A Vision-Based Approach to Imitation Using Heterogeneous Demonstrators

Jeff Allen and John Anderson

Autonomous Agents Laboratory

University of Manitoba

Outline

- Imitation Learning
- Primitive Movements
- Recognizing Primitives from Vision
- Learning Architecture
- Learning from Multiple Demonstrators
- Evaluating Demonstrators

Imitation Learning

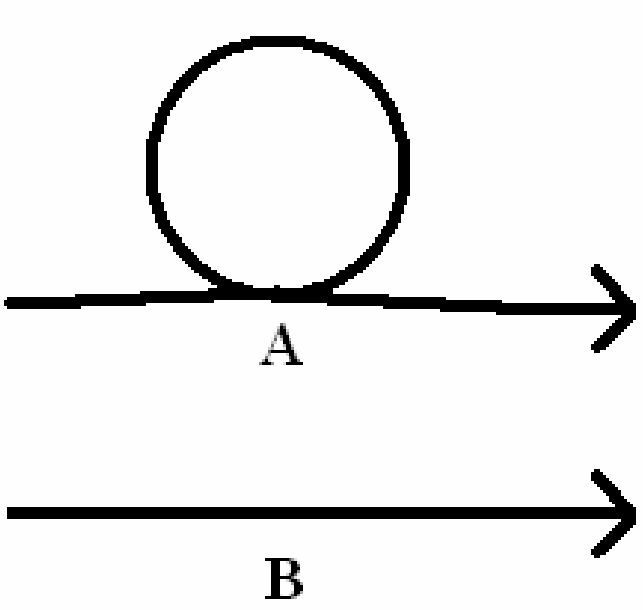
- Learning behaviours by observing the actions of others
  - Evidence of learning through imitation found in birds, primates, and humans
- Robots learn behaviours to complete tasks from demonstrations, rather than requiring separate programming for new tasks
- Most imitation learning research focuses on learning from a single demonstrator

Imitation Learning

- Leaves very little room for improvisation, since the imitator can only learn what it observes
- The imitator can learn undesirable behaviours: e.g., if the demonstrator performed a pointless loop when it should simply drive in a straight line
- With no demonstrators to compare this one to, the imitator would also pick up the inefficient looping behaviour
- Our work involves learning from multiple demonstrators of different physiologies, using global vision to record demonstrations, in a soccer domain

Imitation Learning

- If two behaviours result in the same state change, they can be compared



- A and B both result in the same state change.
- B is the obvious choice.
- Since B can achieve the same results as A, and more efficiently, A may be able to be ignored, unless B is impossible due to physiological differences

Imitation Learning

- Observed behaviours can be broken down into smaller actions
- Simpler actions define smaller state changes, making them easier to relate to the learning robot's own abilities
- These simple actions that compose behaviours are known as *primitives*, which may still have some abstract qualities
- Some previous work has used such primitives as *avoidance, following, foraging*, etc.
- In our work, we have defined the primitives to be the most atomic movements available to the imitating robot

Primitive Movements

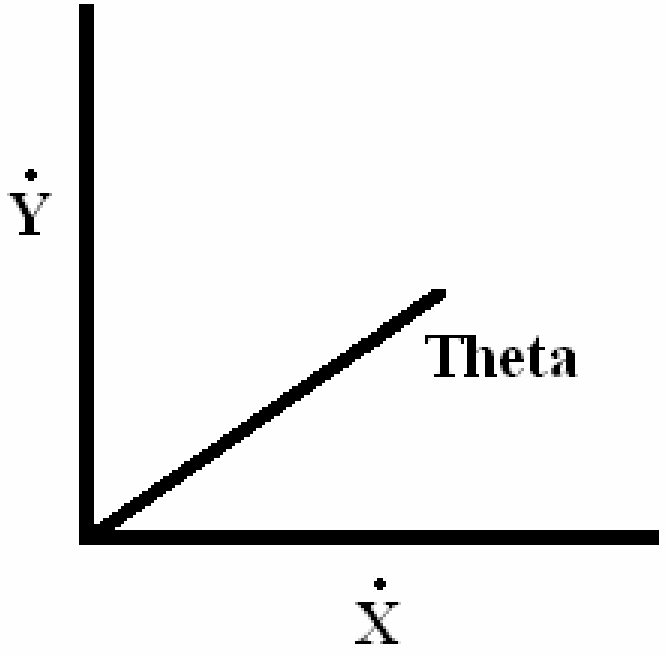
- Our imitator is a two wheeled robot, so we have defined the movement primitives as:
  - Forward
  - Backward
  - Left Turn
  - Right Turn
- A primitive movement for our imitator is obtained by sending one of the above commands for a short duration (~1/4 of a second)

Recognizing Primitives from Vision

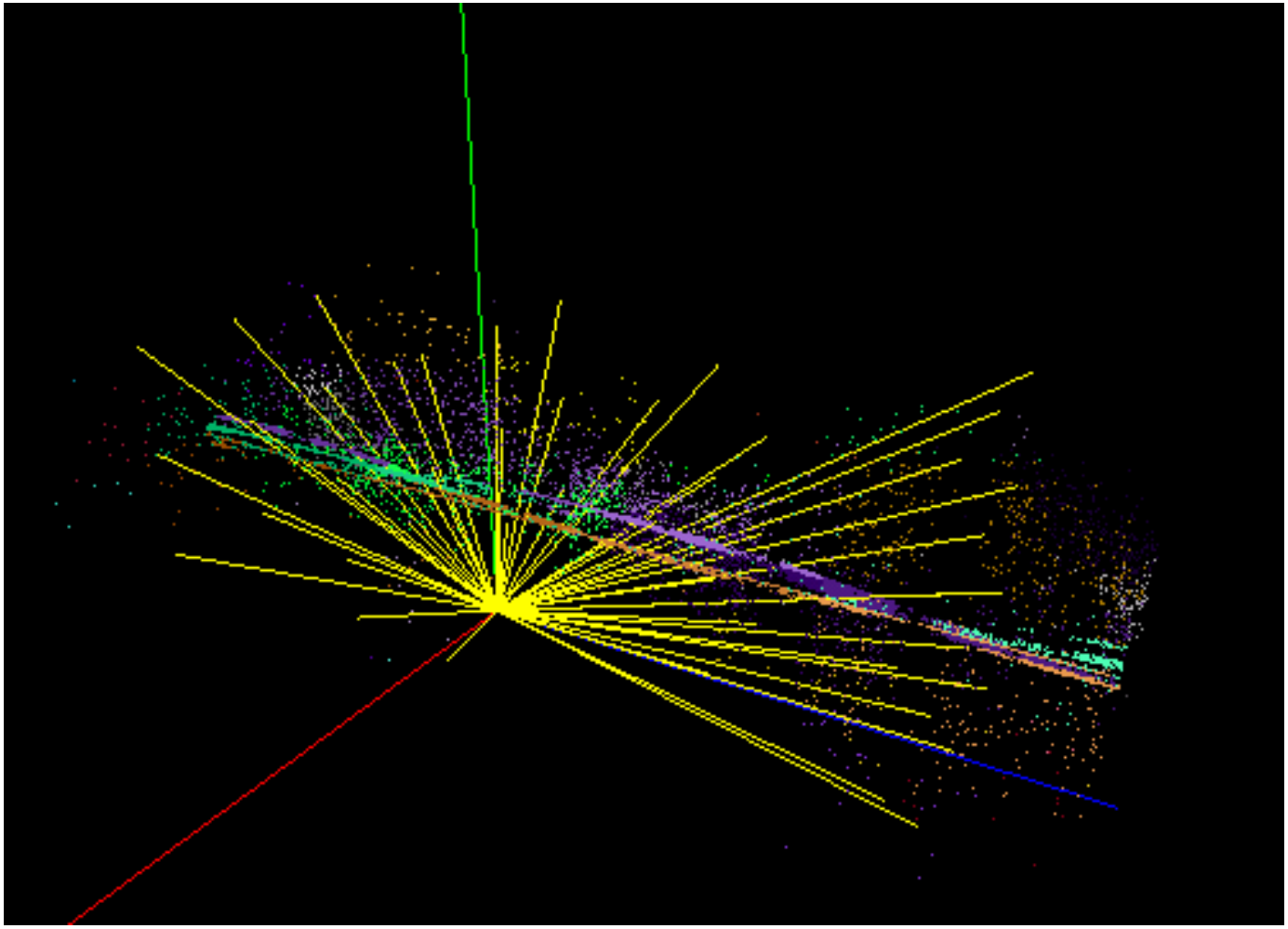
- The imitator first converts a sequence of vision frames into a sequence of its primitives
- In our implementation, a set of Hidden Markov Models is used for the conversion, with a separate HMM for each primitive
- The training data is separated into the HMM states
- The training data is obtained from a global vision server, which supplies a packet for each vision frame, containing the x and y positions and orientation of the robot

Recognizing Primitives from Vision

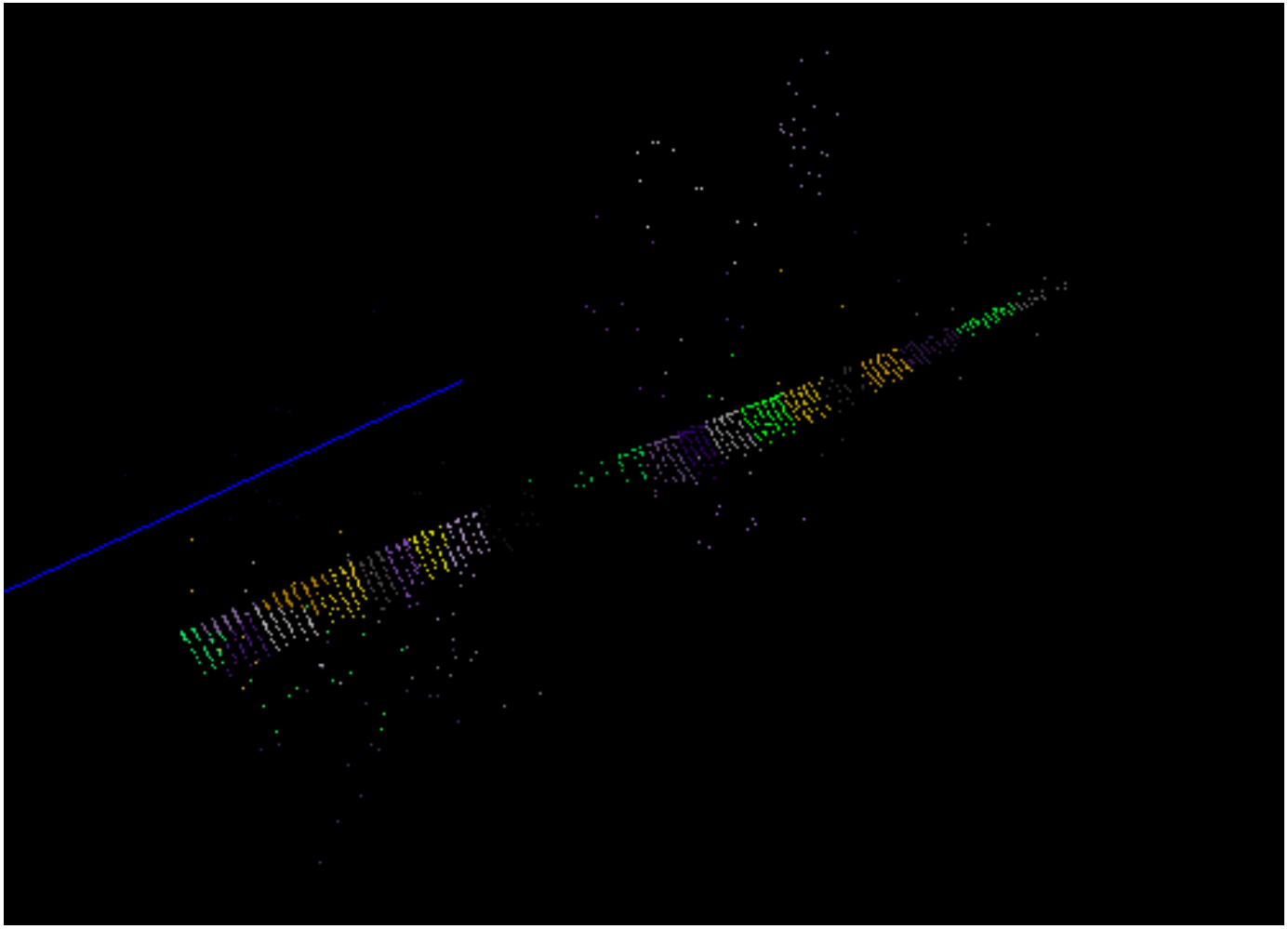
- Vision data is converted into data points
- Each point represents the state change between two frames
- A point contains the change in x, change in y, and resulting theta value



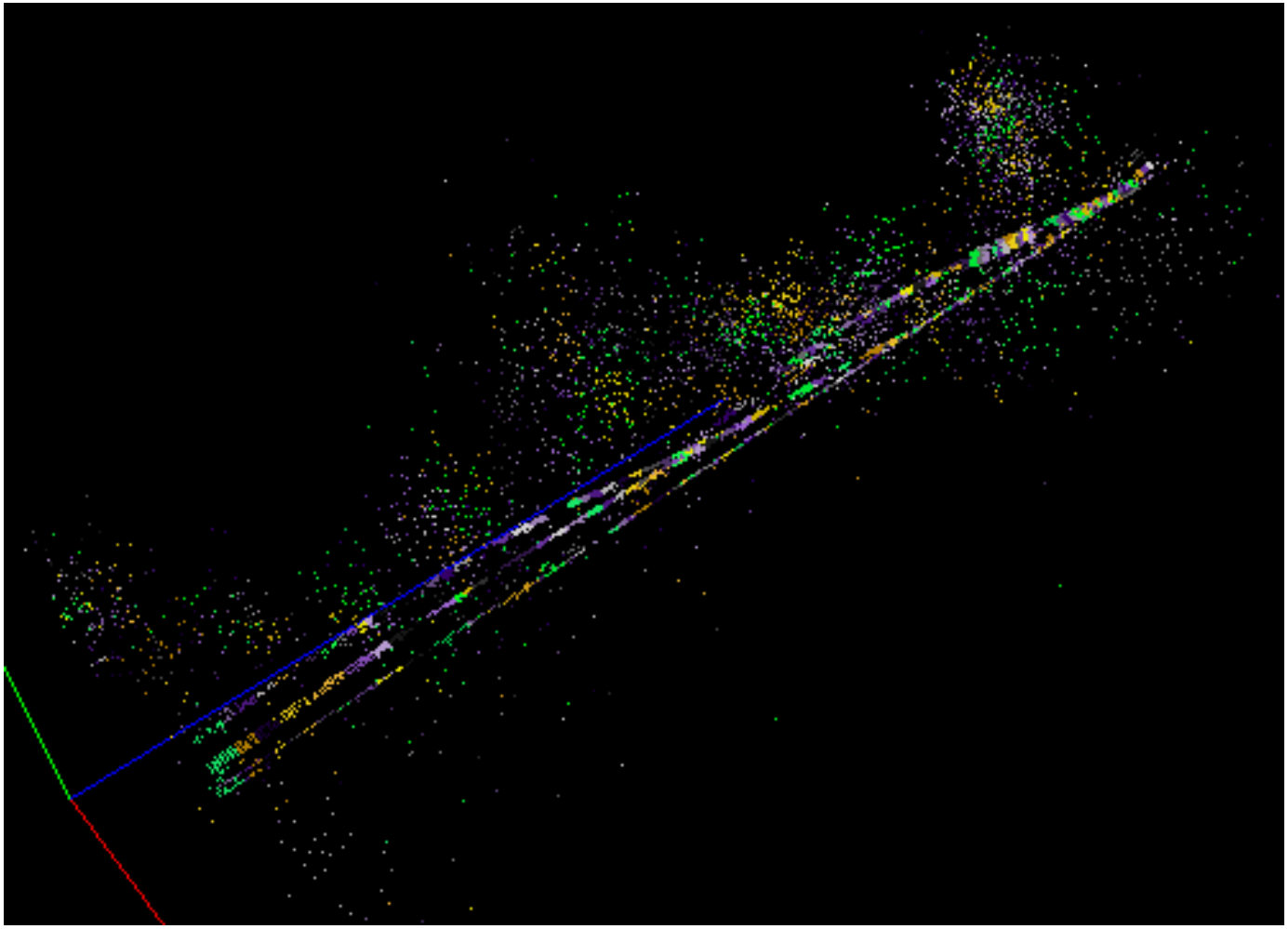
Forward HMM State Centroids



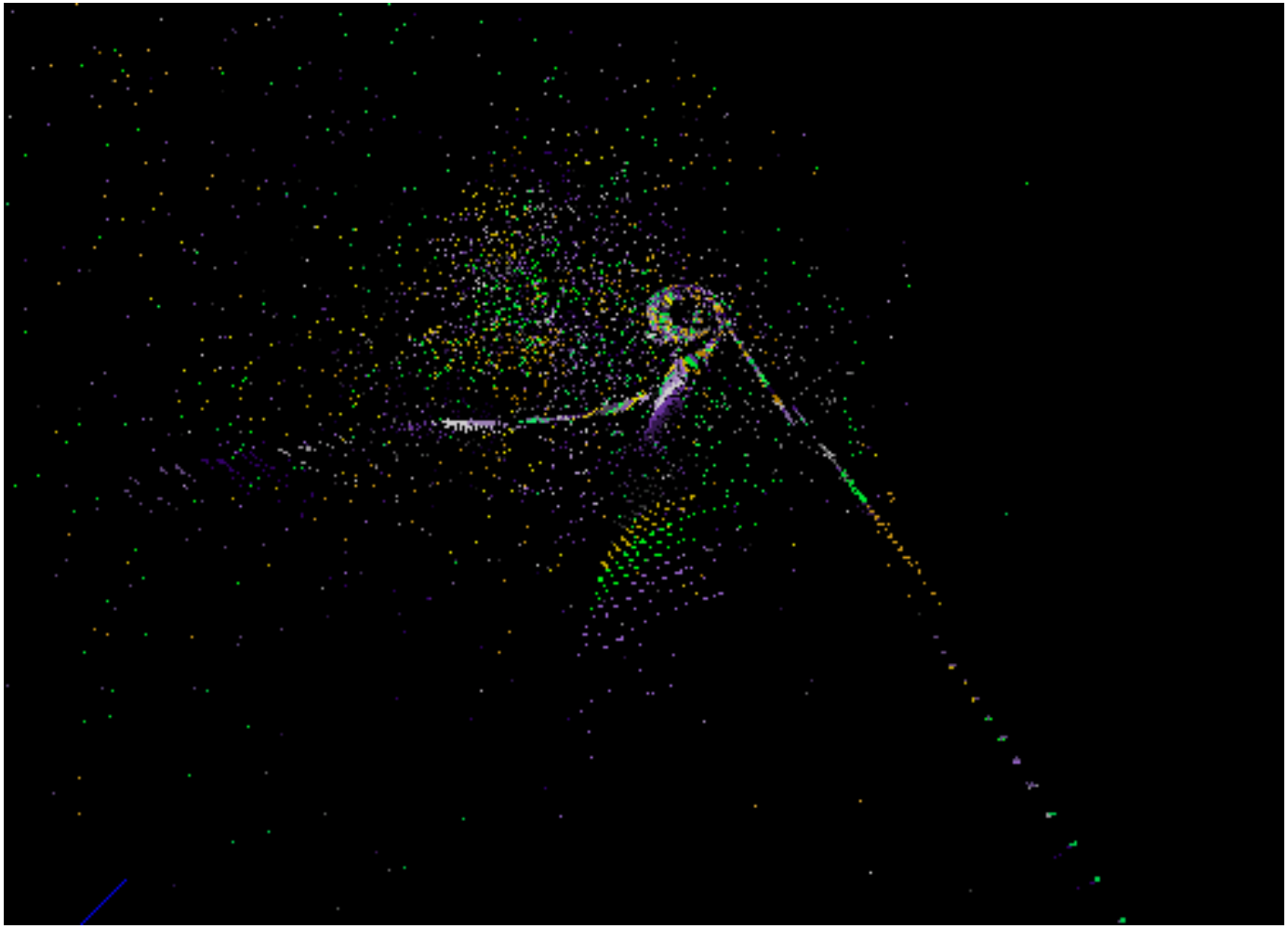
Single HMM's State Clusters



Spiral Pattern



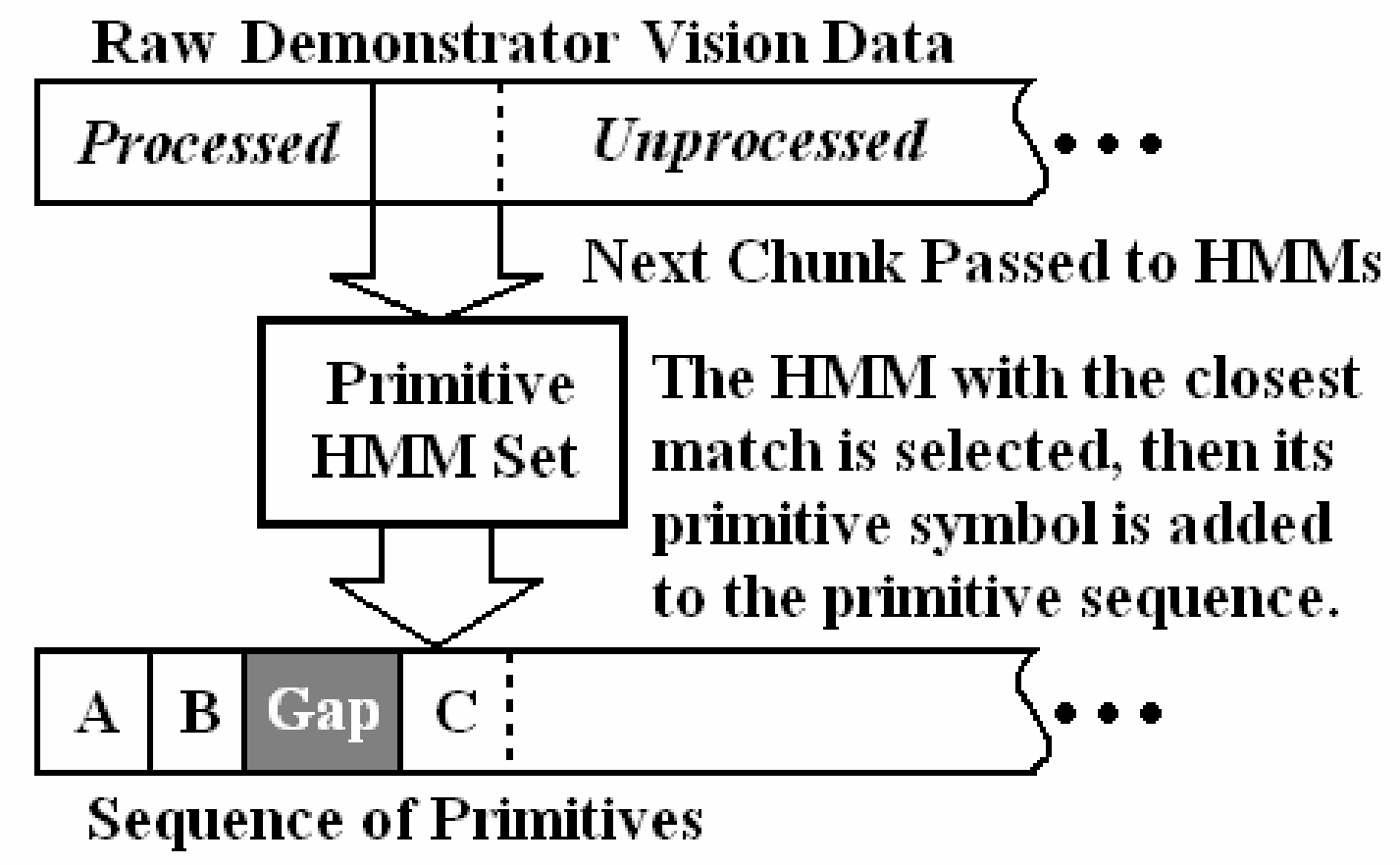
Spiral Pattern



Recognizing Primitives from Vision

- Each state clusters its data using vector quantization to obtain observation symbol probabilities
- The 4 HMMs are then trained separately on their respective data sets (forward, backward, left turn, right turn)
- This sequence of primitives can then be used to construct abstract behaviours to approximate the observed demonstration

Recognizing Primitives from Vision



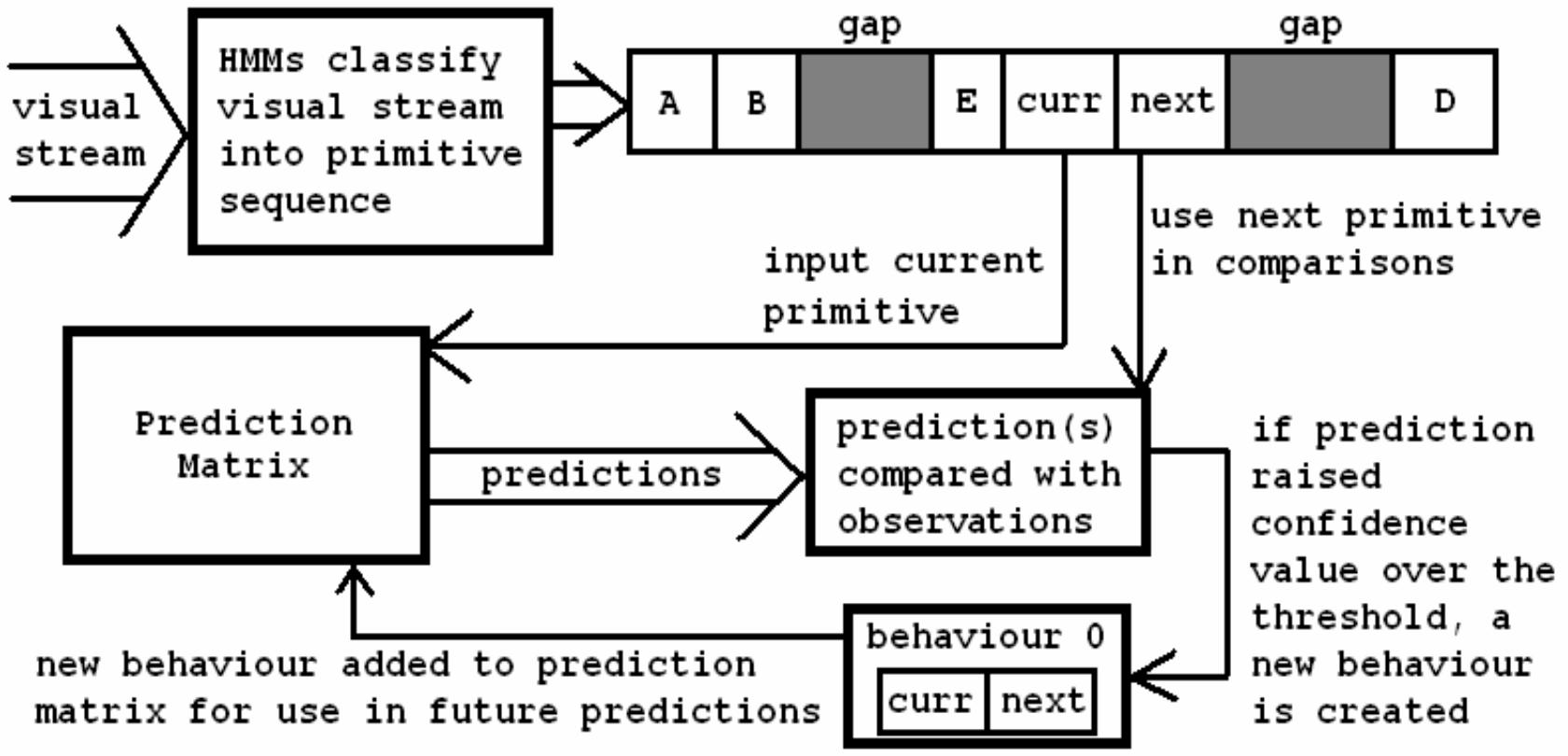
Recognizing Primitives from Vision

- If none of the HMMs can satisfactorily classify a section of the vision data, gaps in classification can occur
- Gaps can result from a demonstrator moving in a way the imitator can’t, due to physiological differences (e.g. wheeled vs humanoid robots)
- These gaps will be bridged later in the learning process, once a behaviour has been learned that can properly approximate the state change of the gap

Learning Architecture

- Our learning architecture employs a type of *forward model*, similar to other work in imitation learning
- Our forward model predicts the next primitive (or behaviour) and applies it to the imitator's internal model of the environmental state
- The accuracy of the predicted state change is compared to the actual state change next time step
- If two primitives are predicted accurately with enough frequency, a new behaviour is created composed of those two primitives in sequence
- The imitator's forward model uses a matrix to keep track of which primitives (or behaviours) follow each other in sequence

Learning Architecture



Learning Architecture

- Each value in the *prediction matrix* corresponds to the *confidence* that a behaviour (or primitive) will follow another
- Predictions are made by using the current behaviour's row, and using that row's highest confidence value as the next predicted behaviour

Learning from Multiple Demonstrators

- Using a single forward model would allow us to learn many behaviours from different demonstrators
- The problem is that different demonstrators may have drastically different behaviours and skill levels
- ...and different physiologies that make the same tasks differ visually in demonstration

Learning from Multiple Demonstrators

- If we use a single forward model, poor behaviours will be learned along with the good ones
- We use a separate forward model for each individual demonstrator
- Each demonstrator is assigned a *learning preference* which is the learning rate specific to that demonstrator
- The imitator will learn behaviours more quickly from good demonstrators, and more slowly from poor ones
- This will speed up the process of acquiring useful behaviours, and guard against learning undesirable behaviours

Evaluating Demonstrators

- Each demonstrator also has a *decay rate* which is 1 – LP (their learning preference)
- At each prediction step, the decay rate is applied to the *permanency* attribute of all the behaviours in the demonstrator's forward model
- If the permanency falls below a threshold, the behaviour is removed
- If it surpasses an upper threshold, it is made permanent, and the decay rate is no longer applied

Evaluating Demonstrators

- When a behaviour is correctly predicted to match the state change, its permanency is increased
- A good behaviour's permanency should be increased frequently enough to overcome its decay rate
- Poor demonstrators' behaviours will decay faster, so only the best behaviours will be learned from them

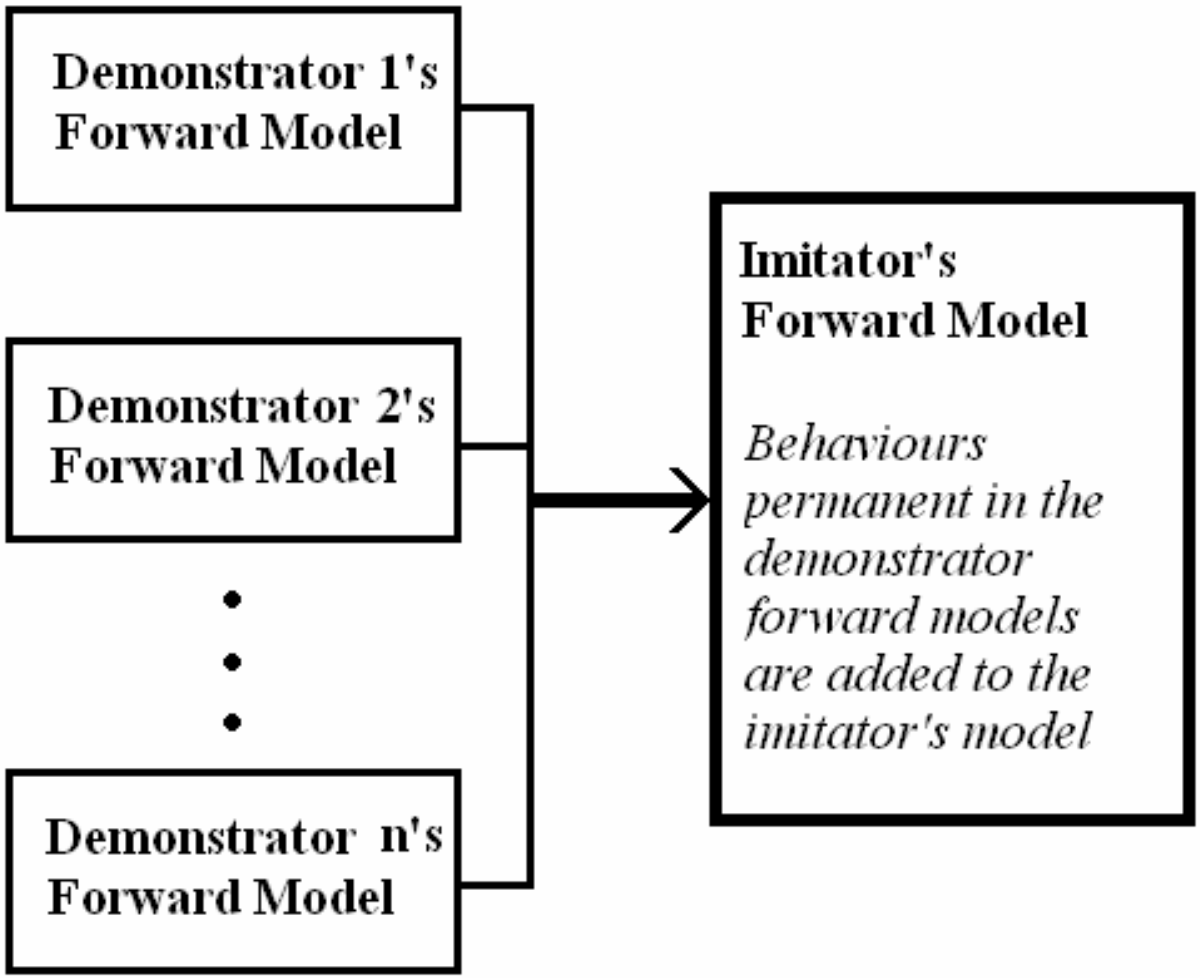
Imitator’s Forward Model

- The imitator also maintains a special ‘main’ forward model that is populated from the demonstrator forward models
- If a behaviour is desirable, it will be added to the imitator’s forward model
- Behaviours from the demonstrators’ forward models are added to the imitator’s forward model if they become permanent

Imitator’s Forward Model

- Each behaviour is added temporarily, and upon further evaluation is either made permanent, or discarded
- The behaviours then have their permanency reset and are evaluated within the imitator’s forward model
- If they achieve permanency in the imitator’s model, they are permanent to the imitator, and are finally a learned behaviour

Imitator’s Forward Model



Summary

- Imitation learning is a powerful tool to acquire behaviours through observation
- Imitation learning can be applied to multiple demonstrators, by separately modelling each demonstrator
- This allows for filtering of behaviours based on the quality of the demonstrator
- Additionally, in a team environment, an imitator could learn from all of its teammates simultaneously
- With little modification, this system could also model opponents, learning their behaviours in order to thwart them