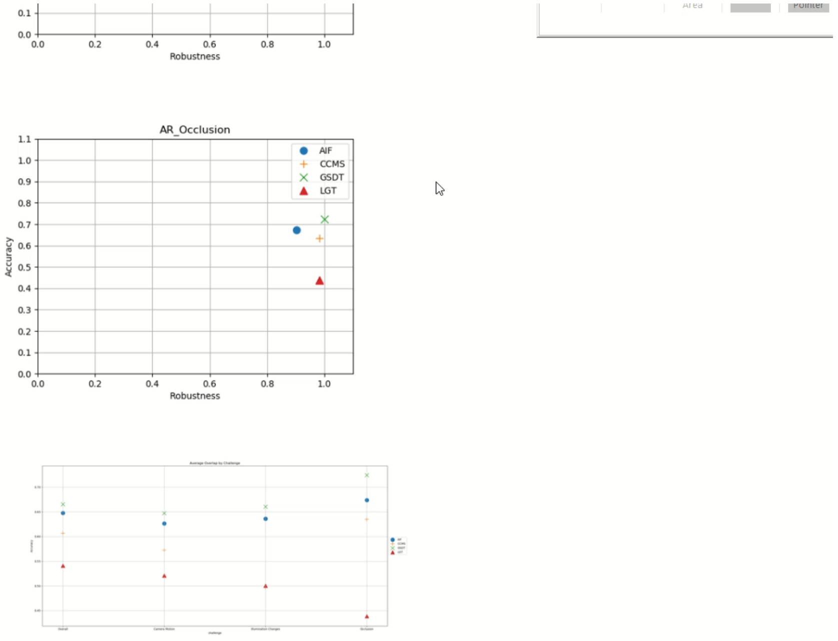


Output of Results

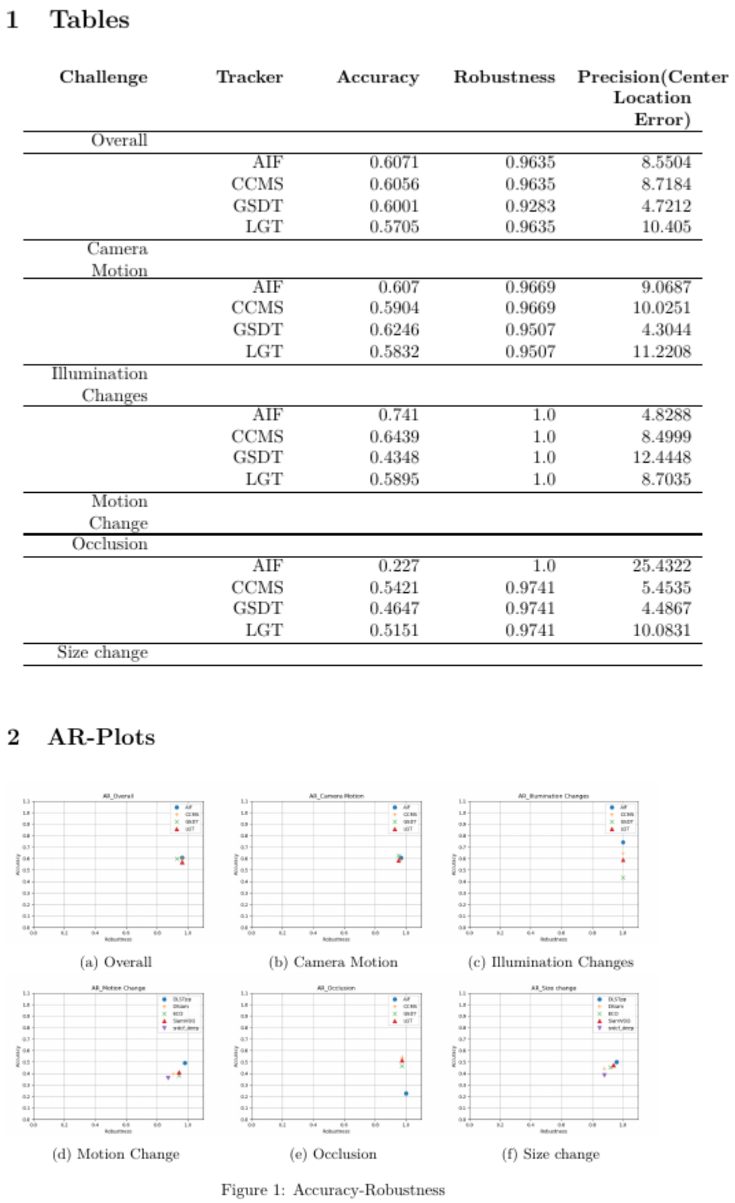
HTML & LaTex files



Challenge	Tracker	Accuracy	Robustness
Overall			
	CFTR	0.5033	0.9256
	DAT	0.432	0.8055
	DeepCSRDCF	0.4868	0.9192
Camera Motion			
	CFTR	0.528	0.9408
	DAT	0.4502	0.8908
	DeepCSRDCF	0.5076	0.9428
Illumination Changes			
	CFTR	0.4477	1.0
	DAT	0.3576	0.9599
	DeepCSRDCF	0.4722	0.9797
Motion Change			
	CFTR	0.4964	0.9776
	DAT	0.4226	0.9503
	DeepCSRDCF	0.4808	0.9739
Occlusion			
	CFTR	0.433	0.958
	DAT	0.3193	0.9158
	DeepCSRDCF	0.3873	0.9718
Size change			
	CFTR	0.4708	0.9819
	DAT	0.4377	0.9403
	DeepCSRDCF	0.4443	0.9774



- HTML output allows immediate visualization of plots and results in tables
- LaTeX output not completed but creates table with metrics and displays AR plots



1 Tables

Challenge	Tracker	Accuracy	Robustness	Precision(Center Location Error)
Overall				
	AIF	0.6071	0.9635	8.5504
	CCMS	0.6056	0.9635	8.7184
	GSDT	0.6001	0.9283	4.7212
	LGT	0.5705	0.9635	10.405
Camera Motion				
	AIF	0.607	0.9669	9.0687
	CCMS	0.5904	0.9669	10.0251
	GSDT	0.6246	0.9507	4.3044
	LGT	0.5832	0.9507	11.2208
Illumination Changes				
	AIF	0.741	1.0	4.8288
	CCMS	0.6439	1.0	8.4999
	GSDT	0.4348	1.0	12.4448
	LGT	0.5895	1.0	8.7035
Motion Change				
Occlusion				
	AIF	0.227	1.0	25.4322
	CCMS	0.5421	0.9741	5.4535
	GSDT	0.4647	0.9741	4.4867
	LGT	0.5151	0.9741	10.0831
Size change				

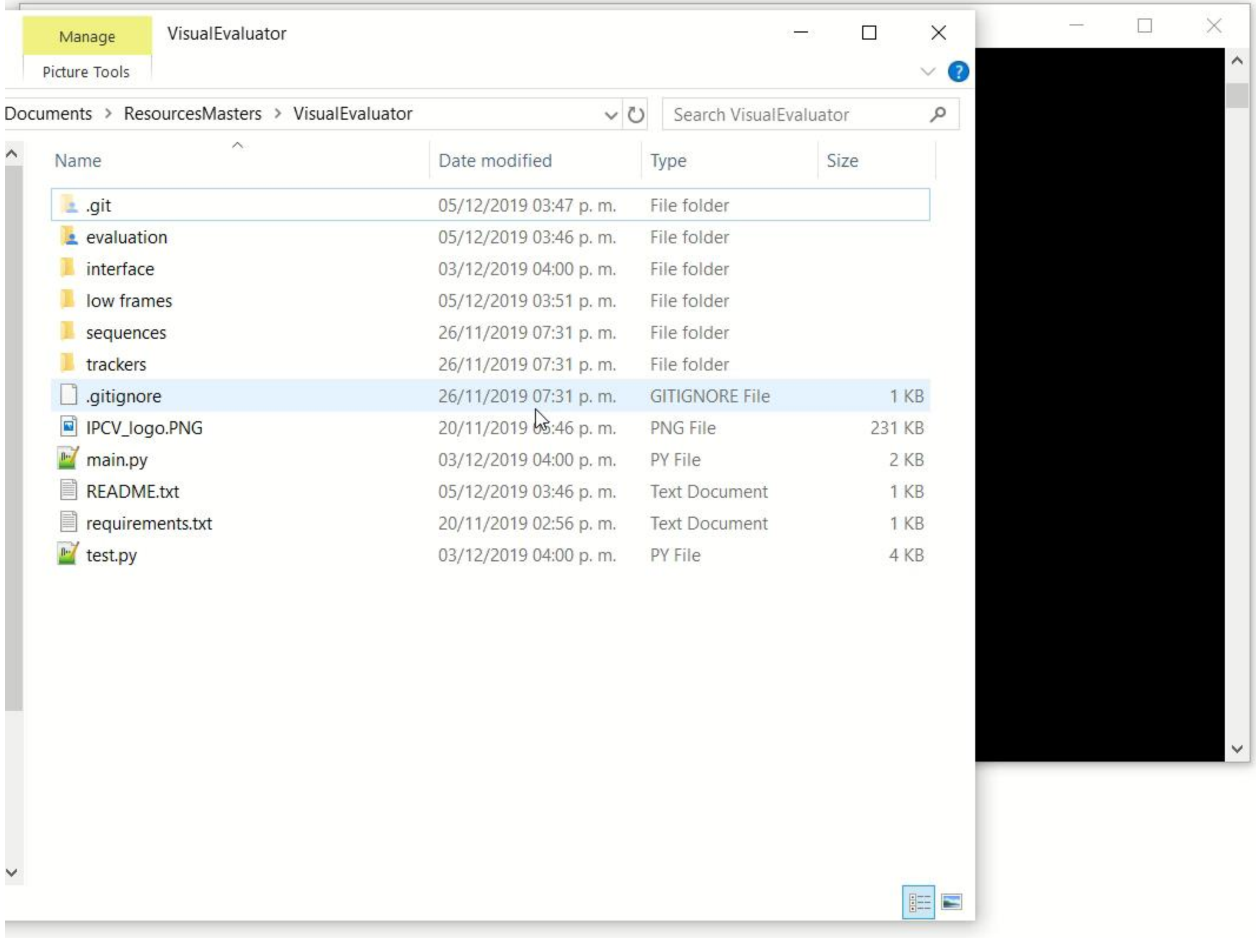
2 AR-Plots

(a) Overall (b) Camera Motion (c) Illumination Changes

(d) Motion Change (e) Occlusion (f) Size change

Figure 1: Accuracy-Robustness

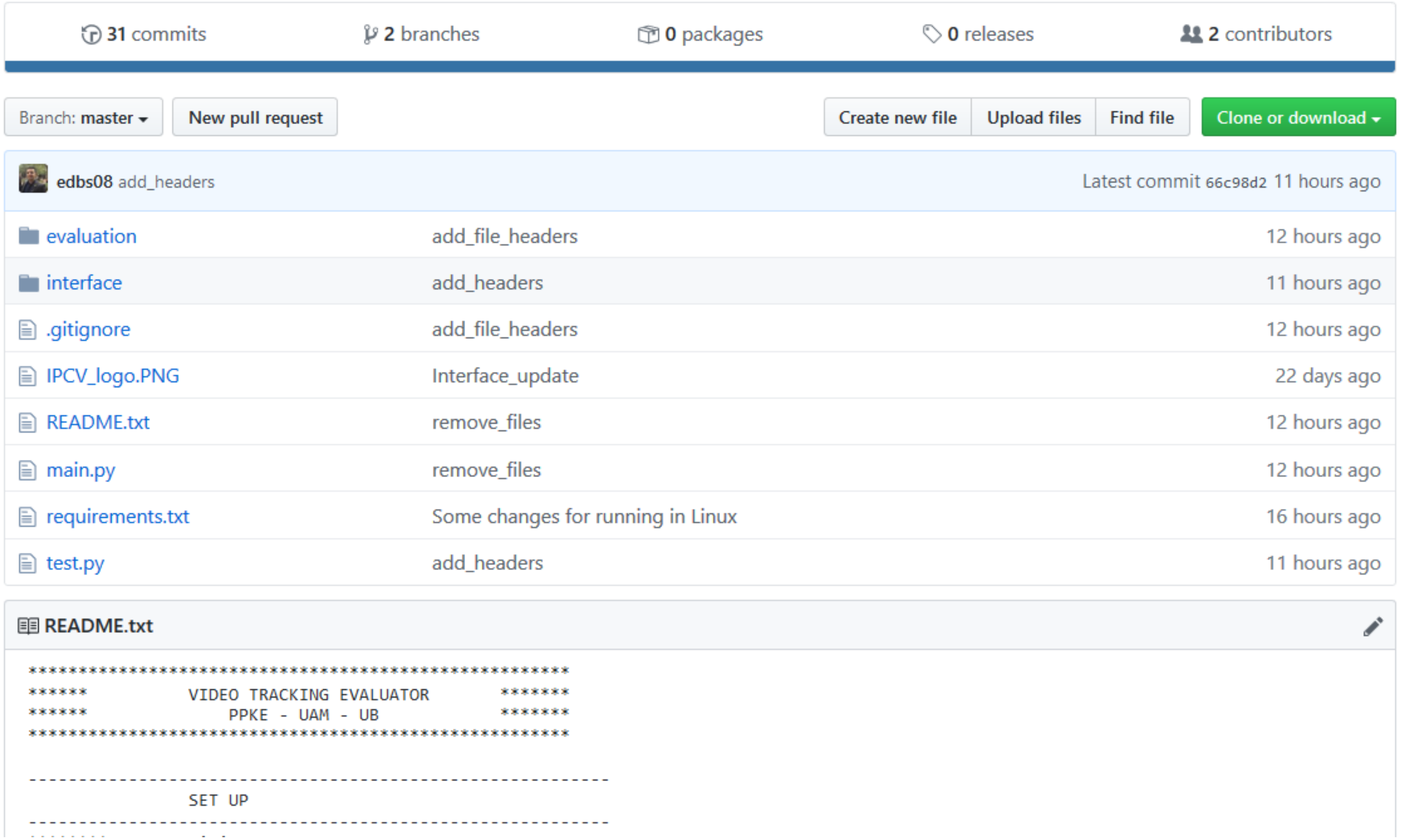
Example of Using Interface



Name	Date modified	Type	Size
.git	05/12/2019 03:47 p. m.	File folder	
evaluation	05/12/2019 03:46 p. m.	File folder	
interface	03/12/2019 04:00 p. m.	File folder	
low frames	05/12/2019 03:51 p. m.	File folder	
sequences	26/11/2019 07:31 p. m.	File folder	
trackers	26/11/2019 07:31 p. m.	File folder	
.gitignore	26/11/2019 07:31 p. m.	GITIGNORE File	1 KB
IPCV_logo.PNG	20/11/2019 05:46 p. m.	PNG File	231 KB
main.py	03/12/2019 04:00 p. m.	PY File	2 KB
README.txt	05/12/2019 03:46 p. m.	Text Document	1 KB
requirements.txt	20/11/2019 02:56 p. m.	Text Document	1 KB
test.py	03/12/2019 04:00 p. m.	PY File	4 KB

Code available on GitHub

https://github.com/edbs08/Tracking_Evaluator



edbs08 add_headers Latest commit 66c98d2 11 hours ago

evaluation	add_file_headers	12 hours ago
interface	add_headers	11 hours ago
.gitignore	add_file_headers	12 hours ago
IPCV_logo.PNG	Interface_update	22 days ago
README.txt	remove_files	12 hours ago
main.py	remove_files	12 hours ago
requirements.txt	Some changes for running in Linux	16 hours ago
test.py	add_headers	11 hours ago

README.txt

```
*******************************************************
******          VIDEO TRACKING EVALUATOR       *******
******              PPKE - UAM - UB            *******
*******************************************************

-----------------------------------------------------------
                SET UP
-----------------------------------------------------------
```

Comparison to Existing Available Evaluators/Benchmarks

	Ours	**Existing**
✓	Implementation in Python with minimal requirements	Often implemented in Matlab which requires license
✓	Flexibility of use with different datasets e.g VOT, OTB, custom, once you have groundtruth annotations and output results .txt file	Often tailored to work only with data generated on the specific benchmark e.g for OTB results tracker data must be stored in .mat files before comparison
✓	Intuitive with user interface	Often require some manipulation of the code to set up analysis
✓	Low performing frame search allowing more detailed information on where tracker is scoring low & LaTeX Report	Often only provides an overall result without a clear indication of how the tracker is failing
✗	Doesn’t allow the running of the tracker according to specific protocol	Benchmarks usually allow running of tracker in a certain routine to ensure fair comparisons e.g reset for VOT
✗	Not rigorously tested, possible unforeseen cases causing errors	Have been used by many researchers and experts who have added suggestions and fixes

Conclusion & Future Work

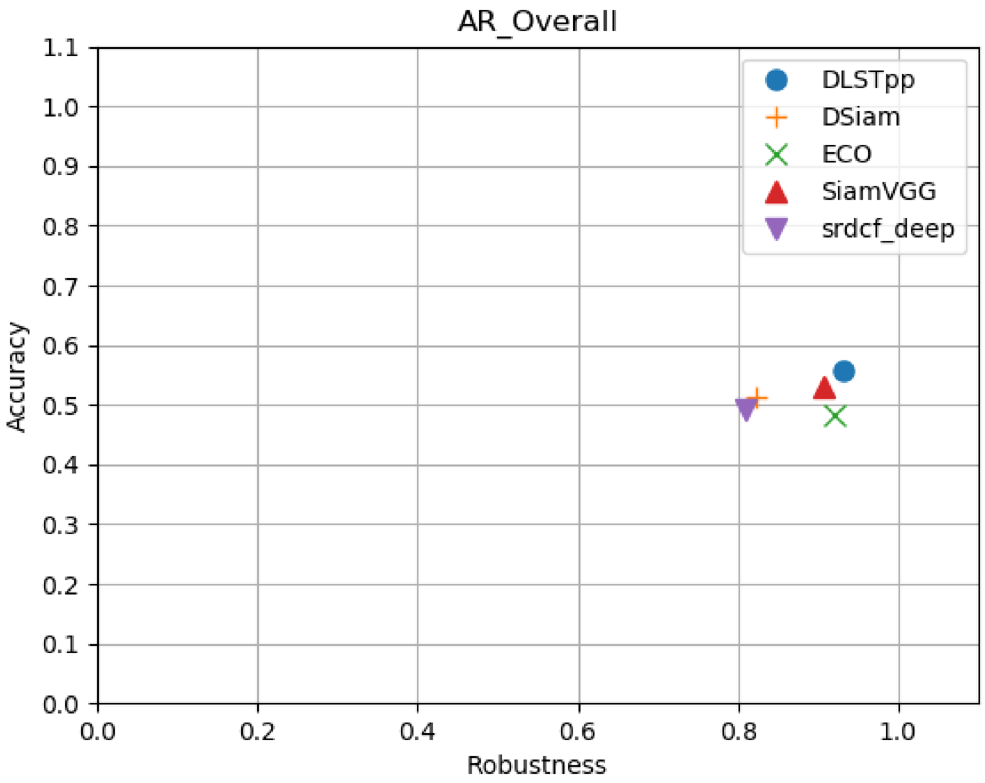
Achievements and Skills gained:

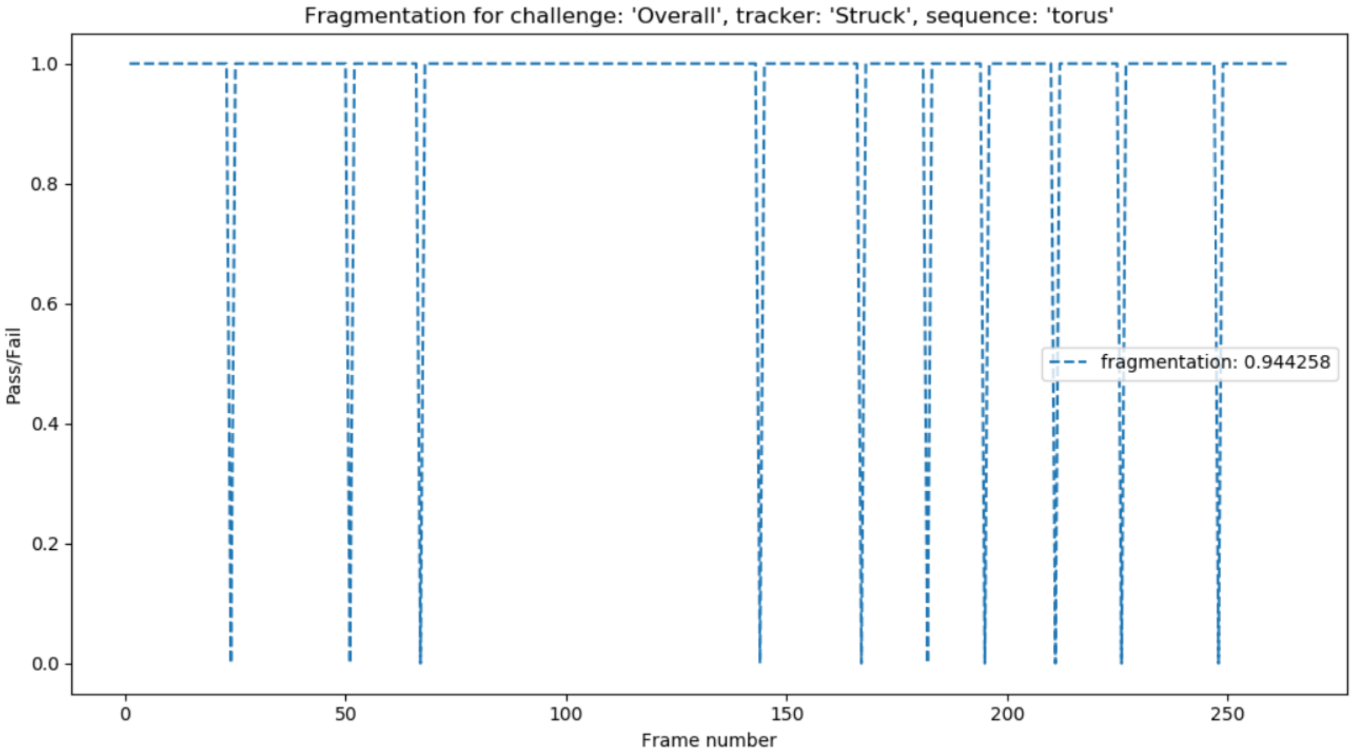
- Design of tracker evaluator in Python which can act as an accessible and useful tool for researchers
- Gained knowledge on important metrics for tracker evaluation
- Gained familiarity with data formatting and visualization in Python
- Use of PyQT for creating user interface
- Project organization and communication

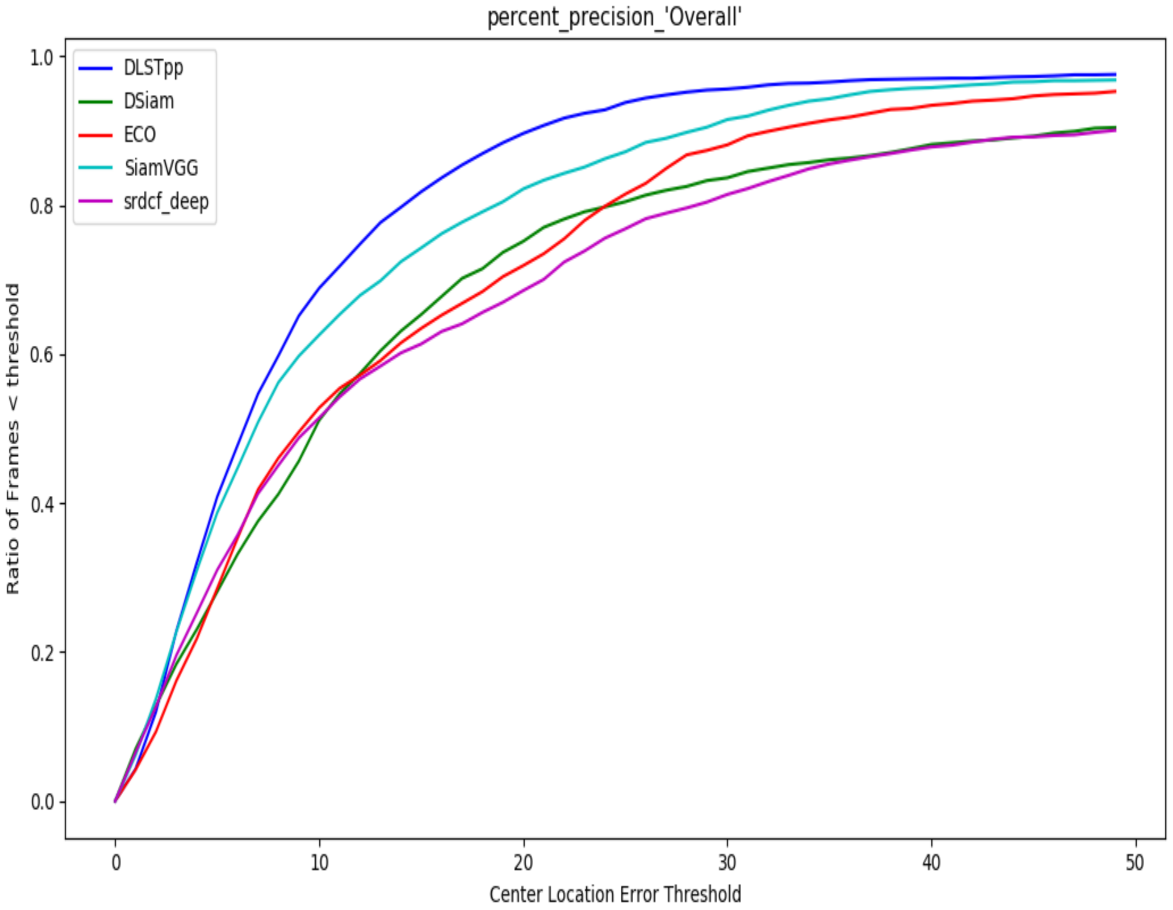
Limitations and Further Improvements:

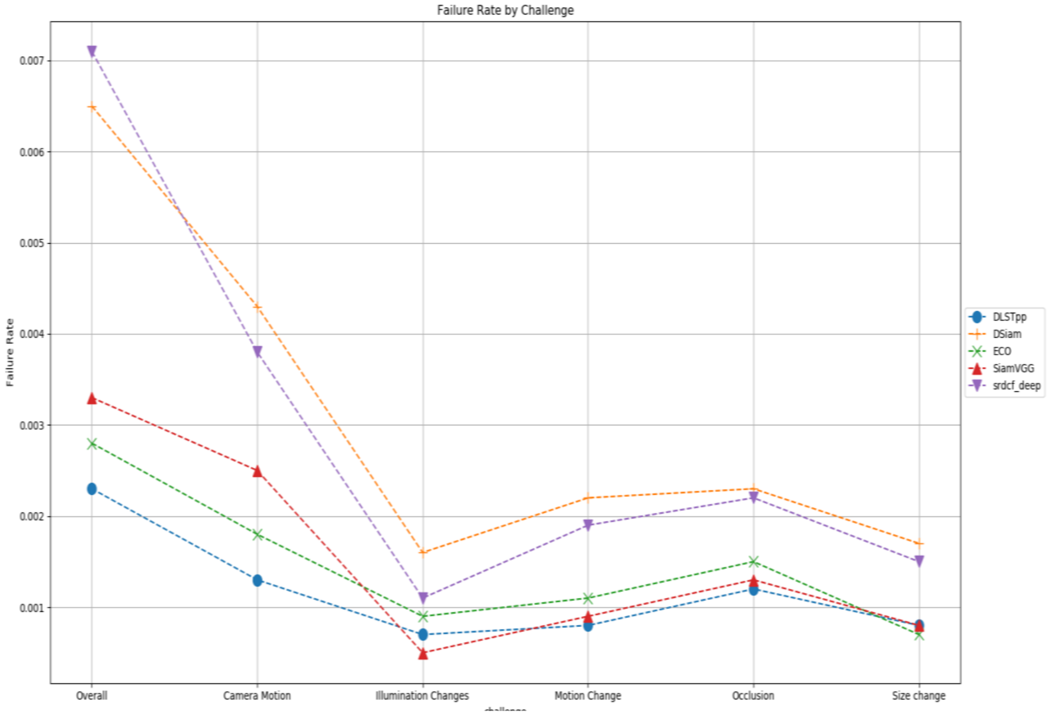
- Independent verification process of the code, where requirements are listed and tested in detail
- Add functionality for user to change thresholds with the UI
- Improve the appearance of the output results
- Increase customizability of the visualization and output
- Add possibilities for calculation of additional metrics

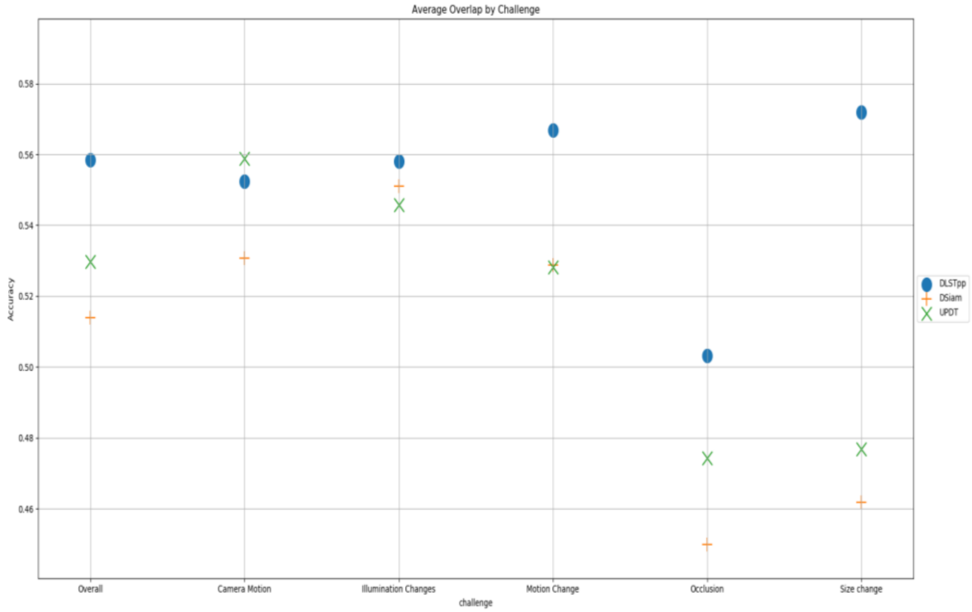
Additional Slides for Possible Questions


AR_Overall
Accuracy
Robustness
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
1.0
DLSTpp
DSiam
ECO
SiamVGG
srdcf_deep


Fragmentation for challenge: 'Overall', tracker: 'Struck', sequence: 'torus'
Pass/Fail
Frame number
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
fragmentation: 0.944258


percent_precision_'Overall'
DLSTpp
DSiam
ECO
SiamVGG
srdcf_deep
Ratio of Frames < threshold
Center Location Error Threshold
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50


Failure Rate by Challenge
Failure Rate
0.001
0.002
0.003
0.004
0.005
0.006
0.007
Overall
Camera Motion
Illumination Changes
Motion Change
Occlusion
Size change
challenge
DLSTpp
DSiam
ECO
SiamVGG
srdcf_deep


Average Overlap by Challenge
Accuracy
0.46
0.48
0.50
0.52
0.54
0.56
0.58
Overall
Camera Motion
Illumination Changes
Motion Change
Occlusion
Size change
challenge
DLSTpp
DSiam
UPDT

Low Frames for tracker: SiamFC challenge: Motion Change

book_13

nature_566

godfather_227

nature_565

rabbit_18

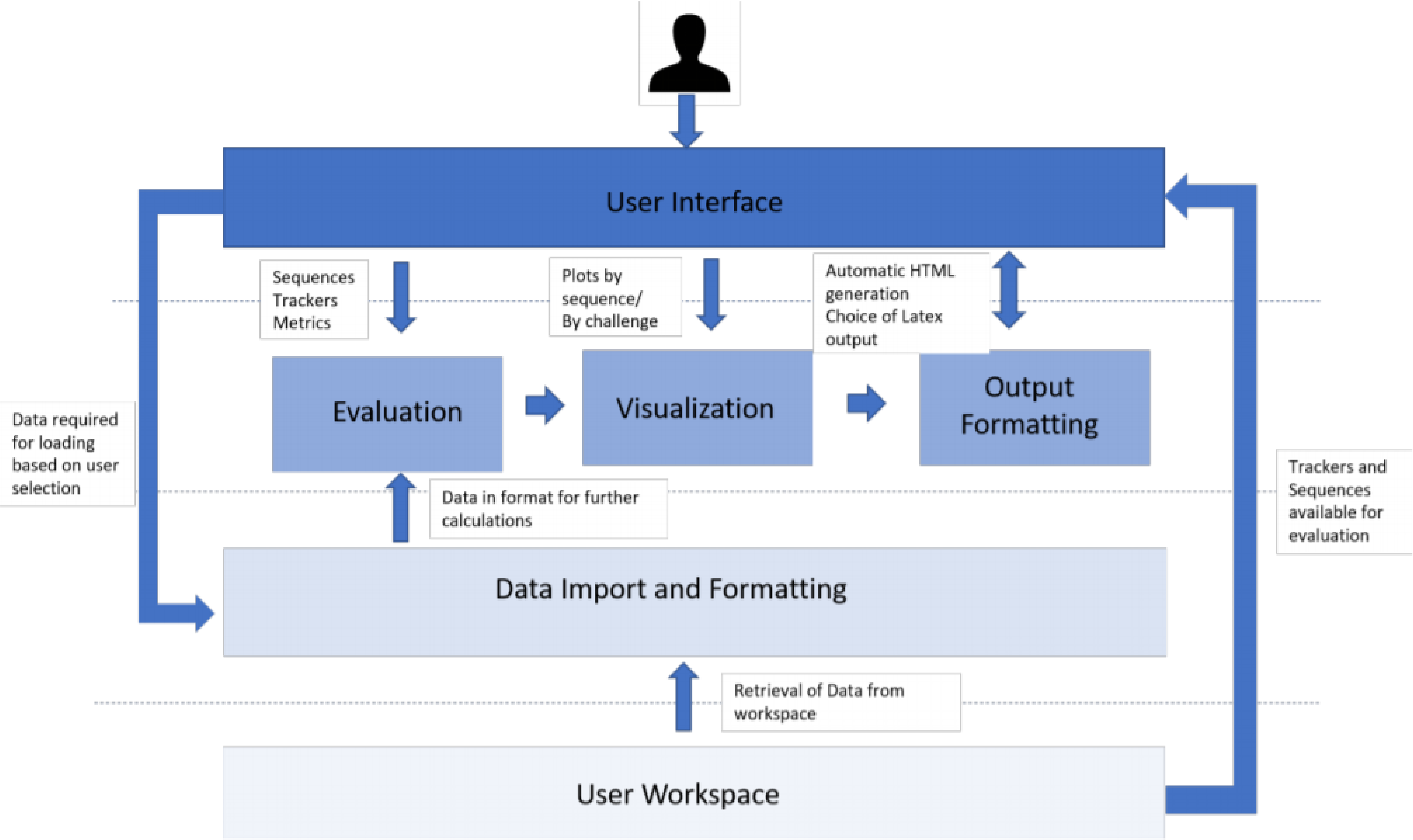
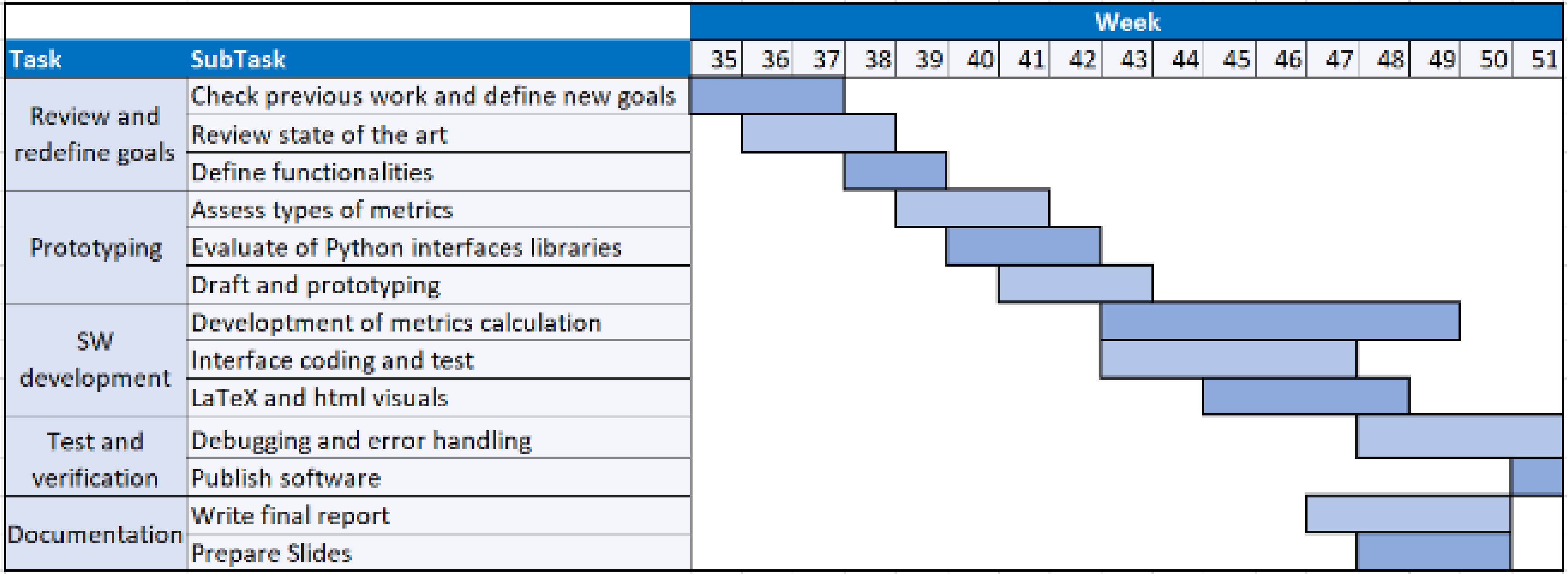
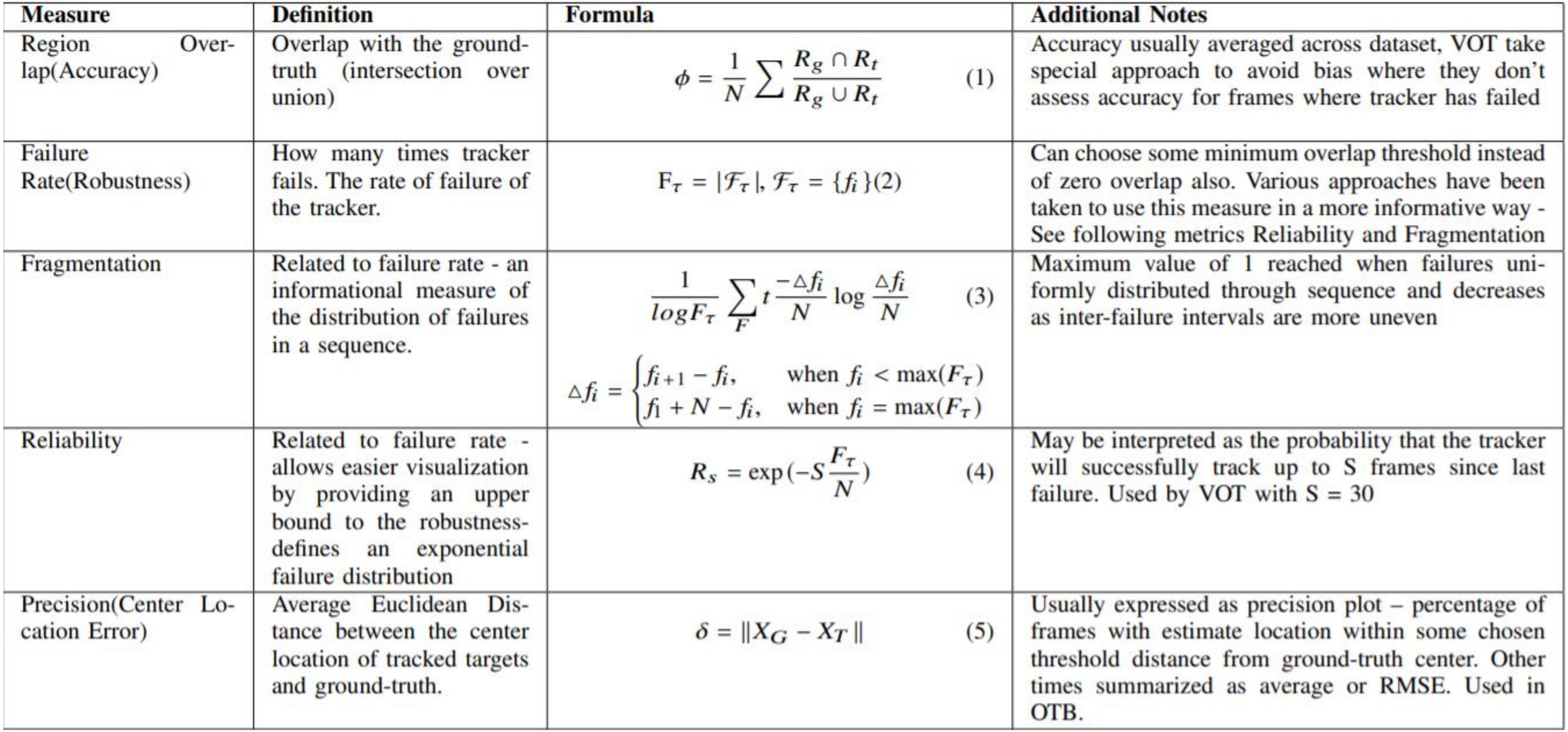


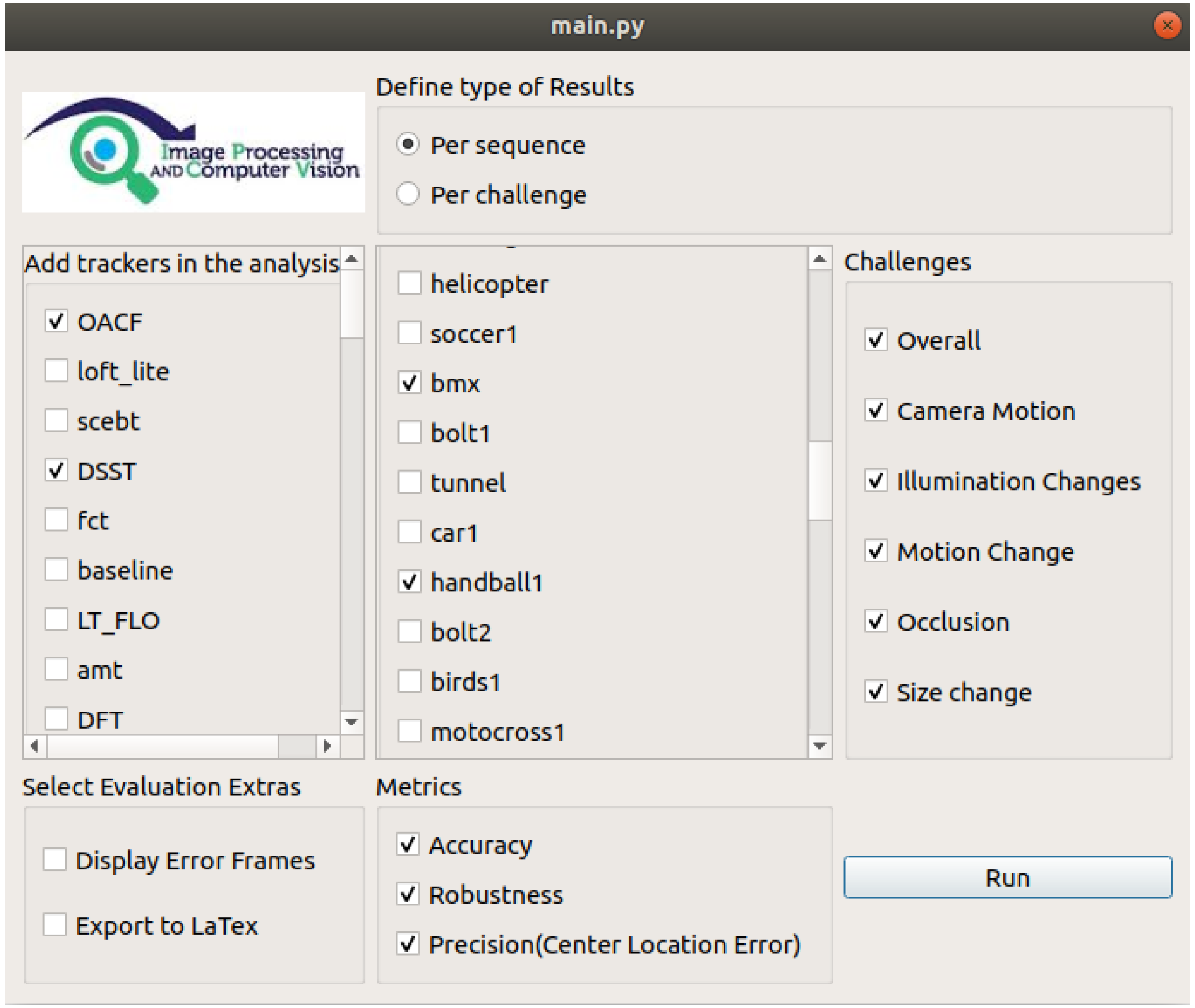
Fig. 2: Schematic of Software Architecture used for application



		Week																
Task	SubTask	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
Review and redefine goals	Check previous work and define new goals	▨	▨	▨														
	Review state of the art		▨	▨	▨													
	Define functionalities				▨	▨												
Prototyping	Assess types of metrics					▨	▨	▨										
	Evaluate of Python interfaces libraries						▨	▨	▨									
	Draft and prototyping							▨	▨	▨								
SW development	Developtment of metrics calculation									▨	▨	▨	▨	▨	▨	▨		
	Interface coding and test									▨	▨	▨	▨	▨				
	LaTeX and html visuals											▨	▨	▨	▨			
Test and verification	Debugging and error handling														▨	▨	▨	▨
	Publish software																	▨
Documentation	Write final report													▨	▨	▨	▨	
	Prepare Slides														▨	▨	▨	



Measure	Definition	Formula	Additional Notes
Region Overlap(Accuracy)	Overlap with the ground-truth (intersection over union)	$\phi = \frac{1}{N}\sum \frac{R_g \cap R_t}{R_g \cup R_t} \quad (1)$	Accuracy usually averaged across dataset, VOT take special approach to avoid bias where they don't assess accuracy for frames where tracker has failed
Failure Rate(Robustness)	How many times tracker fails. The rate of failure of the tracker.	$\mathrm{F}_\tau = \lvert\mathcal{F}_\tau\rvert, \mathcal{F}_\tau = \{f_i\}(2)$	Can choose some minimum overlap threshold instead of zero overlap also. Various approaches have been taken to use this measure in a more informative way - See following metrics Reliability and Fragmentation
Fragmentation	Related to failure rate - an informational measure of the distribution of failures in a sequence.	$\frac{1}{logF_\tau}\sum_F t\frac{-\Delta f_i}{N}\log\frac{\Delta f_i}{N} \quad (3)$ $\Delta f_i = \begin{cases} f_{i+1} - f_i, & \text{when } f_i < \max(F_\tau) \\ f_1 + N - f_i, & \text{when } f_i = \max(F_\tau) \end{cases}$	Maximum value of 1 reached when failures uniformly distributed through sequence and decreases as inter-failure intervals are more uneven
Reliability	Related to failure rate - allows easier visualization by providing an upper bound to the robustness- defines an exponential failure distribution	$R_s = \exp(-S\frac{F_\tau}{N}) \quad (4)$	May be interpreted as the probability that the tracker will successfully track up to S frames since last failure. Used by VOT with S = 30
Precision(Center Location Error)	Average Euclidean Distance between the center location of tracked targets and ground-truth.	$\delta = \lVert X_G - X_T \rVert \quad (5)$	Usually expressed as precision plot – percentage of frames with estimate location within some chosen threshold distance from ground-truth center. Other times summarized as average or RMSE. Used in OTB.


main.py

Define type of Results

● Per sequence

○ Per challenge

Add trackers in the analysis

☑ OACF
□ loft_lite
□ scebt
☑ DSST
□ fct
□ baseline
□ LT_FLO
□ amt
□ DFT

□ helicopter
□ soccer1
☑ bmx
□ bolt1
□ tunnel
□ car1
☑ handball1
□ bolt2
□ birds1
□ motocross1

Challenges

☑ Overall
☑ Camera Motion
☑ Illumination Changes
☑ Motion Change
☑ Occlusion
☑ Size change

Select Evaluation Extras

□ Display Error Frames
□ Export to LaTex

Metrics

☑ Accuracy
☑ Robustness
☑ Precision(Center Location Error)

- Various use cases tested on Windows for:
  - VOT 2018
  - VOT 2013
  - OTB
- Linux:
  - VOT 2015