Algorithmic Human-Robot Interaction

Activity Recognition
System Design Workshop Part II
System Design Pitches

CSCI 7000

Prof. Brad Hayes

University of Colorado Boulder

HRI 2019
Daegu, Korea

No Class Next Week

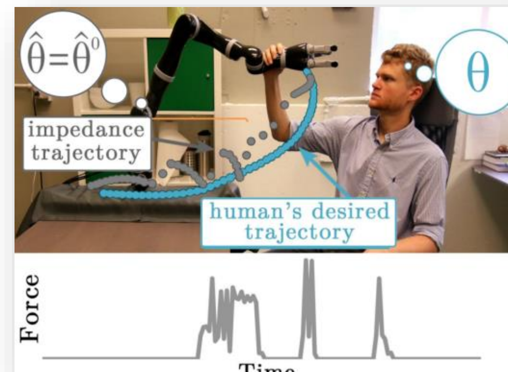
Classes cancelled
3/12 and 3/14
due to the
HRI Conference

Papers for Thursday 3/7:
Interpreting and Expressing Goals

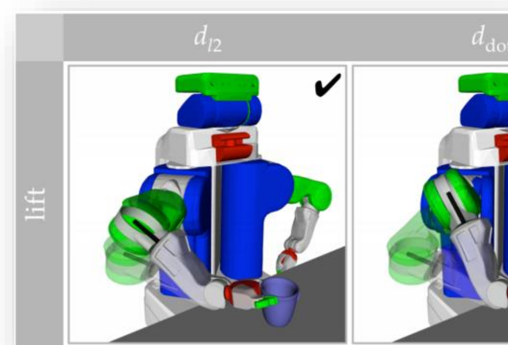
(E-mail Bradley.Hayes@Colorado.edu to sign up)

Learning Robot Objectives from Physical Human Interaction – Bajcsy et al.

Pro: Lakhan Kamireddy
Con: Chandan Naik



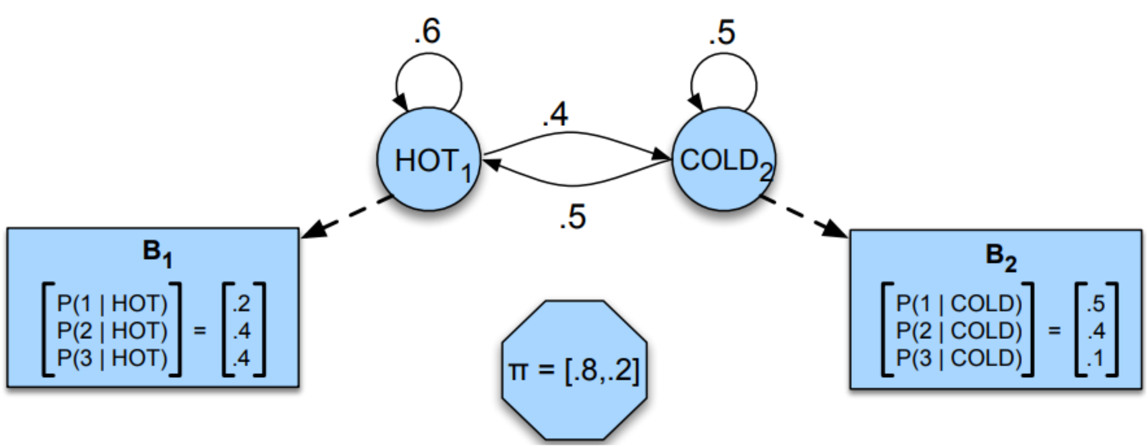
Expressing Robot Incapability – Kwon et al.
Pro: Dhanendra Soni
Con: Shohei Wakayama



Hidden Markov Models

Variables:

$S = s_1, s_2, \ldots, s_N$	(States)
$V = v_1, v_2, \ldots, v_k$	(Observation Vocab.)
$A = a_{11}, \ldots a_{ij}, \ldots a_{NN}$	(Transition prob. Matrix)
$B = P(o_t \mid s_i)\ \forall\ i \in [1, N], t \in [1, T]$	(Obs. Emission Probs)
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	(Initial prob. distribution)
$O = o_1, o_2, \ldots, o_T$	(Observation Sequence)
$Q = s_1, s_2, \ldots, s_T$	(State Sequence)



Three Types of Problems

- **Likelihood:**

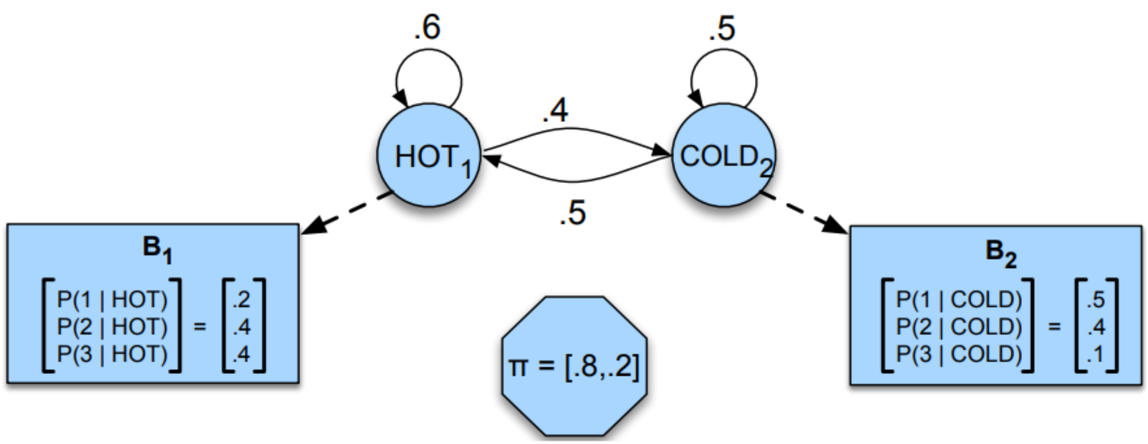
  Given A, B, O … Determine $P(O|A, B)$

- **Decoding:**

  Given $\mathrm{A}, \mathrm{B}, O$ … Determine the ‘best’ hidden state sequence

- **Learning:**

  Given O and S … Determine A, B



Likelihood Computation: Forward Algorithm

Example: Given A,B and $\boldsymbol{O} = \{3, 1, 3\}$ -- Determine $\boldsymbol{P(O|A, B)}$

$$\alpha_t(j) = P(o_1, o_2, \ldots, o_t, q_t = j \mid A, B)$$

$$\alpha_t(j) = \sum_{i=1}^{N} \underbrace{\alpha_{t-1}(i)}_{\text{Prev}} * \underbrace{a_{ij}}_{\text{P(i -> j)}} * \underbrace{b_j(o_t)}_{\text{P(o|s)}}$$

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward[N,T]*

for each state s **from** 1 **to** N **do** ; initialization step

$\quad forward[s,1] \leftarrow \pi_s * b_s(o_1)$

for each time step t **from** 2 **to** T **do** ; recursion step

$\quad$ **for** each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

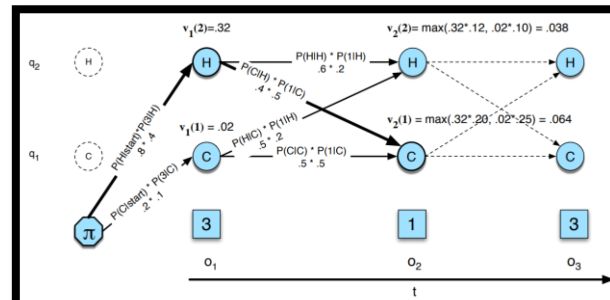
$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T]$; termination step

return *forwardprob*

State Sequence Computation: Viterbi Algorithm

function VITERBI(*observations* of len *T*,*state-graph* of len *N*) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
for each time step *t* **from** 2 **to** *T* **do** ; recursion step
    for each state *s* **from** 1 **to** *N* **do**
        $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
        $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*



Learning an HMM's Parameters

Learning: Given $\boldsymbol{O}$ and $\boldsymbol{S}$ … Determine $\boldsymbol{A}, \boldsymbol{B}$

Challenge: Must simultaneously determine **transition probabilities** AND **emission probabilities**!

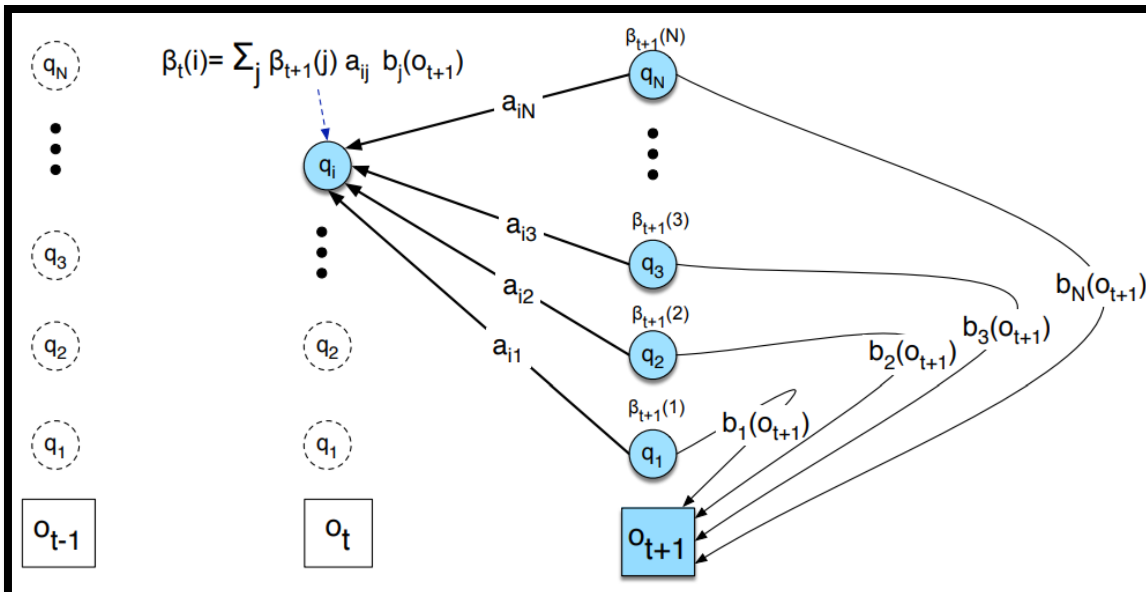
Special case of Expectation-Maximization, iteratively improving an initial estimate.

But first, let's solve for a Markov Chain (*fully observable*) given $\boldsymbol{O}, \boldsymbol{S}, \boldsymbol{Q}$

Backward Algorithm

Backward Probability: $\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots o_t | q_t = i, A, B)$

If we're in state i at time t, what's P(Obs) from then to end?



Initialization: $\beta_T(i) = 1,\ \ i \in [1, N]$

Recursion: $\beta_t(i) = \sum_{j=1}^{N} a_{ij} * b_j(o_{t+1}) * \beta_{t+1}(j)$, $\mathrm{i} \in [1, N], t \in [1, T)$

Termination: $P(O|A, B) = \sum_{j=1}^{N} \pi_j * b_j(o_1) * \beta_1(j)$

Forward-Backward: Learning A

$$\hat{a}_{ij} = \frac{Expected\ \#transitions\ from\ i\ to\ j}{Expected\ \#transitions\ from\ i}$$

To compute numerator:

1. Assume we have probability estimate for $i \rightarrow j$ at time t
2. Now assume we had that for all t: sum over all $t \in [0, T)$ to get the total count for $i \rightarrow j$

Define ξ_t as probability of transition from i to j at time t:

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid O, A, B)$$

…But we don't know the relation between O and Q!

Forward-Backward: Learning A

Define ξ_t as probability of transition from i to j at time t:

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid O, A, B)$$

...But we don't know the relation between O and Q!

So we define sort-of-$\xi_t(i,j) = P(q_t = i, q_{t+1} = j, O \mid A, B)$

$$\text{sort-of-}\xi_t(i,j) = \alpha_t(i) * a_{ij} * b_j(o_{t+1}) * \beta_{t+1}(j)$$

Forward

$$\boxed{\alpha_t(j) = P(o_1, o_2, \ldots, o_t, q_t = j \mid A, B)}$$

Backward

$$\boxed{\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots o_t | q_t = i, A, B)}$$

Forward-Backward: Learning A

$$\text{sort-of-}\xi_t(i,j) = \alpha_t(i) * a_{ij} * b_j(o_{t+1}) * \beta_{t+1}(j)$$

How do we go from $\boxed{P(q_t = i, q_{t+1} = j, O \mid A, B)}$ to $\boxed{P(q_t = i, q_{t+1} = j \mid O, A, B)}$

Recall: $P(X|Y,Z) = \frac{P(X,Y|Z)}{P(Y|Z)}$

Thus, because $P(O|A,B) = \sum_{j=1}^{N} \alpha_t(j) * \beta_t(j)$

$$\xi_t(i,j) = \frac{\alpha_t(i) * a_{ij} * b_j(o_{t+1}) * \beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j) * \beta_t(j)}$$

Forward

$\boxed{\alpha_t(j) = P(o_1, o_2, \dots, o_t, q_t = j \mid A, B)}$

Backward

$\boxed{\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots o_t | q_t = i, A, B)}$

Forward-Backward: Learning A

To compute numerator:

1. Assume we have probability estimate for $i \to j$ at time t
2. Now assume we had that for all t: sum over all $t \in [0, T)$ to get the total count for $i \to j$

$$\hat{a}_{ij} = \frac{Expected\ \#transitions\ from\ i\ to\ j}{Expected\ \#transitions\ from\ i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

$$\xi_t(i,j) = \frac{\alpha_t(i) * a_{ij} * b_j(o_{t+1}) * \beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j) * \beta_t(j)}$$

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid O, A, B)$$

Forward

$$\boxed{\alpha_t(j) = P(o_1, o_2, \ldots, o_t, q_t = j \mid A, B)}$$

Backward

$$\boxed{\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots o_t | q_t = i, A, B)}$$

Forward-Backward: Learning B

$$\hat{a}_{ij} = \frac{Expected\ \#transitions\ from\ i\ to\ j}{Expected\ \#transitions\ from\ i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid O, A, B)$$

Now we need to compute observation emission probability:

$$\widehat{b}_j(v_k) = \frac{Expected\ \#\ of\ v_k\ seen\ in\ state\ j}{Expected\ \#times\ in\ state\ j}$$

Forward

$$\boxed{\alpha_t(j) = P(o_1, o_2, \ldots, o_t, q_t = j \mid A, B)}$$

Backward

$$\boxed{\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots o_t | q_t = i, A, B)}$$

Forward-Backward: Learning B

Now we need to compute observation emission probability:

$$\widehat{b}_j(v_k) = \frac{Expected\ \#\ of\ v_k\ seen\ in\ state\ j}{Expected\ \#times\ in\ state\ j}$$

But first, we need to know prob. of being in state *j* at time *t*

$$\gamma_t(j) = \mathrm{P}(\mathrm{q_t} = j \mid O, A, B) = \frac{P(q_t = j, O|A, B)}{P(O|A, B)}$$

$$\gamma_t(j) = \frac{\alpha_t(j) * \beta_t(j)}{P(O|A, B)}$$

Forward

$\alpha_t(j) = P(o_1, o_2, \ldots, o_t, q_t = j \mid A, B)$

Backward

$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots o_t | q_t = i, A, B)$

Forward-Backward: Learning B

Now we need to compute observation emission probability:

$$\hat{b}_j(v_k) = \frac{Expected\ \#\ of\ v_k\ seen\ in\ state\ j}{Expected\ \#times\ in\ state\ j} = \frac{\sum_{t=1}^{T} \gamma_t(j) * I(o_t = v_k)}{\sum_{t=1}^{T} \gamma_t(j)}$$

$\gamma_t(j) =$ prob. of being in state j at time t

$$\gamma_t(j) = \frac{\alpha_t(j) * \beta_t(j)}{P(O|A,B)}$$

Forward

$$\boxed{\alpha_t(j) = P(o_1, o_2, \dots, o_t, q_t = j \mid A, B)}$$

Backward

$$\boxed{\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots o_t | q_t = i, A, B)}$$

Expectation-Maximization on A, B

E-Step: Compute state occupancy count γ, expected state transition count ξ using existing A, B probabilities

M-Step: Compute A, B using existing γ and ξ probabilities

$\alpha_t(j)$ = prob. to be in state j at t

$\beta_t(j)$ = prob. of O from state j at t

$\xi_t(i,j)$ = prob. of transition from i to j at time t

$\gamma_t(j)$ = prob. of being in state j at time t

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** *HMM=(A,B)*

initialize A and B
iterate until convergence
E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t. O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Looking Ahead

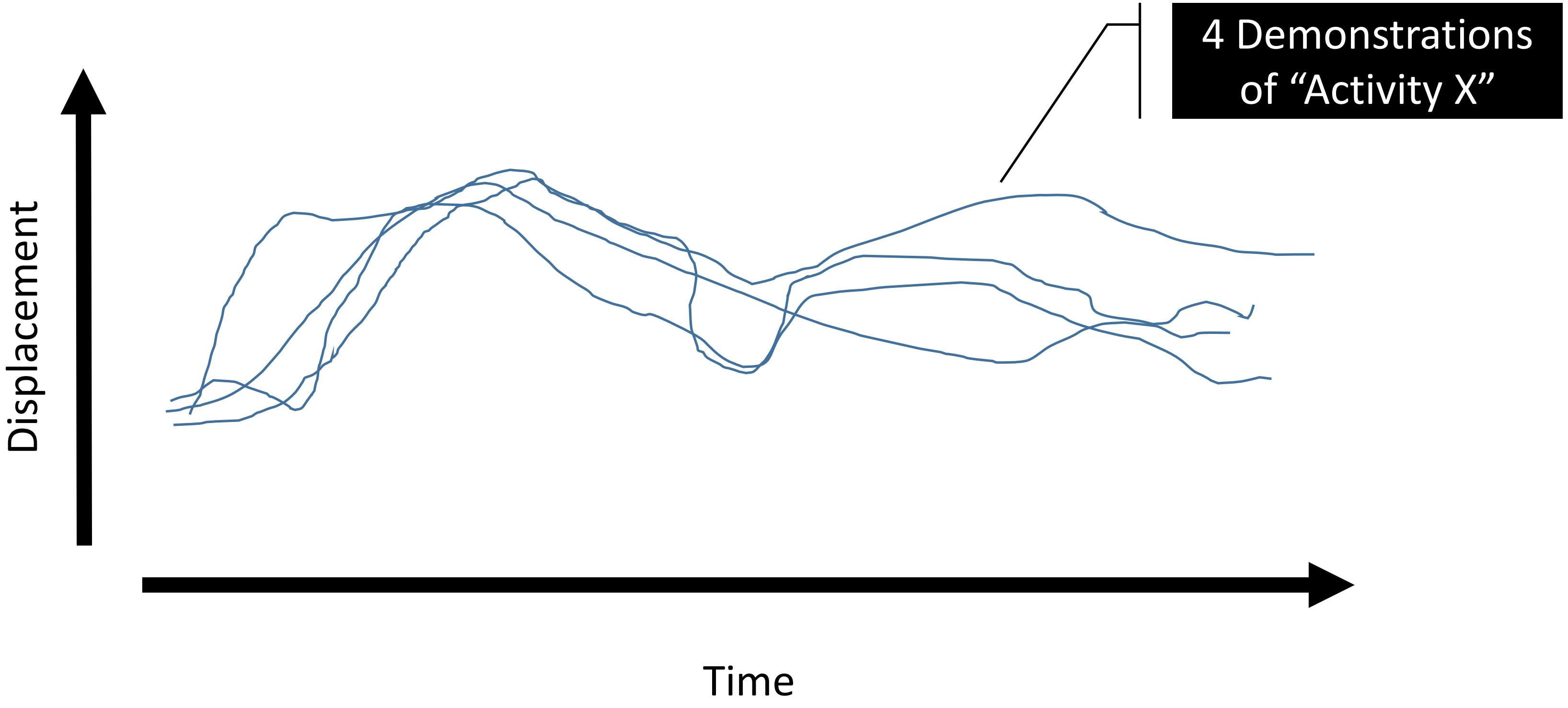
3/7	Thursday:	Paper presentations, Lit Review workshop
3/12	Tuesday:	No class, work on projects, start Intro/Related Work writeup
3/14	Thursday:	No class, work on projects, start Intro/Related Work writeup
3/19	Tuesday:	Guest Lecture - Dr. Dan Grollman
3/21	Thursday:	Inverse Reinforcement Learning and ROS
3/26	Tuesday:	Spring Break
3/28	Thursday:	Spring Break
4/2	Tuesday:	ROS, Computer Vision and Robot Control
4/4	Thursday:	HRI 2019 Papers, Evaluation Workshop
4/9	Tuesday:	Explainable AI and In-progress Project Presentations
4/11	Thursday:	Explainable AI and XAI Papers
4/16	Tuesday:	Reinforcement Learning
4/18	Thursday:	Reinforcement Learning and RL Papers
4/23	Tuesday:	Guest Lecture – Dr. Alessandro Roncone

Activity Classification and Segmentation

Related Work

Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification

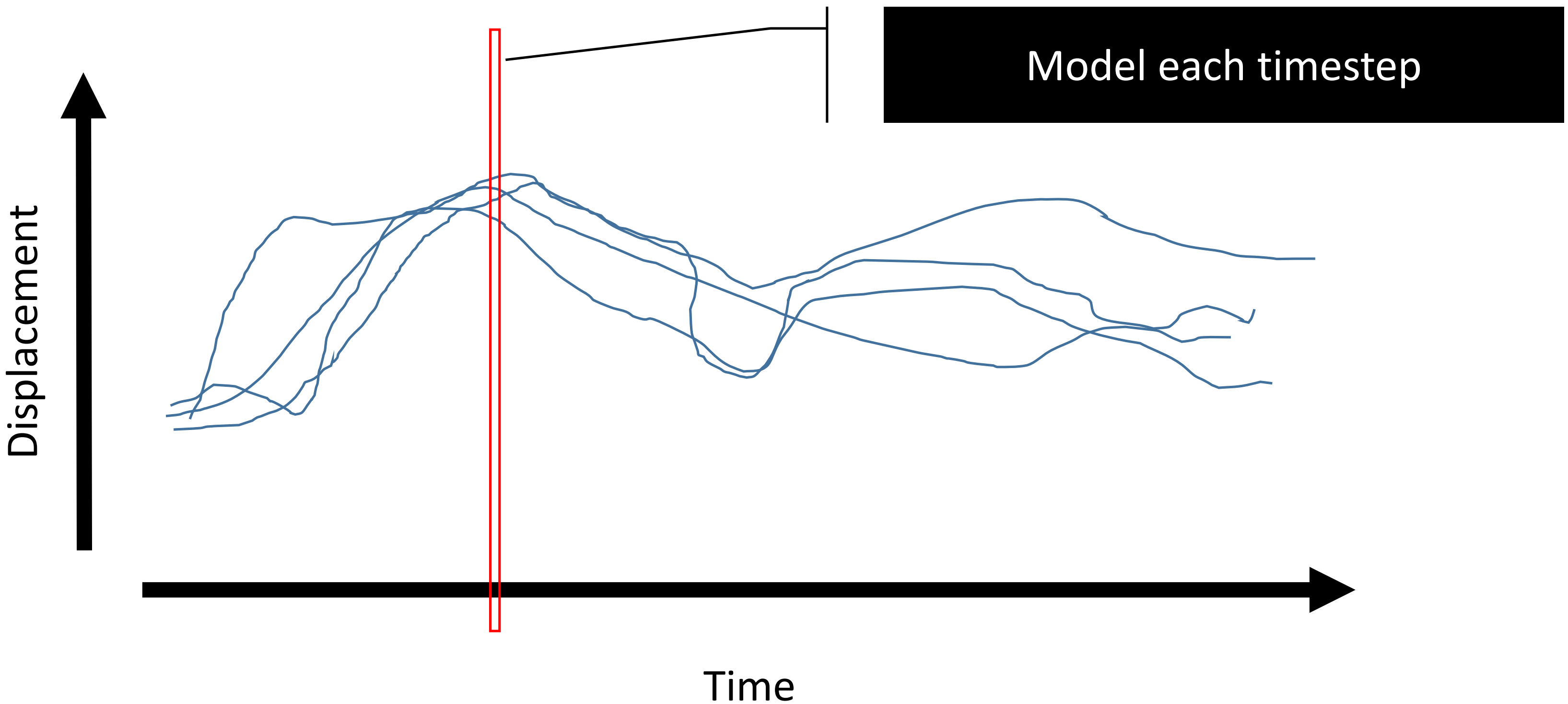
(Perez D'Arpino ICRA15)



Related Work

Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification

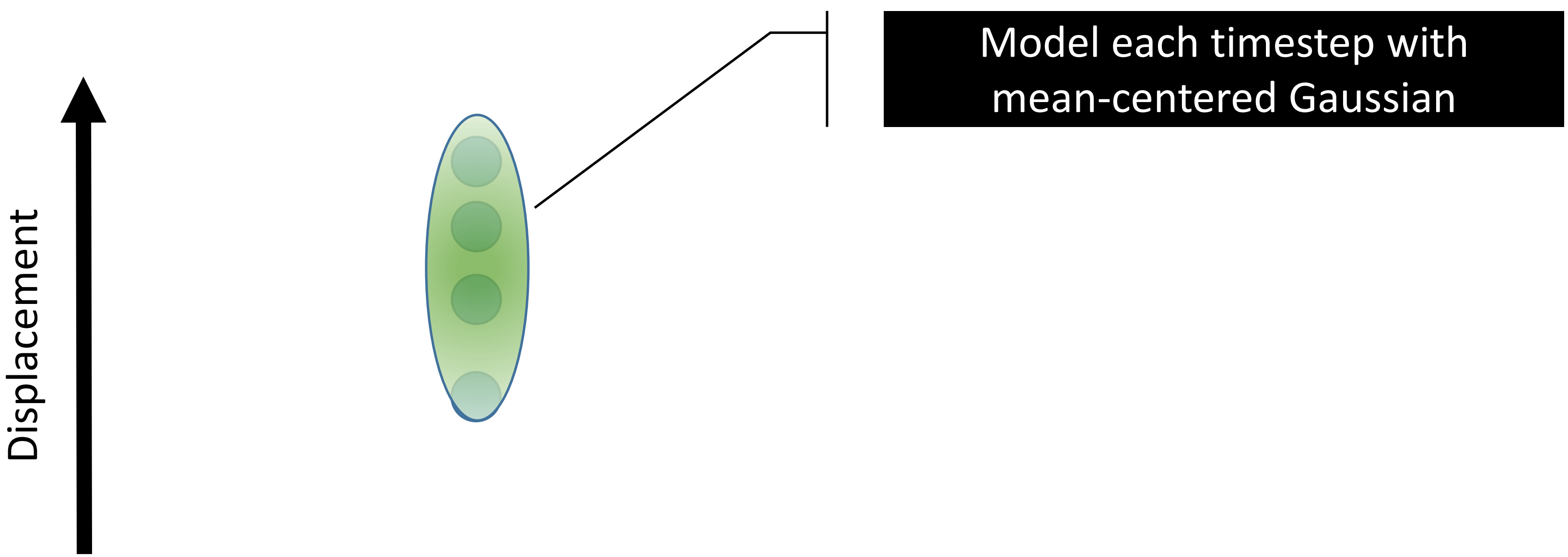
(Perez D’Arpino ICRA15)



Related Work

Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification

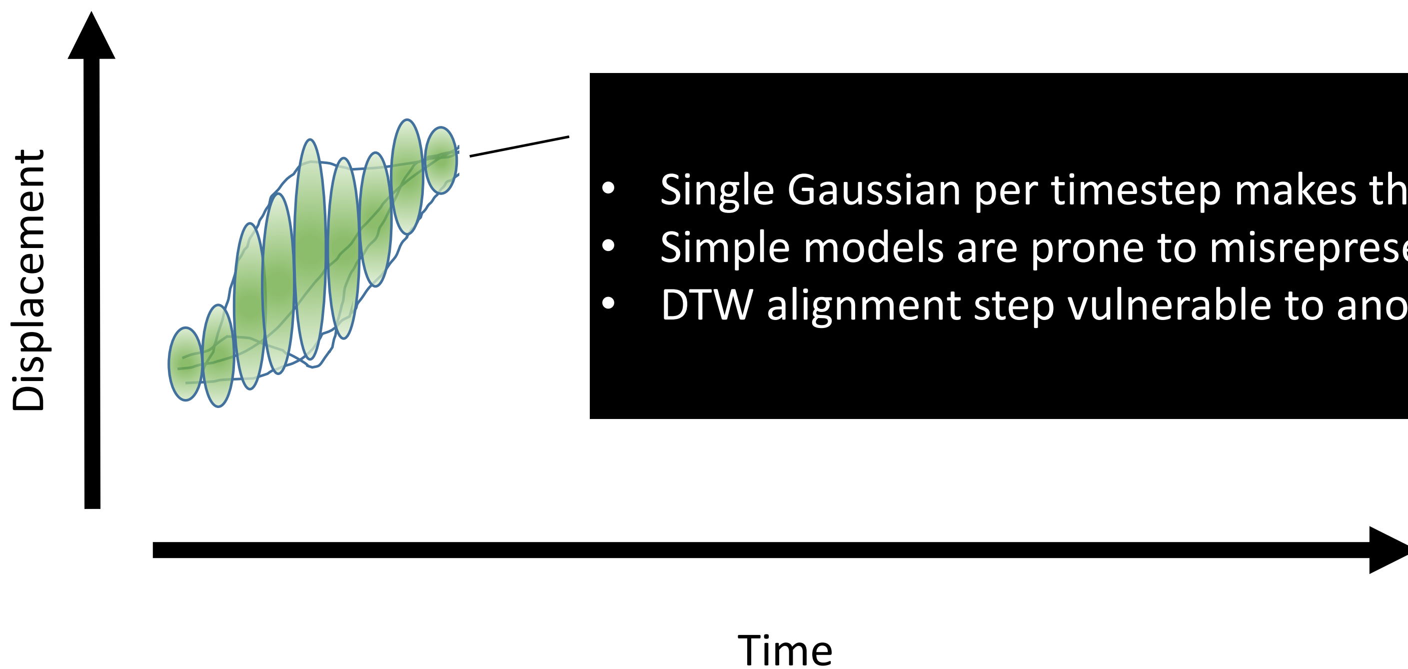
(Perez D'Arpino ICRA15)



Related Work

Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification

(Perez D'Arpino ICRA15)



- Single Gaussian per timestep makes this fast
- Simple models are prone to misrepresenting data
- DTW alignment step vulnerable to anomalies

Common Activity Classifier Pipeline

Training

Feature Extraction → Keyframe Clustering (Usually KNN) → Point to Keyframe Classifier (Usually SVM) → HMM trained on keyframe sequences

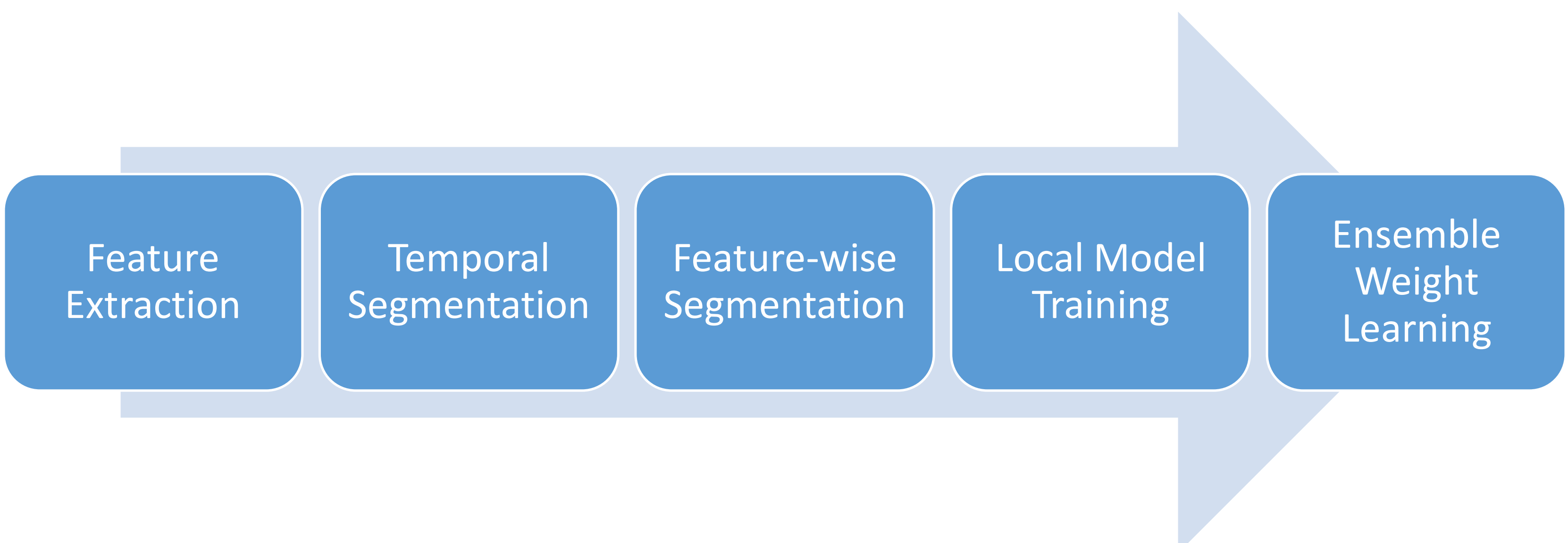
Testing

Feature Extraction → Keyframe Classification → HMM Likelihood Evaluation (Forward Algorithm) → Choose model with greatest posterior probability

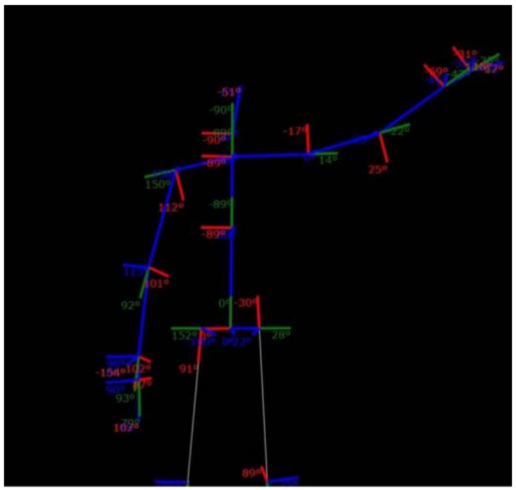
- P. Koniusz, A. Cherian, and F. Porikli, “Tensor representations via kernel linearization for action recognition from 3d skeletons.”
- Gori, J. Aggarwal, L. Matthies, and M. Ryoo, “Multitype activity recognition in robot-centric scenarios,”
- E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, “A human activity recognition system using skeleton data from rgbd sensors.”
- L. Xia, C. Chen, and J. Aggarwal, “View invariant human action recognition using histograms of 3d joints.”

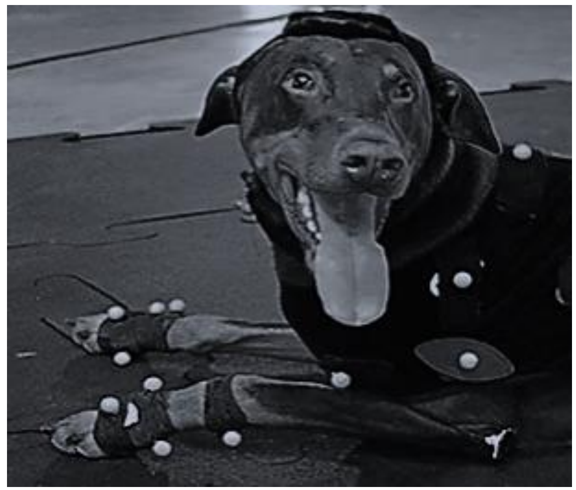
Rapid Activity Prediction Through Object-oriented Regression (RAPTOR)

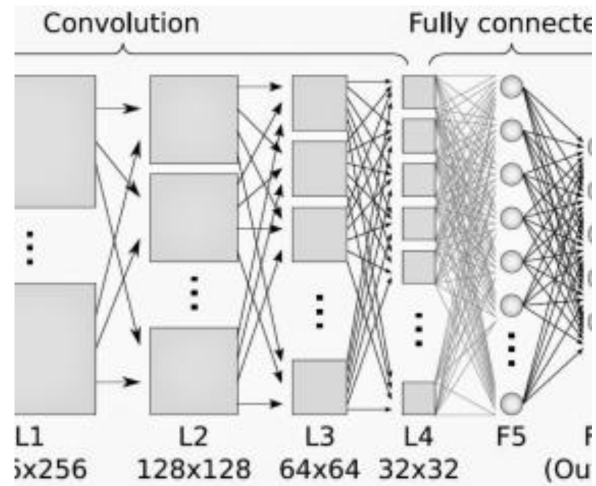
A highly parallel ensemble classifier that is resilient to temporal variations



Activity Model Training Pipeline







Kinect Skeletal Joints

VICON Markers

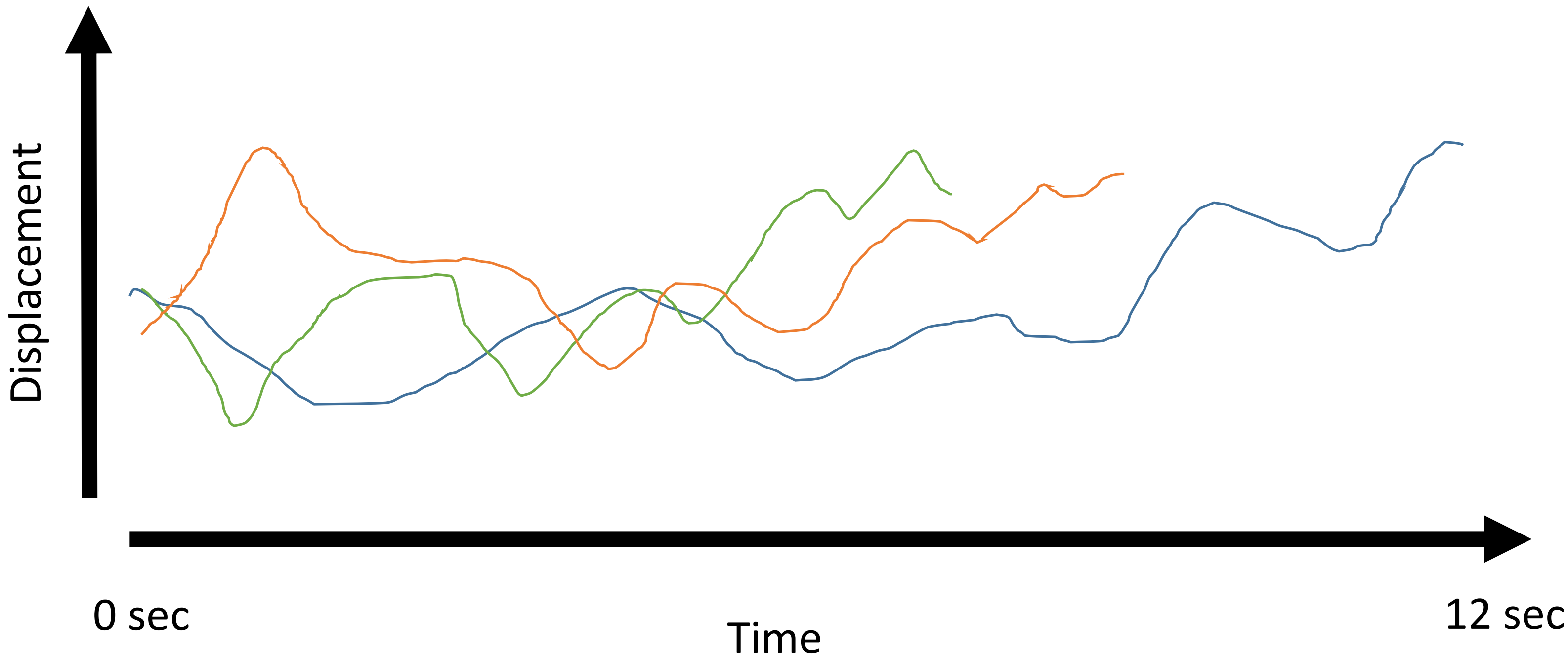
Learned Feature Extractor

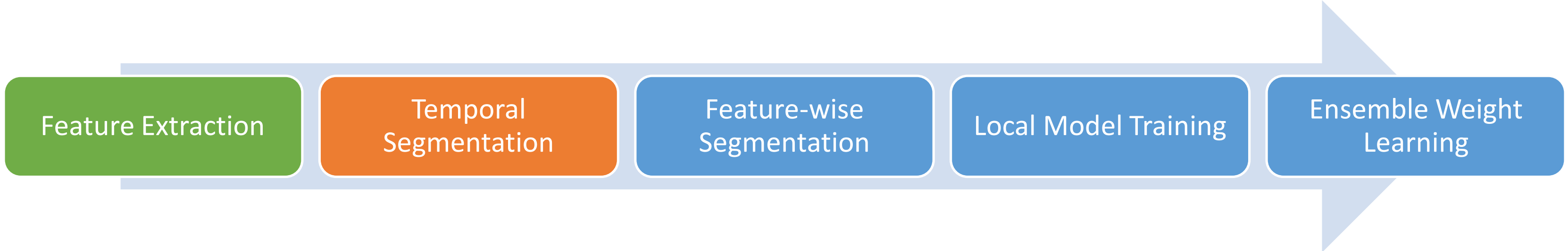
$$\begin{pmatrix} 1. & 0.04 & -0.67 & -0.4 & -0.54 & -0.74 & -0.22 & -0.75 & -0.56 \\ 0.04 & 1. & 0.45 & 0.41 & -0.03 & -0.4 & -0.44 & -0.28 & 0.16 \\ -0.67 & 0.45 & 1. & 0.39 & 0.49 & 0.2 & -0.16 & 0.15 & 0.35 \\ -0.4 & 0.41 & 0.39 & 1. & 0.06 & 0.2 & 0.11 & 0.13 & 0.38 \\ -0.54 & -0.03 & 0.49 & 0.06 & 1. & 0.36 & 0.02 & 0.16 & 0.39 \\ -0.74 & -0.4 & 0.2 & 0.2 & 0.36 & 1. & 0.37 & 0.57 & 0.19 \\ -0.22 & -0.44 & -0.16 & 0.11 & 0.02 & 0.37 & 1. & 0.11 & -0.25 \end{pmatrix}$$

[Timestep x Feature] Matrix

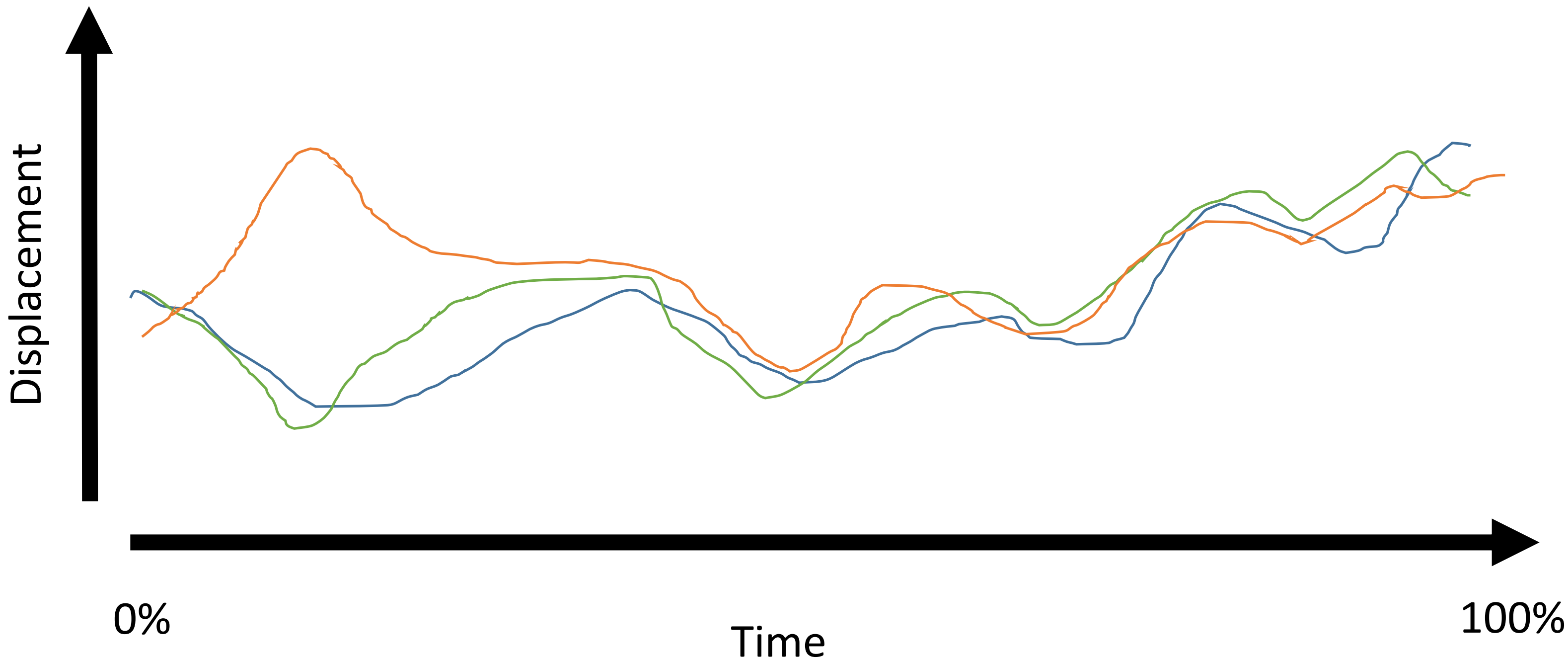
Feature Extraction → Temporal Segmentation → Feature-wise Segmentation → Local Model Training → Ensemble Weight Learning

Activity Model Training Pipeline



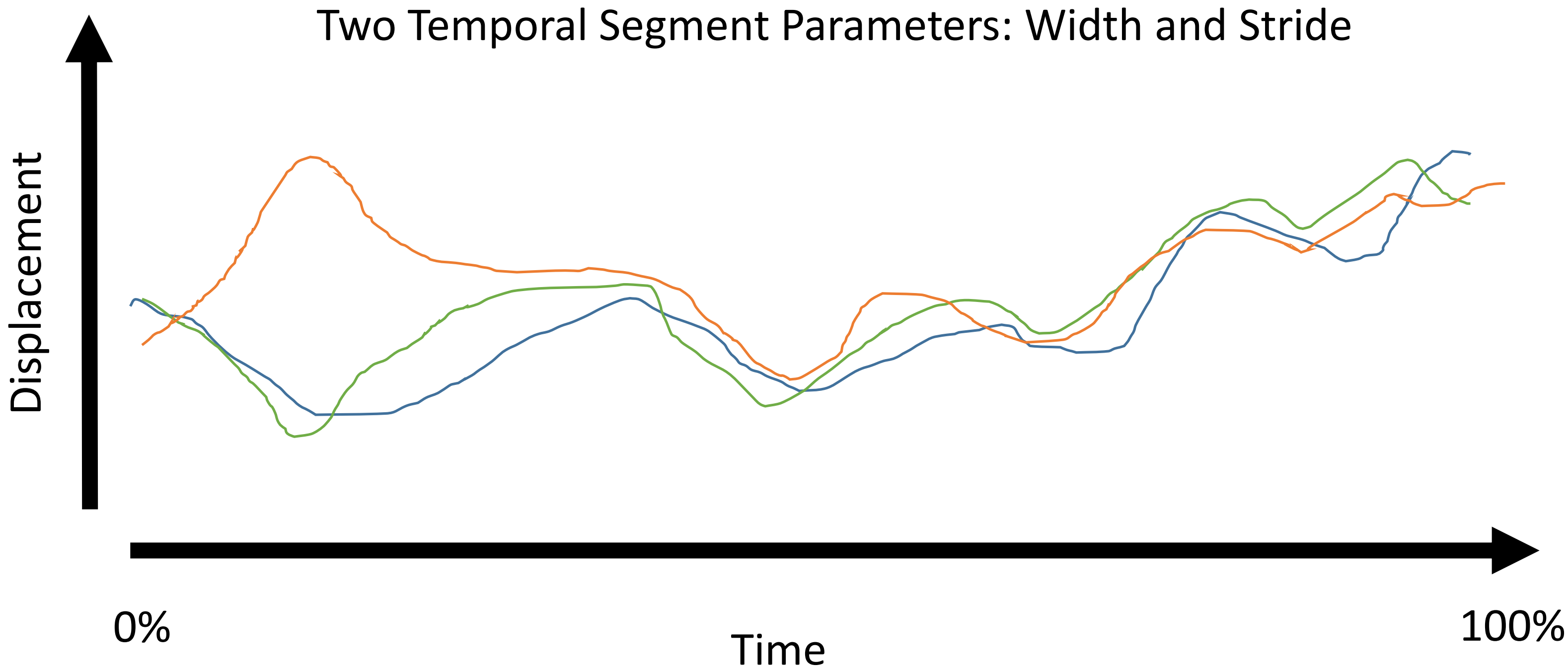


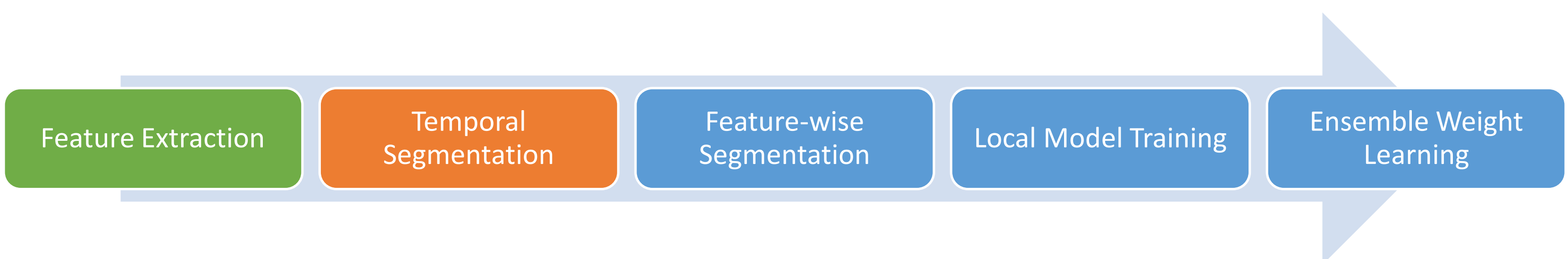
Activity Model Training Pipeline



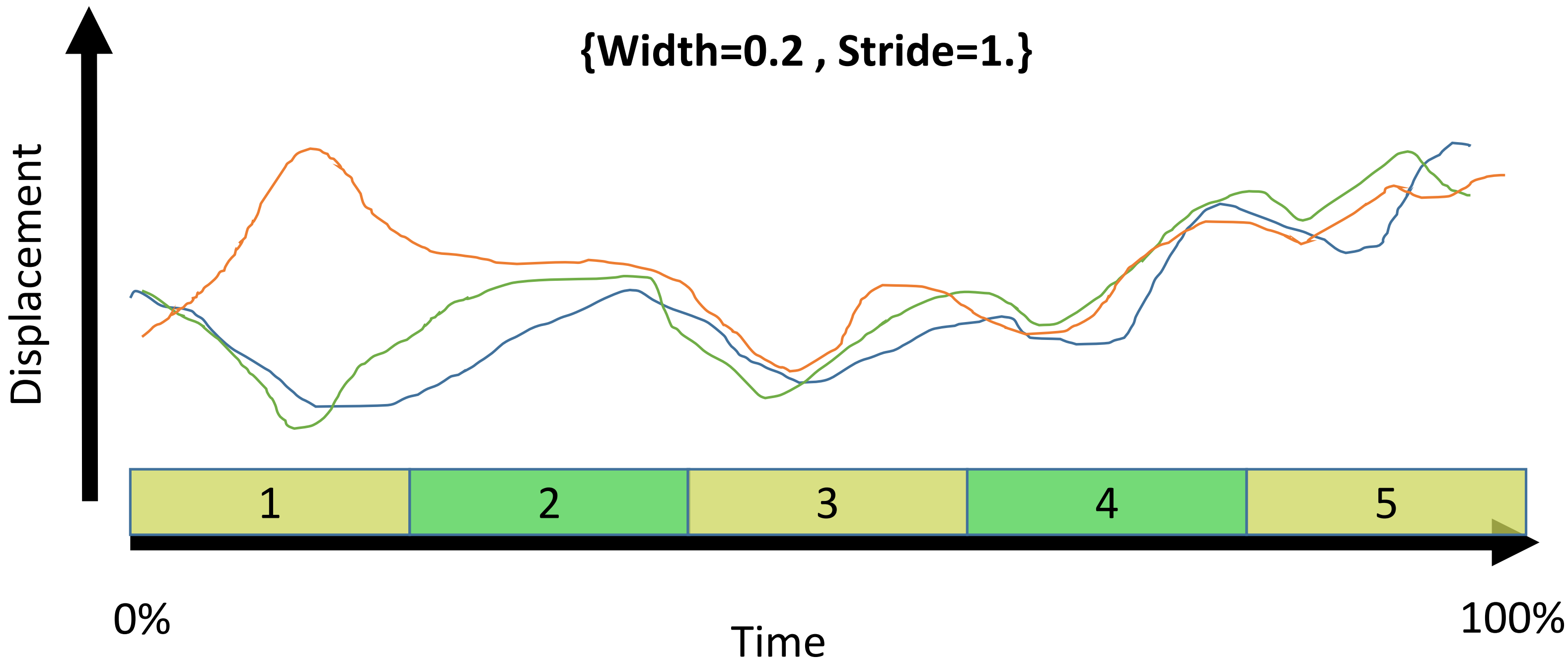
Feature Extraction → Temporal Segmentation → Feature-wise Segmentation → Local Model Training → Ensemble Weight Learning

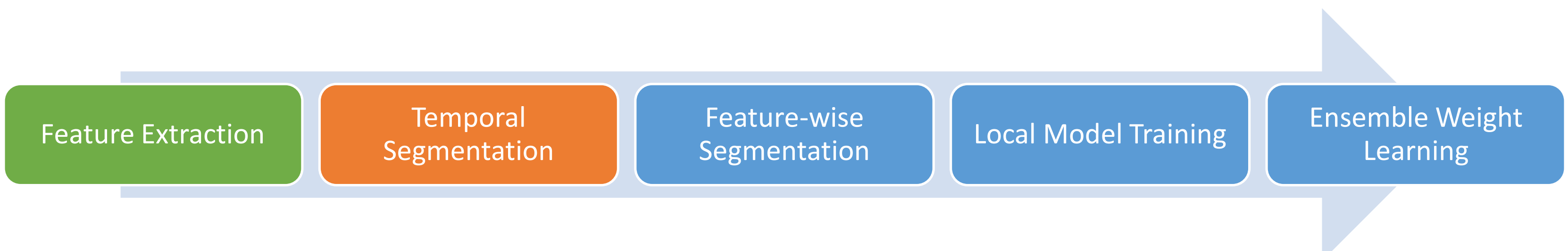
Activity Model Training Pipeline



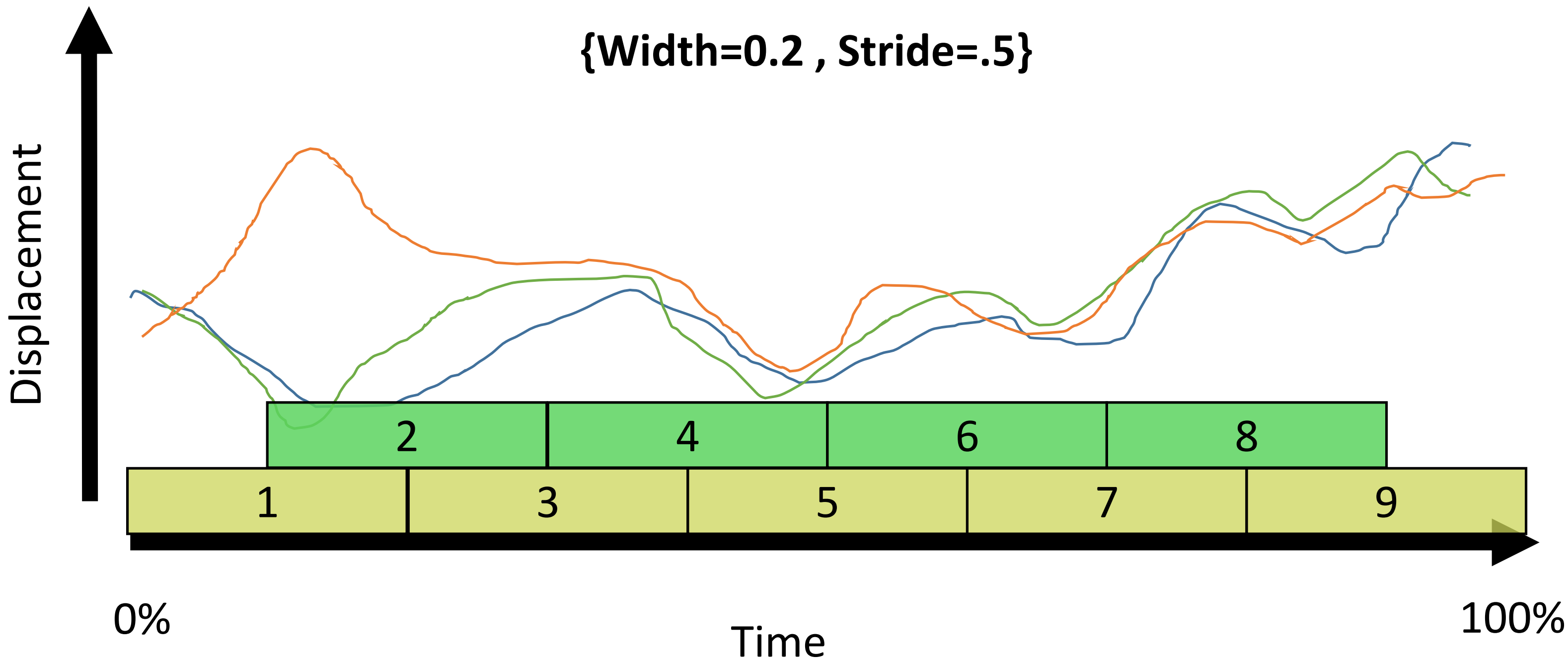


Activity Model Training Pipeline

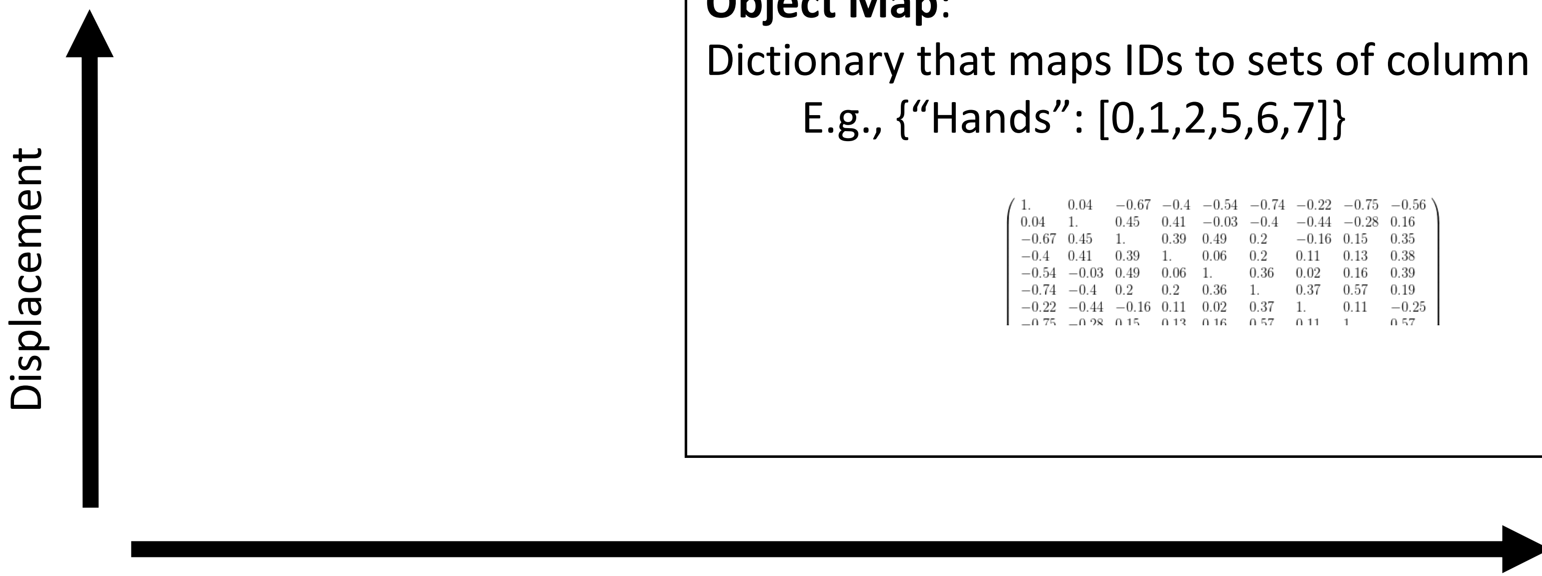




Activity Model Training Pipeline



Activity Model Training Pipeline

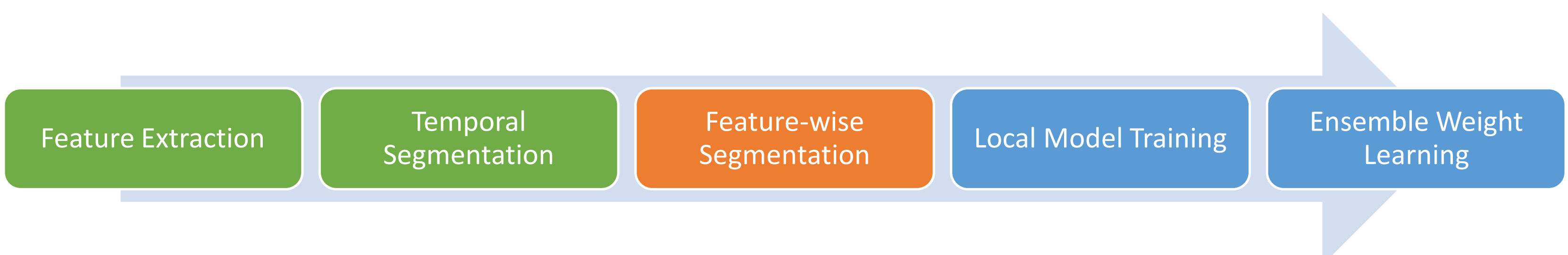


Object Map:

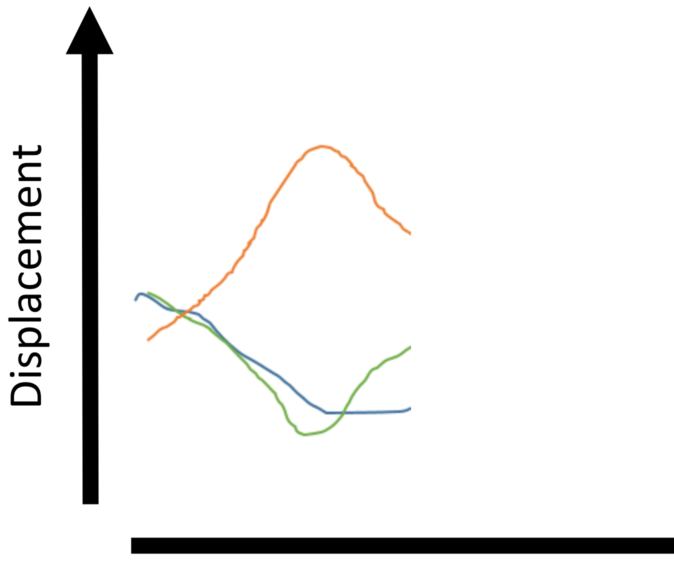
Dictionary that maps IDs to sets of column indices

E.g., {“Hands”: [0,1,2,5,6,7]}

$$\begin{pmatrix} 1. & 0.04 & -0.67 & -0.4 & -0.54 & -0.74 & -0.22 & -0.75 & -0.56 \\ 0.04 & 1. & 0.45 & 0.41 & -0.03 & -0.4 & -0.44 & -0.28 & 0.16 \\ -0.67 & 0.45 & 1. & 0.39 & 0.49 & 0.2 & -0.16 & 0.15 & 0.35 \\ -0.4 & 0.41 & 0.39 & 1. & 0.06 & 0.2 & 0.11 & 0.13 & 0.38 \\ -0.54 & -0.03 & 0.49 & 0.06 & 1. & 0.36 & 0.02 & 0.16 & 0.39 \\ -0.74 & -0.4 & 0.2 & 0.2 & 0.36 & 1. & 0.37 & 0.57 & 0.19 \\ -0.22 & -0.44 & -0.16 & 0.11 & 0.02 & 0.37 & 1. & 0.11 & -0.25 \\ -0.75 & -0.28 & 0.15 & 0.13 & 0.16 & 0.57 & 0.11 & 1. & 0.57 \end{pmatrix}$$



Activity Model Training Pipeline

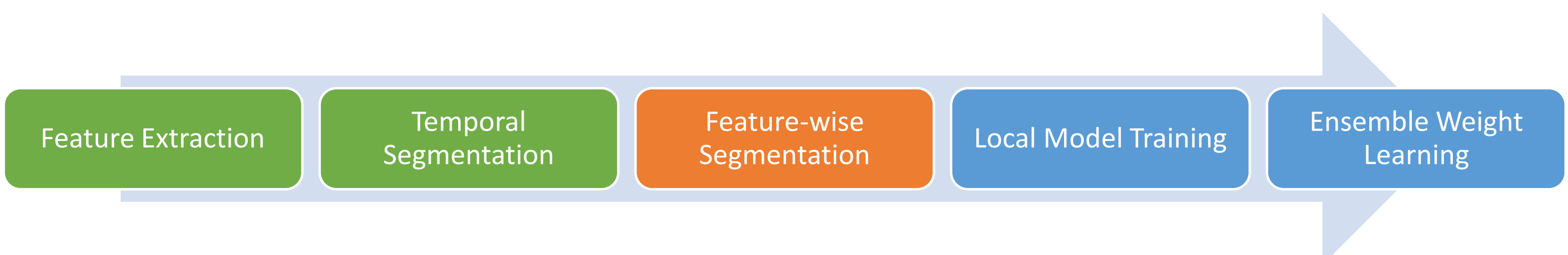


Within each temporal segment:

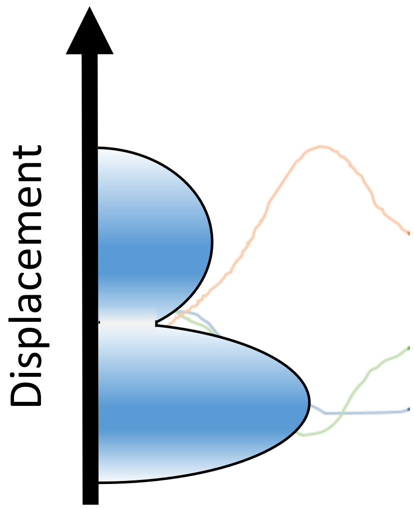
- Isolate columns of each demonstration trajectory according to (pre-defined) object map

$$\begin{pmatrix} 1. & 0.04 & -0.67 & & -0.74 & -0.22 & -0.75 \\ 0.04 & 1. & 0.45 & & -0.4 & -0.44 & -0.28 \\ -0.67 & 0.45 & 1. & & 0.2 & -0.16 & 0.15 \\ -0.4 & 0.41 & 0.39 & & 0.2 & 0.11 & 0.13 \\ -0.54 & -0.03 & 0.49 & & 0.36 & 0.02 & 0.16 \\ -0.74 & -0.4 & 0.2 & & 1. & 0.37 & 0.57 \\ -0.22 & -0.44 & -0.16 & & 0.37 & 1. & 0.11 \\ -0.75 & -0.28 & 0.15 & & 0.57 & 0.11 & 1. \end{pmatrix}$$

- Create local model for each object



Activity Model Training Pipeline



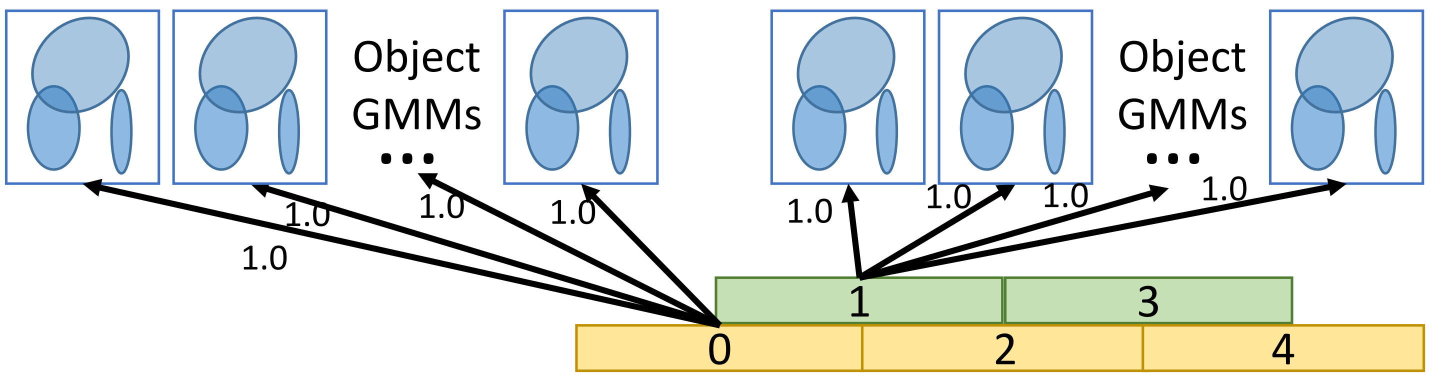
Within each temporal-object segment:

- Ignore temporal information for each data point
- Treat as general pattern recognition problem
- **Model the resulting distribution using a GMM**

Result: An activity classifier ensemble across objects and time!

Activity Model Training Pipeline

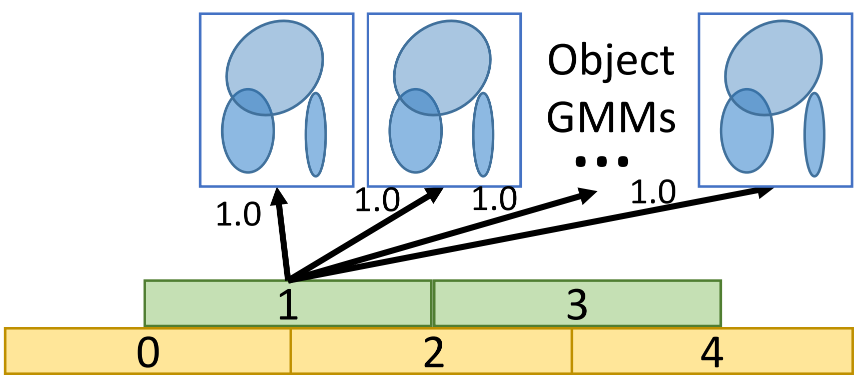
Need to find the most discriminative Object GMMs per time segment

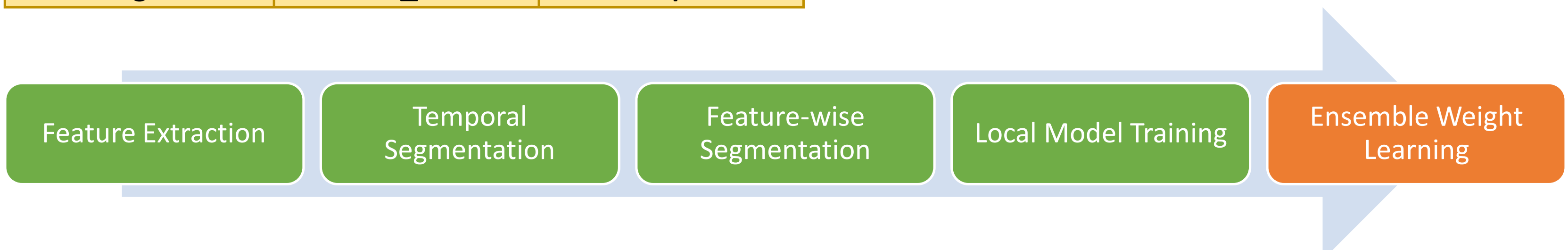


Activity Model Training Pipeline

Need to find the most discriminative Object GMMs per time segment

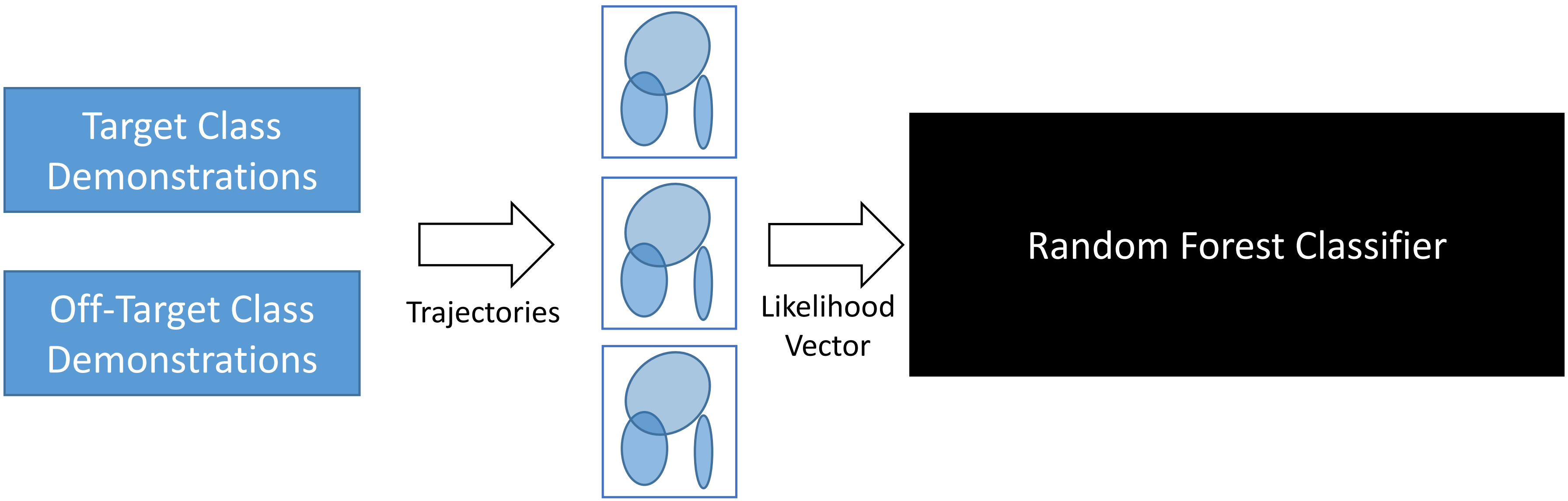
Random Forest Classifier





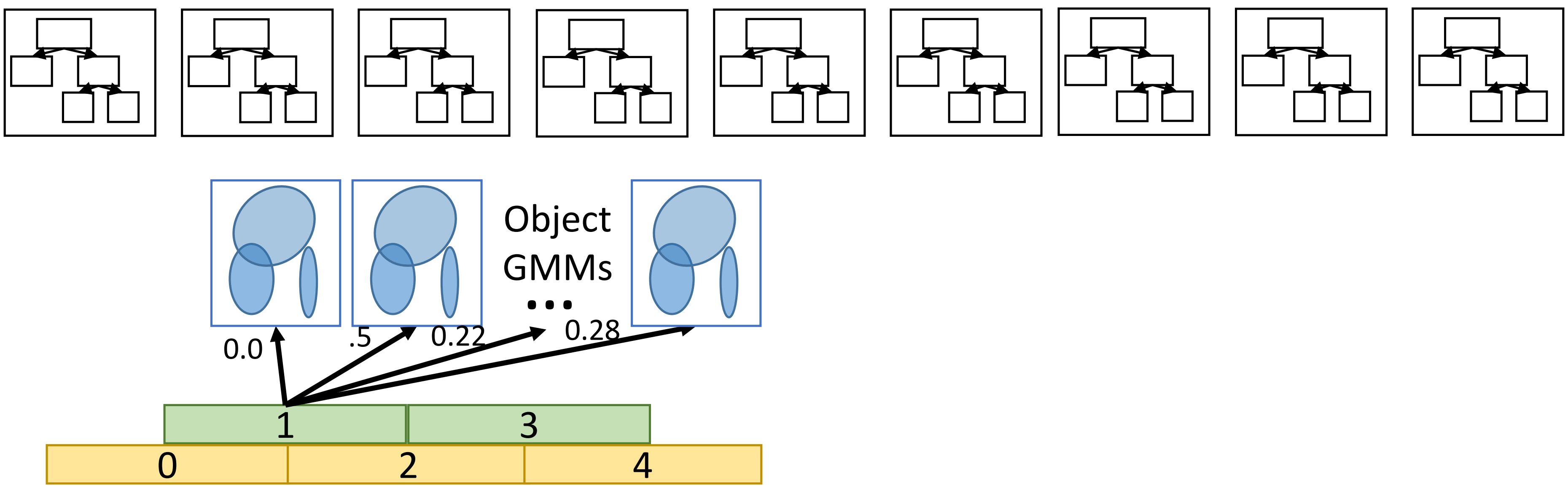
Activity Model Training Pipeline

Need to find the most discriminative Object GMMs per time segment



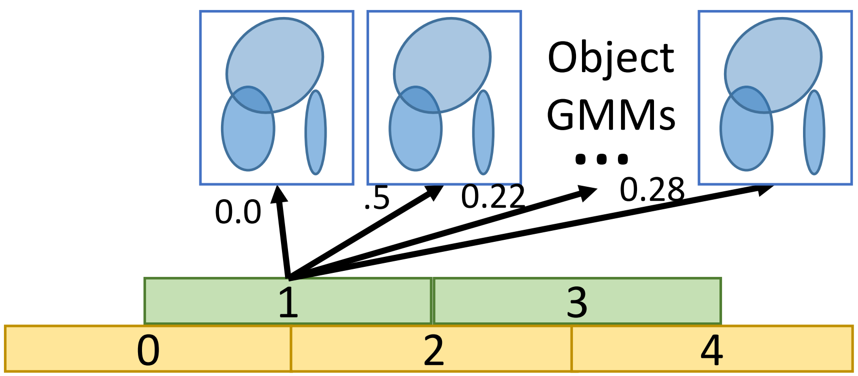
Activity Model Training Pipeline

- Choose top-N most discriminative features from the Random Forest classifier
- Weight each GMM proportional to its discriminative power



Activity Model Training Pipeline

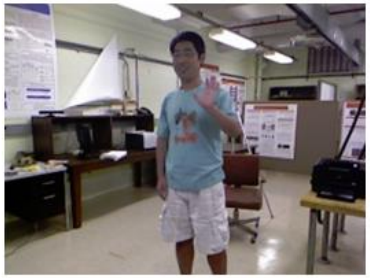
- Choose top-N most discriminative object-based classifiers
- Weight each object proportionally to its discriminative power



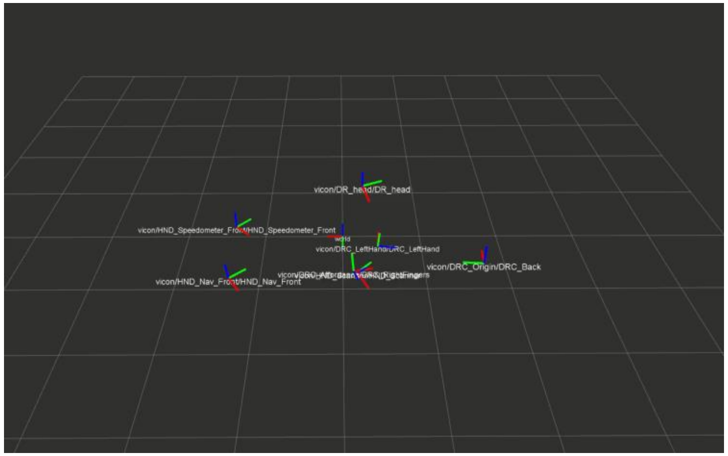
Result: Trained Highly Parallel Ensemble Learner with Temporal/Object-specific sensitivity

Results: Three Datasets

- **UTKinect** publicly available benchmark (Kinect Joints)
- **Dynamic** Actor Industrial Manufacturing Task (Joint positions)
- **Static** Actor Industrial Manufacturing Task (Joint positions)



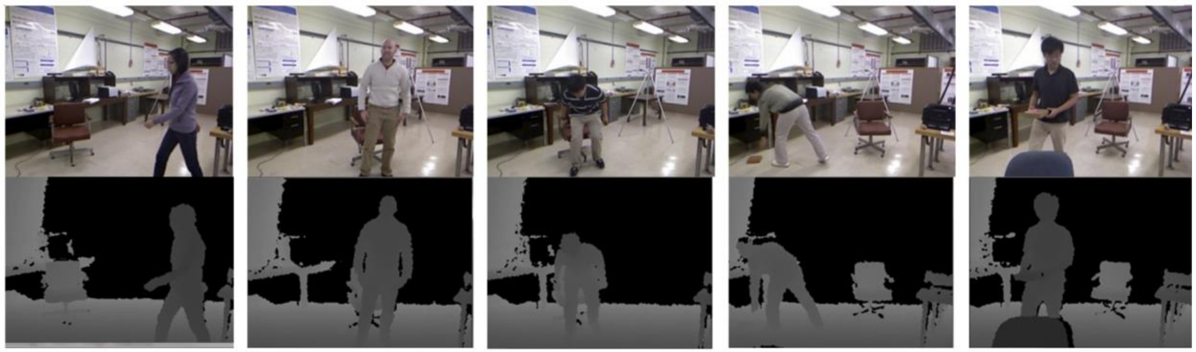
UTKinect



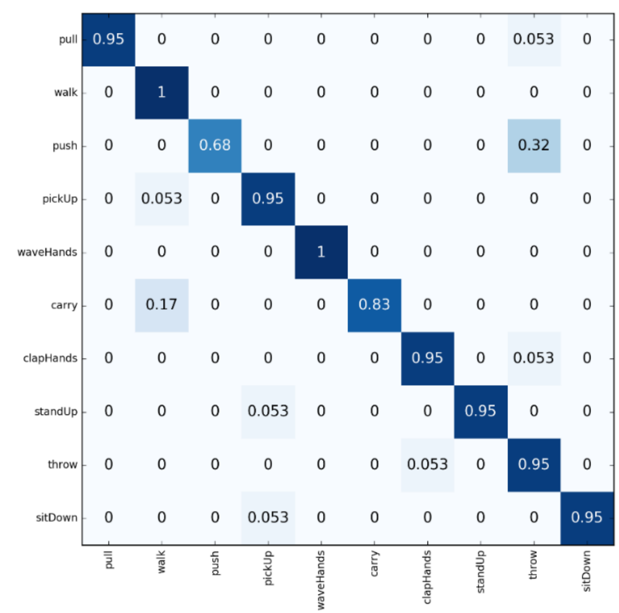
Automotive Final Assembly

Sealant Application

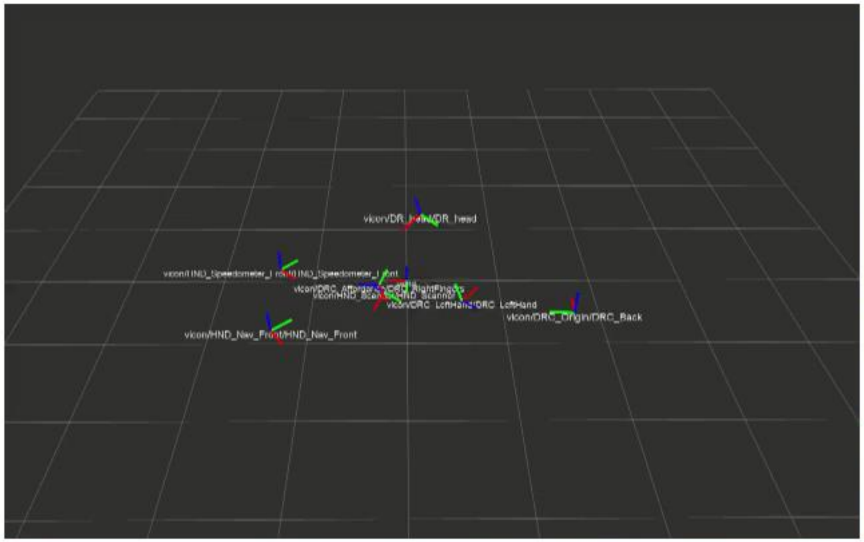
Recognition Results: UTKinect-Action3D

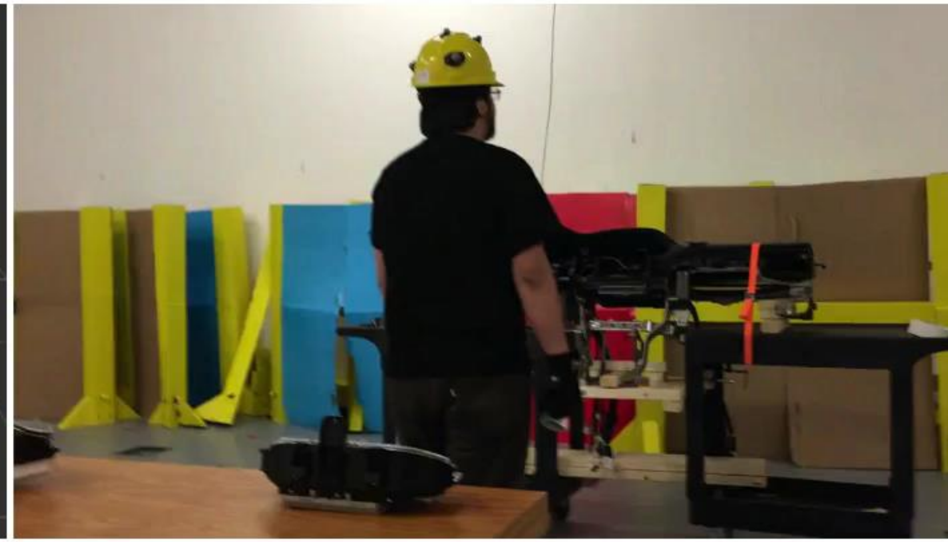


Real-time UTKinect Activity Recognition Accuracy	
Classifier	**Accuracy**
Slama et al. (2015) [21]	88.5%
Chrungoo et al. (2014) [18]	89.45%
Xia et al. (2012) [11]	90.9%
Wang et al. (2015) [24]	90.9%
Devanne et al. (2013) [20]	91.5%
RAPTOR (proposed method)	**92.1%**



Results: Online Prediction

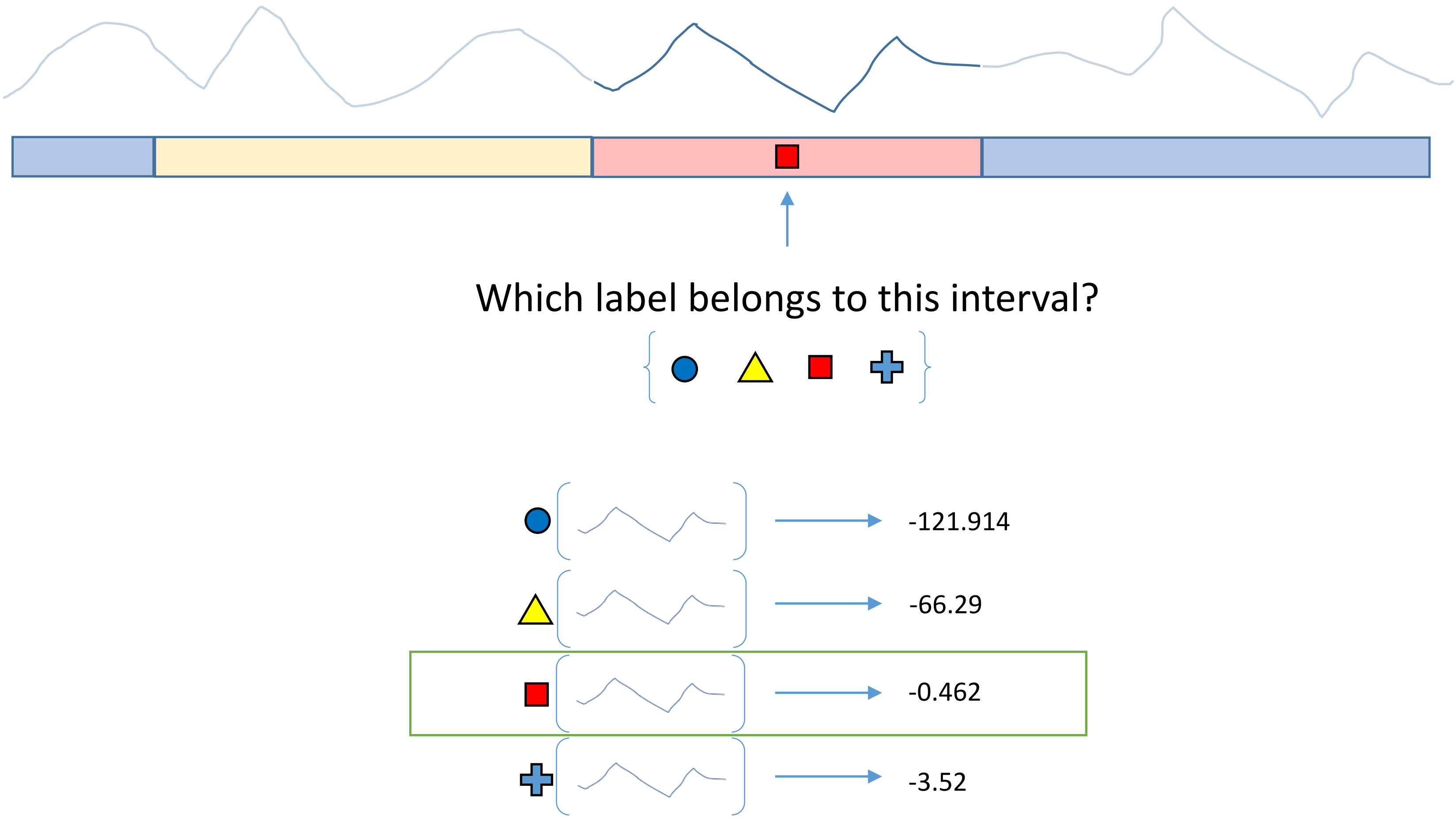




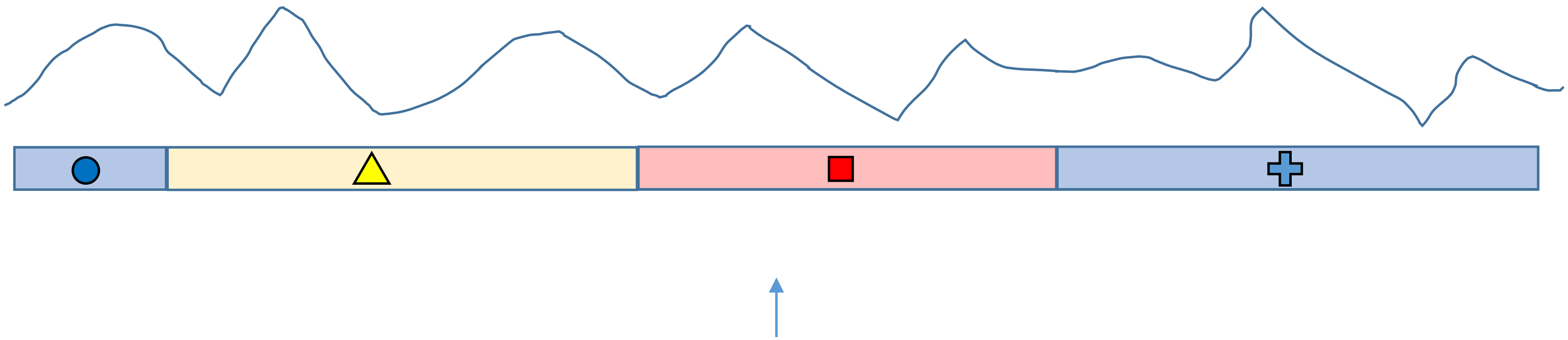
```
Elapsed Time: 0.1sec      Classified activity move_to_dash with likelihood 0.84128       Ground Truth: None
Elapsed Time: 0.13sec     Classified activity move_to_dash with likelihood 0.84811       Ground Truth: None
Elapsed Time: 0.17sec     Classified activity move_to_dash with likelihood 0.86419       Ground Truth: None
Elapsed Time: 0.2sec      Classified activity move_to_dash with likelihood 0.867         Ground Truth: None
Elapsed Time: 0.23sec     Classified activity move_to_dash with likelihood 0.95099       Ground Truth: None
```

RAPTOR Online Activity Prediction Accuracy				
Dataset	**25%**	**50%**	**75%**	**100%**
UTKinect	79.4%	83.1%	84.7%	92.1%
Static-Reach	69.7%	77.2%	93.8%	97.5%
Dynamic-AutoFA	91.7%	88.1%	90.5%	92.0%

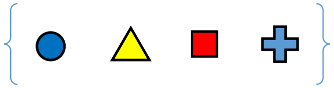
Classification vs. Segmentation



Classification vs. **Segmentation**



What are the right intervals?
Which intervals should get labels?
Which labels should be where?



A Naïve Changepoint Detection Approach

<u>Scenario</u>

Duration: 2700 frames – 1.5 minutes of data
Classifiers: 11 – Avg run-time of 0.2s each

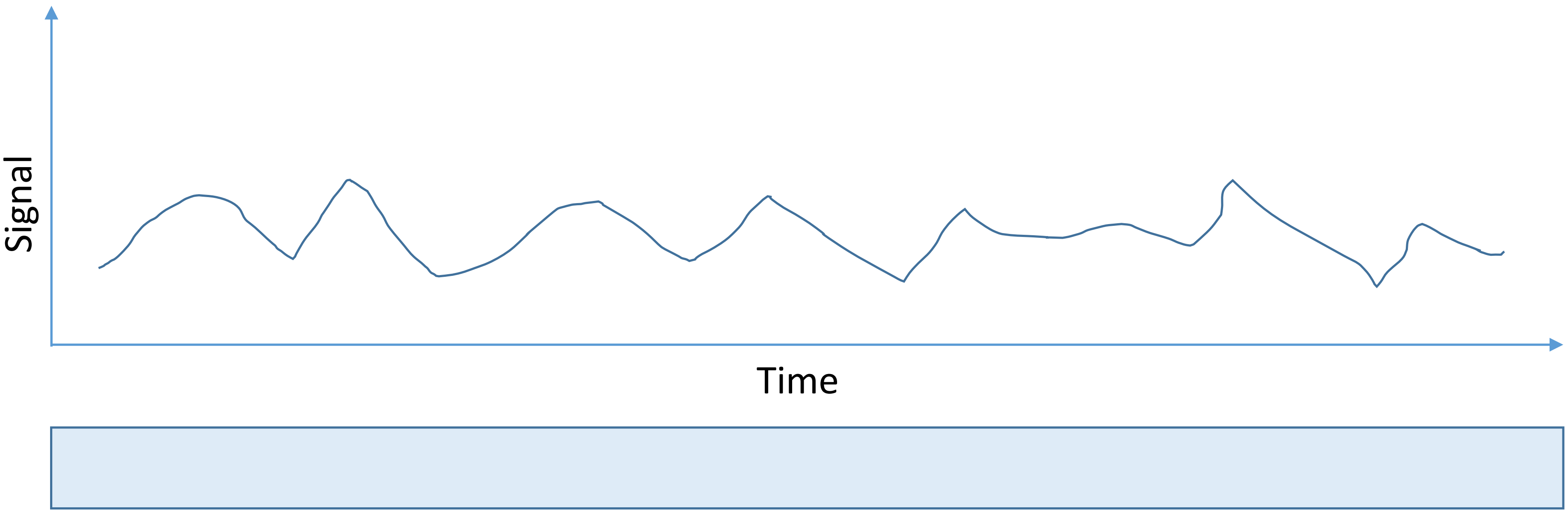
IDEA: Run every activity classifier over every possible segment

- Given n frames:
  - For every interval *q* in the range [0, n]:
    - Evaluate each classifier on *q*
  - Sort results by likelihood
  - Assign class labels to uncovered intervals from highest likelihood classifications until no unlabeled frames remain
- Return timeline (list of intervals)

$2700^2 * 0.2 = 1458000\text{sec}$
~16.88 days

Classifiers must be ideal (sensitive to trajectory length, non-overlapping, comparable tolerance to noise, etc.)

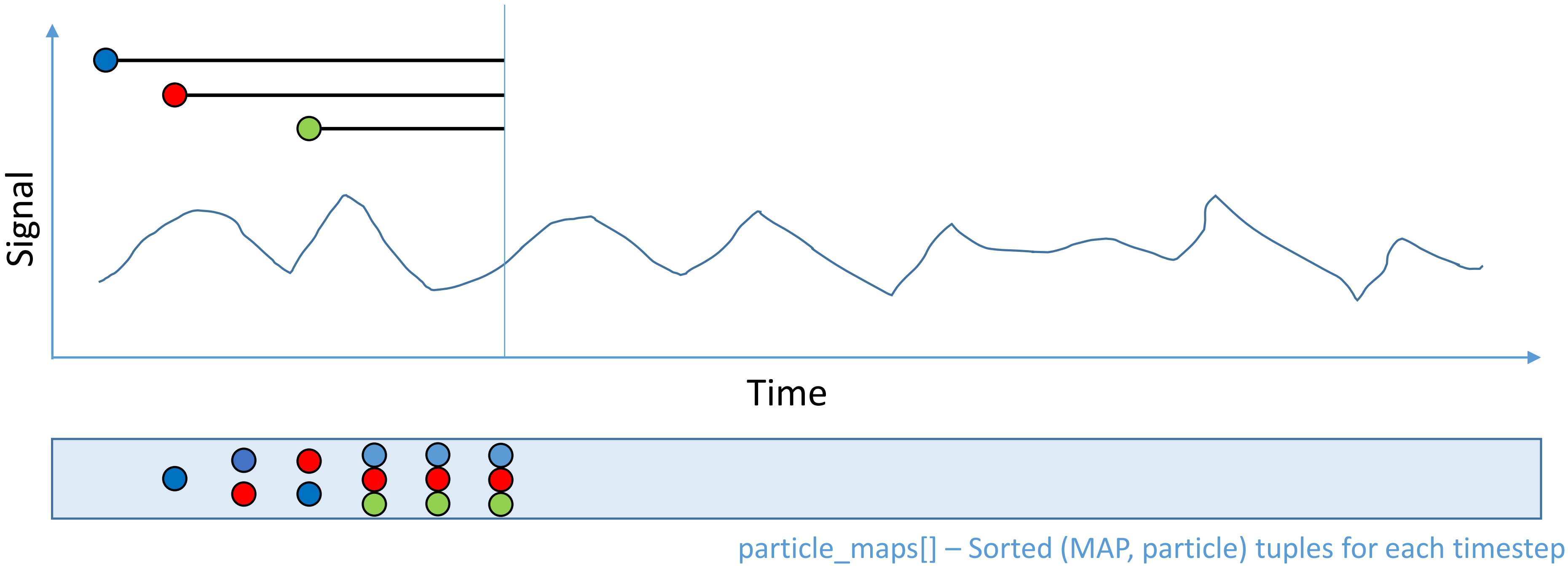
Particle Filtering for Changepoint Detection



particle_maps[] – Sorted (MAP, particle) tuples for each timestep

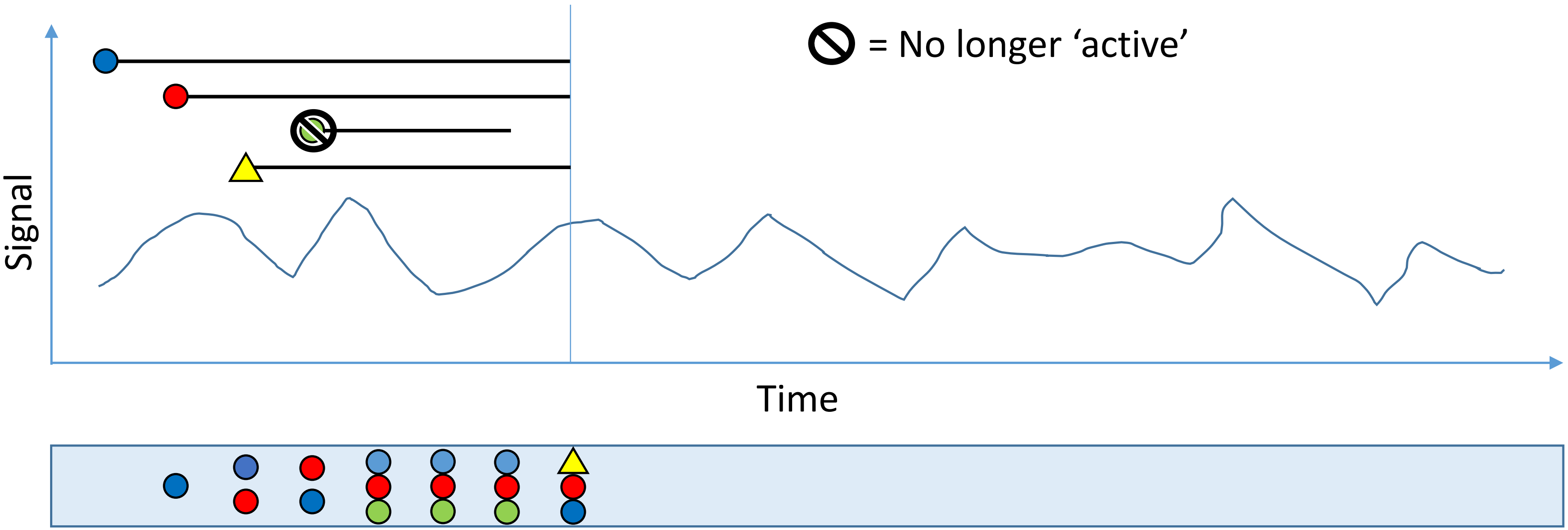
- At each time step *t*:
  - Create new particles for all eligible classes
    - *start_time* = *t* – *minimum_class_duration*
    - *prev_interval* = particle with highest MAP estimate in best[*start_time*]
  - Evaluate existing particles' likelihoods over the interval [*p.start_time*, *t*] and store as (likelihood, p) tuples in *particle_maps*
  - Terminate stale particles

Particle Filtering for Changepoint Detection



- At each time step *t*:
  - Create new particles for all eligible classes
    - *start_time* = *t* – *minimum_class_duration*
    - *prev_interval* = particle with highest MAP estimate in best[*start_time*]
  - Evaluate existing particles' likelihoods over the interval [*p.start_time*, *t*] and store as (likelihood, p) tuples in *particle_maps*
  - Terminate stale particles

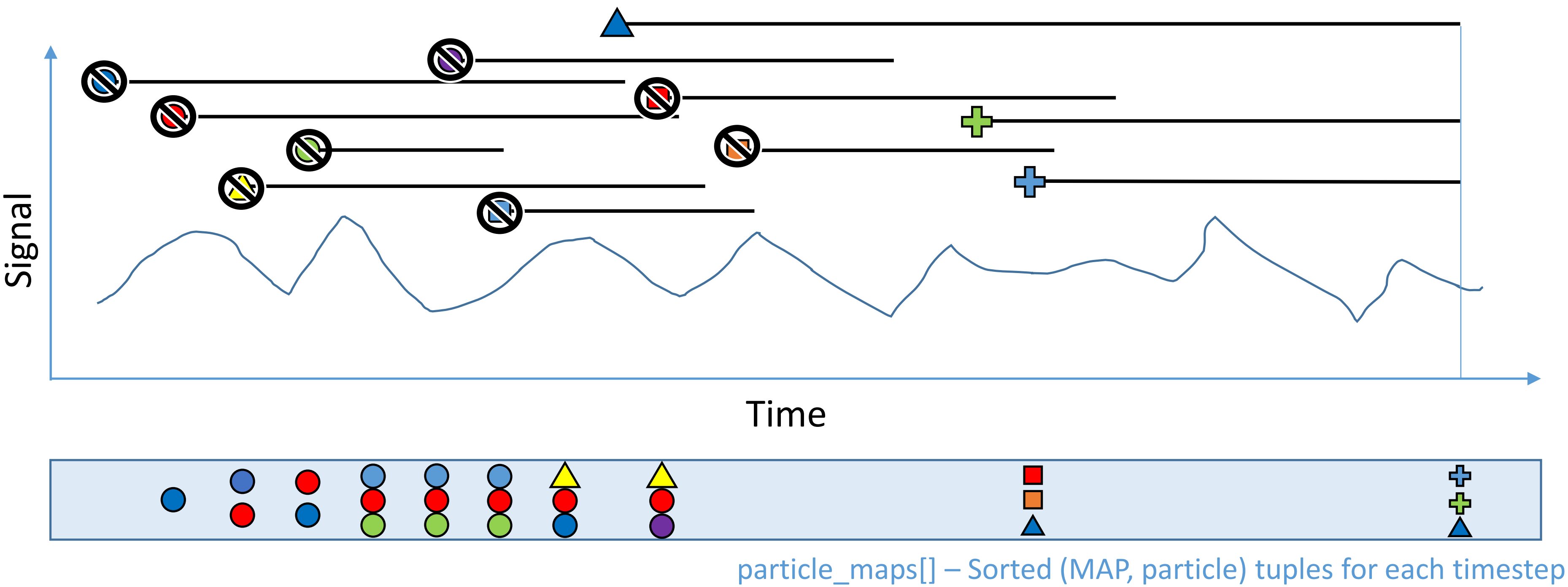
Particle Filtering for Changepoint Detection



particle_maps[] – Sorted (MAP, particle) tuples for each timestep

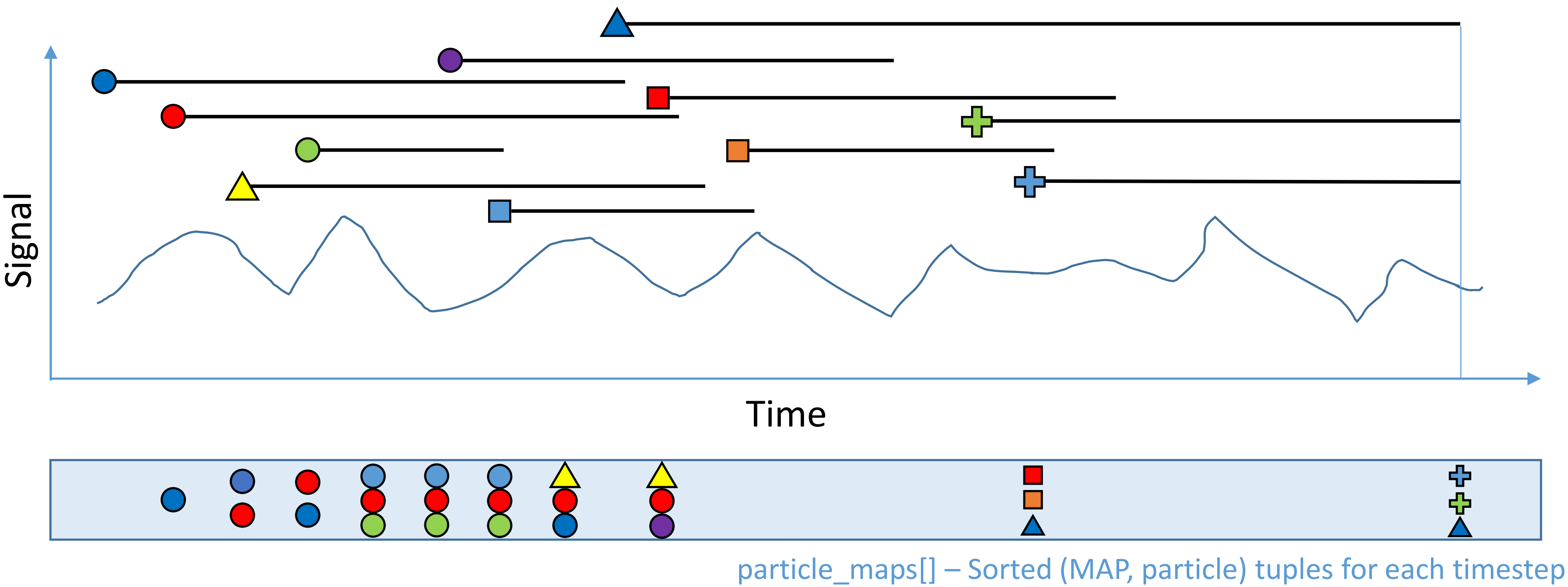
- At each time step *t*:
  - Create new particles for all eligible classes
    - *start_time* = *t* – *minimum_class_duration*
    - *prev_interval* = particle with highest MAP estimate in best[*start_time*]
  - Evaluate existing particles’ likelihoods over the interval [*p.start_time*, *t*] and store as (likelihood, p) tuples in *particle_maps*
  - Terminate stale particles

Particle Filtering for Changepoint Detection



- At each time step *t*:
  - Create new particles for all eligible classes
    - *start_time* = *t* – *minimum_class_duration*
    - *prev_interval* = particle with highest MAP estimate in best[*start_time*]
  - Evaluate existing particles' likelihoods over the interval [*p.start_time*, *t*] and store as (likelihood, p) tuples in *particle_maps*
  - Terminate stale particles

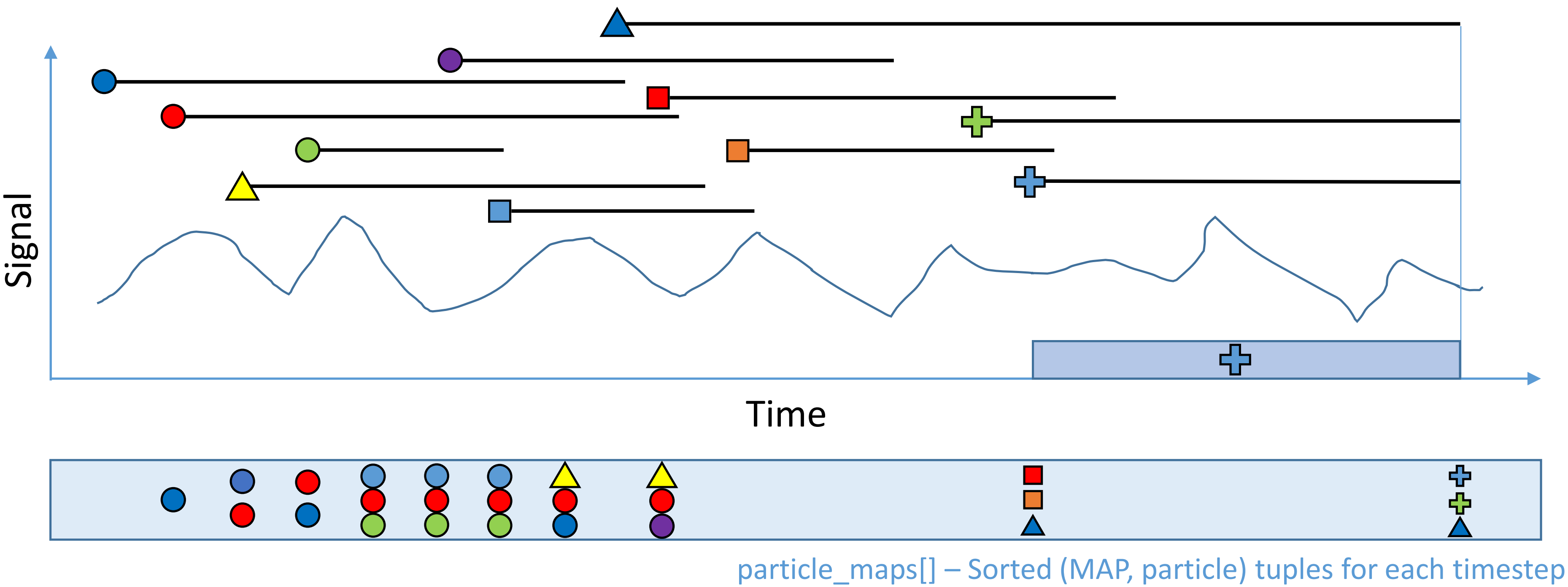
Particle Filtering for Changepoint Detection



To extract the most likely segmentation:

- *Set f = final frame index*
- While *f > 0* and *particle_maps[f] != None:*
    - Take best (MAP, particle) at particle_maps index *f*
    - Annotate segment [*shape_start*, *f*] with *shape_class*
    - Set *f = particle.start_time*

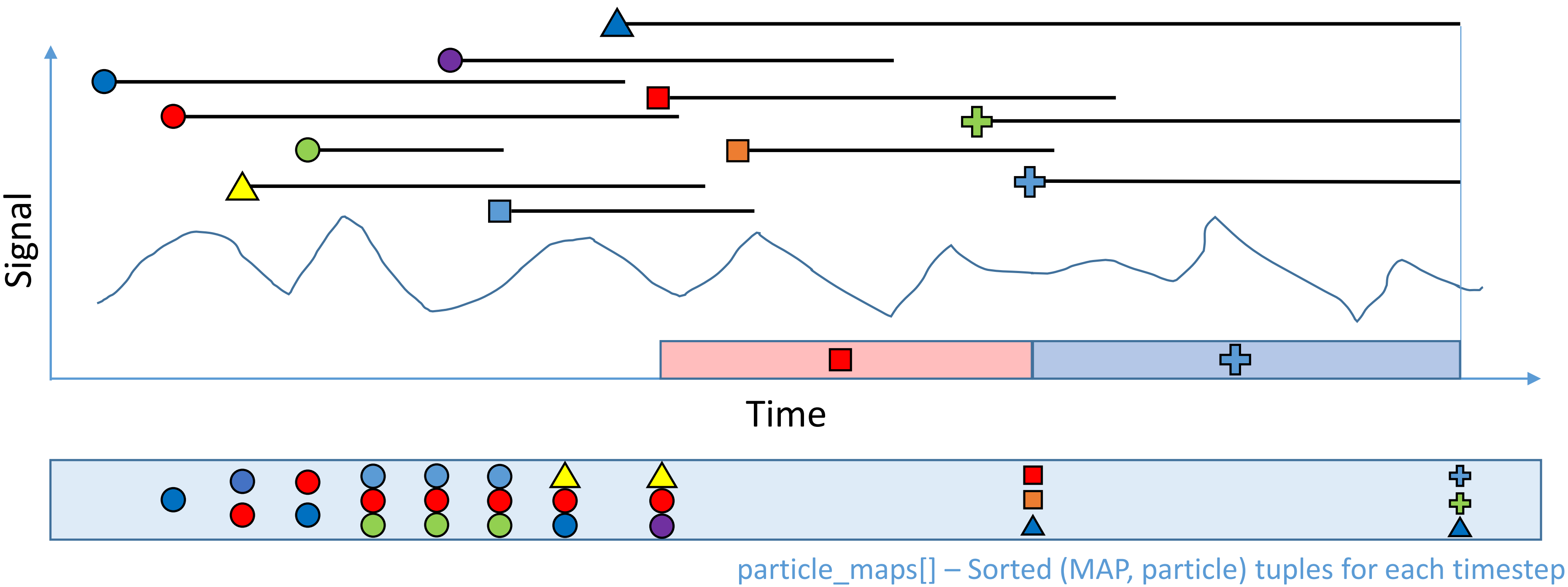
Particle Filtering for Changepoint Detection



To extract the most likely segmentation:

- *Set f = final frame index*
- While *f > 0* and *particle_maps[f] != None:*
  - Take best (MAP, particle) at particle_maps index *f*
  - Annotate segment [*shape_start*, *f*] with *shape_class*
  - Set *f = particle.start_time*

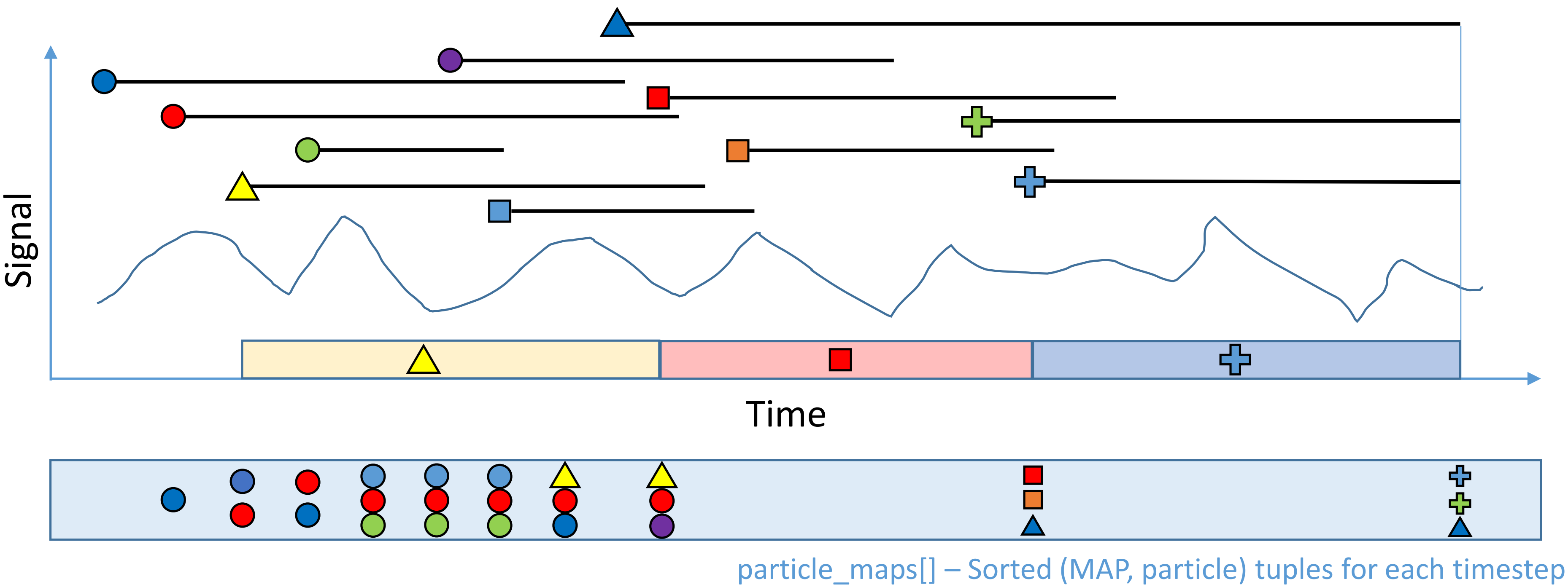
Particle Filtering for Changepoint Detection



To extract the most likely segmentation:

- *Set f = final frame index*
- While *f > 0* and *particle_maps[f] != None:*
    - Take best (MAP, particle) at particle_maps index *f*
    - Annotate segment [*shape_start*, *f*] with *shape_class*
    - Set *f = particle.start_time*

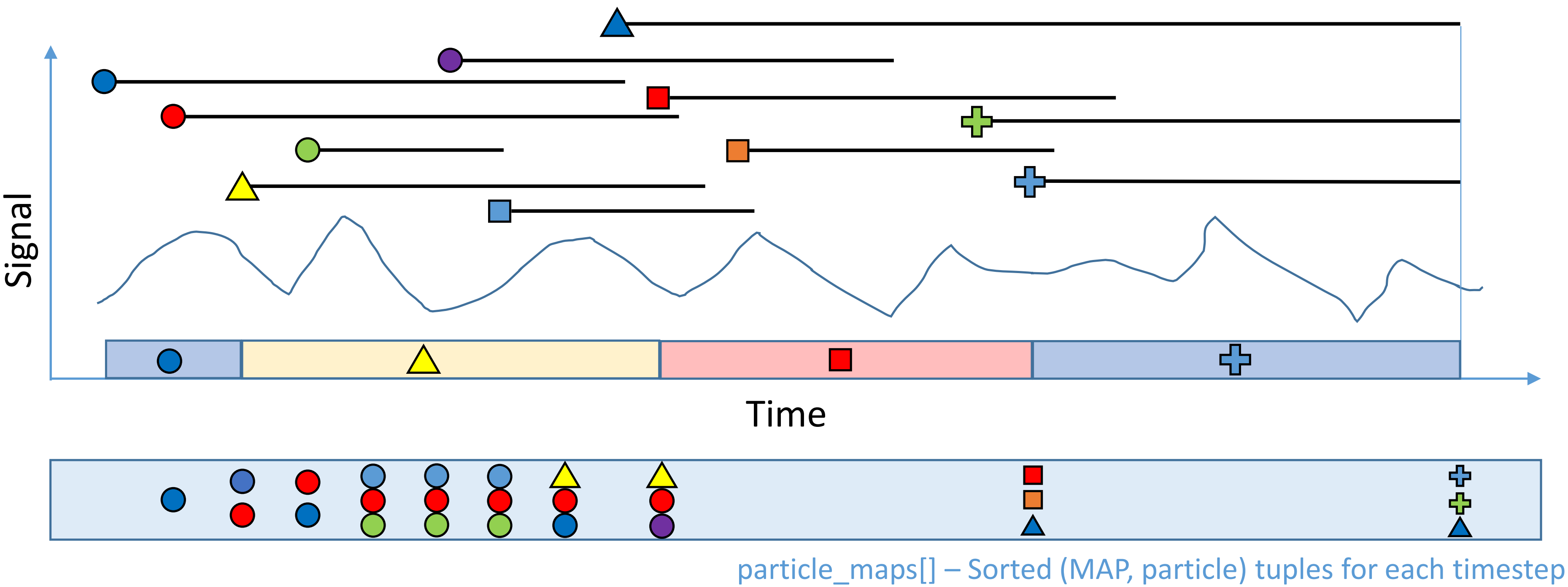
Particle Filtering for Changepoint Detection



To extract the most likely segmentation:

- *Set f = final frame index*
- While *f > 0* and *particle_maps[f] != None:*
  - Take best (MAP, particle) at particle_maps index *f*
  - Annotate segment [*shape_start*, *f*] with *shape_class*
  - Set *f = particle.start_time*

Particle Filtering for Changepoint Detection



To extract the most likely segmentation:

- *Set f = final frame index*
- While *f > 0* and *particle_maps[f] != None:*
    - Take best (MAP, particle) at particle_maps index *f*
    - Annotate segment [*shape_start*, *f*] with *shape_class*
    - Set *f = particle.start_time*

System Design Workshop (Part II)

Finish defining the major components of your project

If already done, begin designing your evaluation!

Deliverable: Block diagram with details for each component

(Powerpoint slide deck is generally the easiest format)

Submit on Moodle by Friday

1. Set up Git repo
2. Draw your system diagram
3. Describe your nodes' purposes
4. What are your inputs/outputs?
    - Group into ROS topics/message types
5. Outline your nodes' functionality
    - Increasingly descriptive pseudocode
6. Start turning pseudocode into real code

Summary