# ACTIVITY MODELING USING EVENT PROBABILITY SEQUENCES

**A MINOR PROJECT REPORT**

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

**BACHELOR OF TECHNOLOGY**

**(Information Technology)**

**SUBMITTED TO**

**DELHI TECHNOLOGICAL UNIVERSITY, DELHI**

**SUBMITTED BY:**

ABHISHEK RAI 2K12/EC/010

**SUPERVISED BY :**

**Dr. S.Indu**

May 2015



**DELHI TECHNOLOGICAL UNIVERSITY, DELHI**

# CERTIFICATE

This is to certify that the thesis entitled “**ACTIVITY MODELING USING EVENT PROBABILITY SEQUENCES**” submitted byAbhinavgupta, Abhishek rai,Chetan in partial fulfilment of the requirements for the award of Bachelor of Technology Degree in Electronicand Communicationfrom Delhi Technological University (Formerly Delhi College of Engineering) for their Minor Project in the Sixth Semester is an authentic work carried out by them under my supervision & guidance.

**Dr. S.Indu**

**Department of Electronics and Communication Engineering**

**Delhi Technological University**

# DELHI TECHNOLOGICAL UNIVERSITY



**DECLARATION**

We hereby declare that the work which is being presented in the B.Tech Minor Project 2 Report entitled “ ACTIVITY MODELING USING EVENT PROBABILITY SEQUENCES”, in partial fulfillment of the requirements for the award of the Bachelor of Technology in Information Technology and submitted to the Department of Electronics and Communication of Delhi Technological University, Delhi is an authentic record of our own work carried out during a period of 6th semester under the supervision of **Professor Dr. S.Indu Electronics and Communication Department.**

The matter presented in this Project Report has not been submitted by us for the award of any other degree elsewhere.

This is to certify that the above statement made by the student(s) is correct to the best of my knowledge.

**Dr. S.Indu**

**Department of Electronics and Communication Engineering**

**Delhi Technological University**

**Table of Contents**

1. ABSTRACT

2. INTRODUCTION

3. PRIOR WORK

3.1 Activities, Events and Actions

3.2 State Space Models For Activity Modeling

3.3 Event Probability Sequence

4. Approach

5. HIDDEN MARKOV MODEL

4.1 Introduction

4.1.1 Terminology

4.2 State diagrams

4.2.1 How to Determine Probability

4.2.2 How to Determine The Parameters

5. EXPERIMENTAL RESULTS

6. Event Modeling Based On Syntatic Representation

7. CONCLUSIONS

8. REFERENCES

Abstract

Changes in motion properties of trajectories provide useful cues for modeling and recognizing human activities. We associate an event with signiﬁcant changes that are localized in time and space, and represent activities as a sequence of such events. Thelocalizednatureofeventsallowsfordetectionofsubtlechanges or anomalies in activities. In this paper, we present a probabilistic approach for representing events using the hidden Markov model (HMM) framework. Using trained HMMs for activities, an event probability sequence is computed for every motion trajectory in the training set. It reﬂects the probability of an event occurring gate every time instant. Though the parameters of the trained HMMs depend on viewing direction, the event probability sequences are robust to changes in viewing direction. We describe sufﬁcient conditions for the existence of view invariance. The usefulness of the pro- posed event representation is illustrated using activity recognition and anomaly detection. Experiments using the indoor University of Central Florida human action dataset, the Carnegie Mellon University Credo Intelligence, Inc., Motion Capture dataset, and the outdoor Transportation Security Administration airport tarmac surveillance dataset show encouraging results.

Index Terms—Activity modeling, event detection, hidden Markov model (HMM).

INTRODUCTION

Activity modeling is basically to observe an activity using any model. By model we observe the sequences of objects used in the activity and determine the result. The model which we will use in this is HMM (Hidden Markov Model). Using HMM, an event probability sequence is computed for every trajectory of the activity.

Human activities can be decomposed into a sequence of events that have a natural physical interpretation. This can be accomplished using semantic approaches in which events are prespeciﬁed; or using statistical approaches , in which modeling is viewed as a problem of inferring (hidden) events from observed data. We present a statistical approach for modeling activities as a sequence of events. At the outset, we brieﬂy discuss the terms activities, actions, primitives and events. Existing approaches distinguish between actions and activities depending on the scale of representation [2], [12]; i.e., individual parts of the body are said to perform actions such as picking up and putting down objects, whereas human interaction with the environment constitutes activities.

Events can be deﬁned based on dominant and persistent characteristics of the data. For example, events can be associated with key frames or exemplars . On the other hand, they canbedeﬁnedusingsigniﬁcantchangesinvelocity,curvatureof motion trajectories and other motion properties. Change basedeventsarenaturallysuitedtoanomalouseventdetection. Also, as discussed in Section III, change-based events can characterize several commonly occurring activities.

We propose an event detection technique using the hidden Markov model (HMM) framework that focusses on stable state transitions under the hypothesis that certain state level transitions denote events. Transitions at the state level are robust to changes that are triggered by noisy measurements. Robustness to noise is enhanced by stable state changes. By stable change, we mean that the probability of event occurrence depends on the value of state variable for several frames before and after the event.

During event detection, several state sequences of the HMM are explored as follows. Consider an observed data sequence (e.g., motion trajectory) of length . In an ergodic HMM with states, there are possible state sequences, each of which can generate the observed data with some probability. The optimal state sequence which is oneamong these state sequences, maximizes thislikelihood; but need not have a semantic interpretation.

It is desirable to ﬁnd methods that are invariant to viewing conditions. The HMM, however, is view-dependent since 2-D motion trajectories are used in training. Multiple HMMs are required if the appearance of trajectories changes signiﬁcantly. We describe the conditions on the HMMs that enable detection of similar events irrespective of viewing direction. Event probability sequences, unlike HMMs, are shown to be quasi-view invariant .

The utility of event probability sequences for activity recognition and anomaly detection is demonstrated using both indoor and outdoor scenarios. It is common to ﬁnd several samples of normal activities, but very few corresponding to anomalous events. It is not practical to model all possible anomalous activities, some of which can arise due to subtle, statistically in- signiﬁcantdeviationsfromnormalcases.Eventsintheproposed method can be used to detect such anomalies since they are a result of local changes.

PRIOR WORK

# ACTIVITY, EVENTS AND ACTIONS

**Activity** is something that you do or something that is going on.

An example of an activity

1. regular customer-bank employee interaction;
2. outsider enters the safe;
3. a bank robbery attempt—the suspected assailant does;
4. not make a getaway;
5. A successful bank robbery;
6. An employee accessing the safe on behalf of a customer.

Basically, activity is a sequence of events that are performed by the objects involved in that activity.

Like in the above example

Activity is – a successful bank robbery.

**Events are** – customer bank employee interaction and like this all the steps mentioned above.

Events were considered as long-term temporal objects at multipletemporal scales andthe chi-squared distance between empirical distributions was used to compare event sequences .

Objects are- bank employee, customer, robber, safe etc.

**Actions -** Actions were segmented using changes in velocity curves in . Sharp changes in curvature of trajectories were used in , instead of velocity curves. These changes are quasi view- invariant, but sensitive to noise, in part because of second-order derivatives used in their computation. Also, many activities do not contain trajectories with changes in curvature. View invariants for human actions in both 2-D and 3-D were developed. In 3-D, actions were represented as curves in an invariance space and the cross ratio was used to ﬁnd invariants.

# STATE SPACE MODELS FOR ACTIVITY MODELING

HMMs have been applied in many vision and speech applications. We provide a representative review of HMM approaches in activity modeling. One of the earliest applications of HMMs in vision was for recognizing tennis strokes . Since then, HMMs have been used for recognizing sign language, gait-based identiﬁcation of humans , and activities

Trajectories were modeled in two phases: primitives were represented by HMMs, and temporal sequencing between primitives was enforced using SCFG. Presegmentation for training and manual design of **SCFG** might limit scalability of the approach. In a related work, unsurpervised clustering based on factorization of an afﬁnity matrix of the output of HMMs was used in place of SCFG .

Trajectories were divided into subtrajectories based on changes in curvature. The subtrajectories were resampled to ensure that all segments have an equal number of elements. The top few principal components of the subtrajectories were modeled using **GMMs and HMMs**.

**Coupled HMMs** were used to model actions involving body parts such as hands and head in which states of HMMs rep- resenting the motion in different parts of the body are forced to make transitions simultaneously. Generally, deciding which HMM states have to be coupled may not be obvious.

**Variable length HMMs**allow for dependencies with varying number of time steps. Increasing the order is tantamount to augmenting the state space, which, in turn, increases computational complexity [33] and the amount of training data required for reliable parameter estimation.

**In layered HMMs**, activities are modeled at multiple levels of temporal granularity using several HMMs in parallel . Generally, the states of an HMM need not represent physically meaningful entities when the model is constructed using the maximum likelihood criterion. Instead, entropy minimization was used to estimate the HMM parameters. The resulting states were structured and interpretable.

# EVENT PROBABILITY SEQUENCE

We propose event probability sequences to quantify the notion of important characteristics of activities. Every motion trajectory is associated with an event probability sequence that is computed in two phases: an HMM is learned using the given motion trajectories, and event probability sequences are computed using the learned HMM and given motion trajectories. The HMM enables easy generalization, i.e., the structure of the model need not be manually speciﬁed for different activities. Usingthe HMM, weexplore a subsetof state sequencestodetect events. The hypothesis is that signiﬁcant changes in the video sequence are reﬂected as events, and a sequence of events forms an activity.

The key idea is that stable transitions at the state level reﬂect signiﬁcant changes in motion properties that are denoted as events. State-level transitions provide a robust representation of change compared to those deﬁned at the data-level. Moreover, the number of distinct changes at the state level at any given time is ﬁnite (and equal to ), and its probability of occurrence can be computed efﬁciently.

An event is speciﬁed by the following quantities:

• probability of the event ;

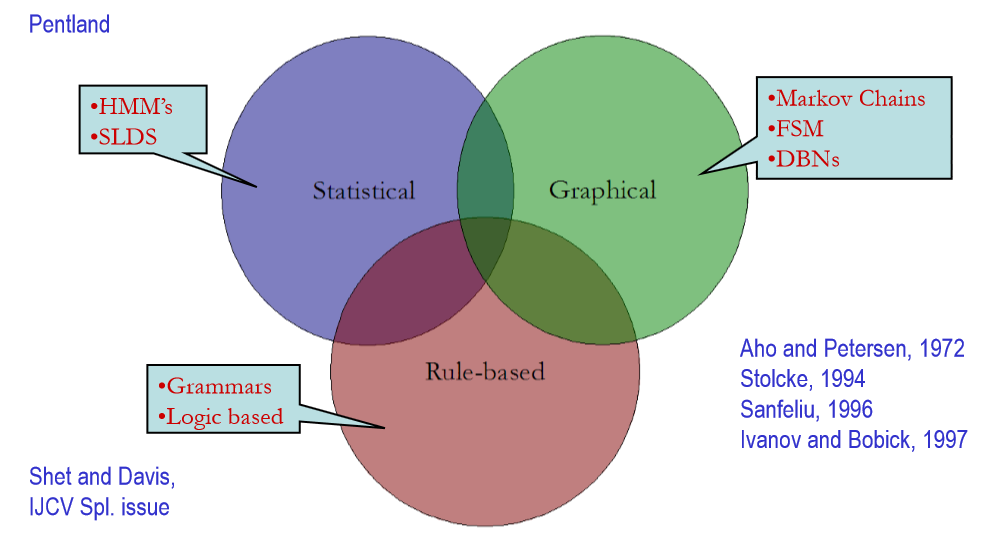
• scale parameter p denoting the length of each frame.

• event type(k, l) denoting the states before and after the event.

APPROACH

There are many approaches for this.

Few of the are given below :



**Statistical approach**

In which modeling is viewed as a problem of inferring (hidden) events from observed data.

**Semantics approach**

In which events are prespeciﬁed

**Graphical approach**

In which events and activity is represented graphically using markov chains, DBNs.

But we will use statistical approach only that is via HMM.

Now how dose HMM works and detail of HMM is given in next section.

HIDDEN MARKOV MODEL

# Introduction

HMMs can be used to evaluate a query sequence: to determine how likely the query is to fit the model.

It is used to summarize multiple sequence alignments, and score new sequences.

* The Markov Property: the state of a system depends only on its previous state.
  + Works very well for speech recognition: a sound is modified by the sound immediately preceding it.
  + In a first order Markov model, the current event only depends on the immediately preceding event. Second order models use the two preceding events, third order models use the three preceding, etc.

Example :all the HMMs for protein sequences that are readily available are first order models

HMM

* Example 1 :The basic problem is: given a series of observations, determine the most probable set of hidden states that generate them.
  + For matching a query sequence to a model, each residue in the query is assigned one of 3 states: it is a match/mismatch to a residue in the model, or it is deleted, or it is inserted

Example 2: I have a friend (Carol) in St. Paul who has only 3 interests in life: going for a walk, shopping, and cleaning her apartment.

* + St. Paul weather can be either sunny or rainy. Carol decides which of her 3 activities she will do based on the weather.
  + Every day she e-mails me to tell me what she is doing. (exciting!)
  + From the sequence of her activities, I infer the most likely weather for each day.

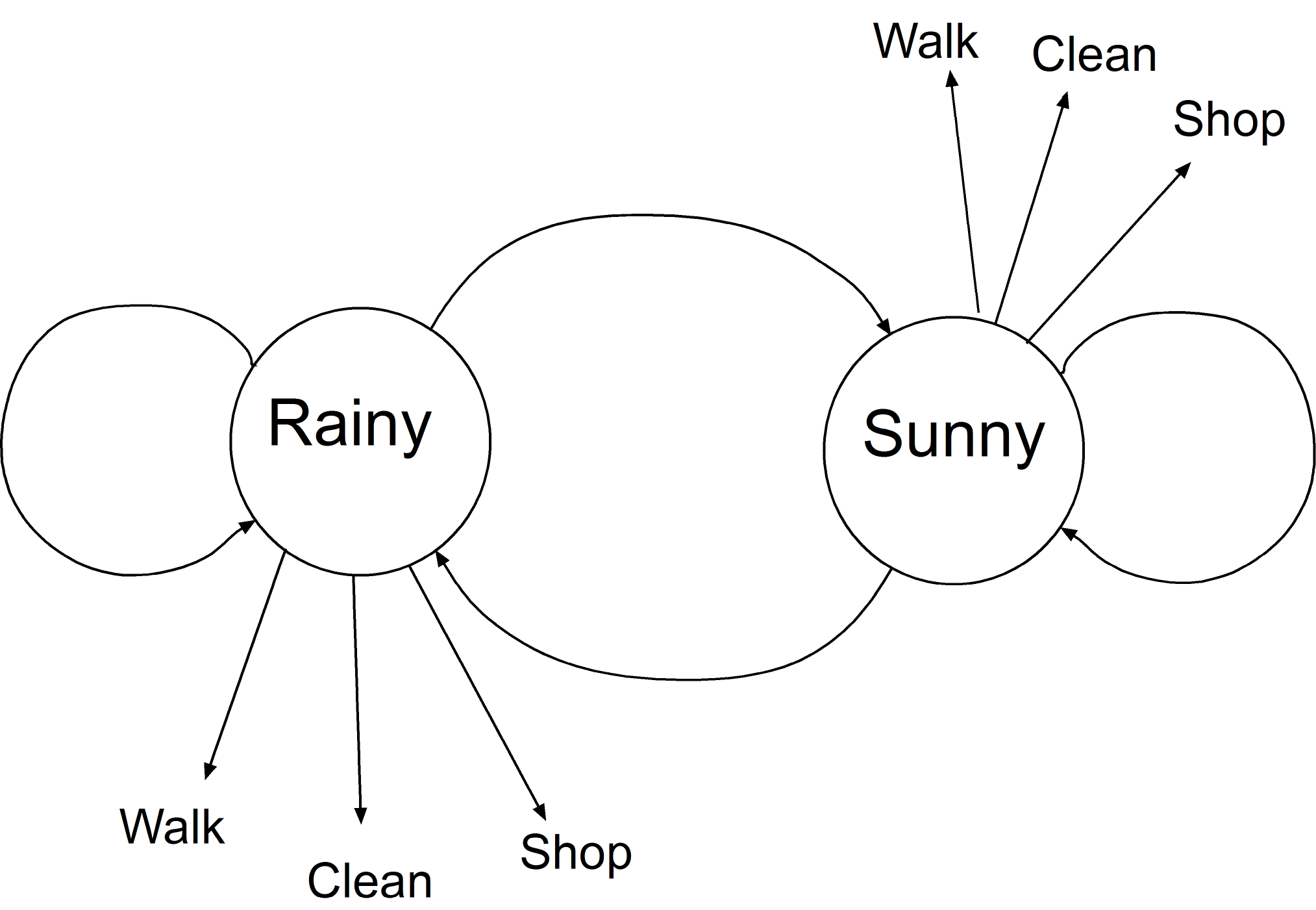
# TERMINOLOGY

* The HMM for Life in St. Paul has 2 states: rainy and sunny.
  + These states are hidden: we don’t know them and can only infer them from Carol’s behavior.
* For each set, there is a set of transition probabilities: what will happen the next day.
  + Which depends only on what happened today (the Markov property)
  + prob R -> R (stays rainy) = 0.7
  + prob R -> S (goes from rain to sun) = 0.3
  + prob S-> S (stays sunny) = 0.6
  + prob S -> R (goes from sun to rain) = 0.4
* For each state, there is a set of emission probabilities: what Carol’s behavior will be under different weather conditions.
  + These behaviors are our observations: she tells us what she is doing each day.
  + if state is Rainy:
    - walk = 0.1
    - clean = 0.5
    - shop = 0.4
  + if state is Sunny:
    - walk = 0.6
    - clean = 0.1
    - shop = 0.3

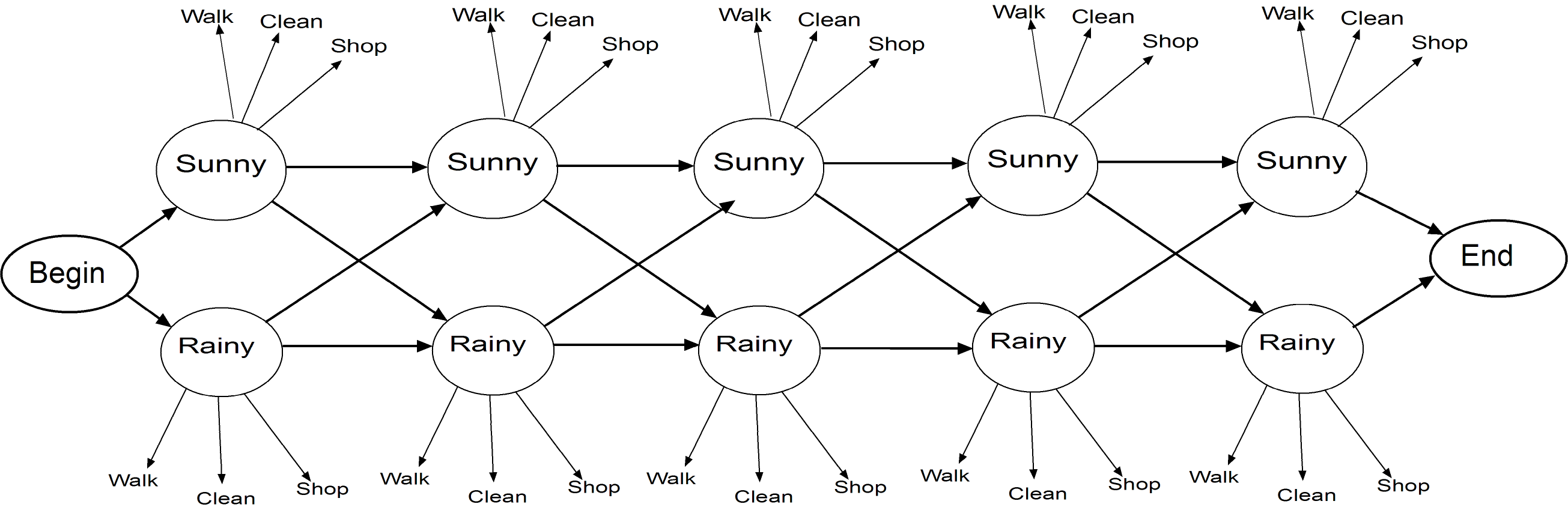
A path is a sequence of states through the model: what the state is for each day. We are trying to determine the optimal (most likely) path that fits our observations

# STATE DIAGRAMS

We can represent the above example via state diagram.



* One way to draw state diagram is to have arrows going back and forth between states. Note that there is also a transition back to the same state.
  + This represents eternity: each day is in one of the other state.
* Another way is to move forward in time, with a new pair of Rainy and Sunny states for each new day. I like this because it is a closer approximation to progressing through a sequence, trying to get a match between the query sequence and the model.



## How to determine probability

* + What is the overall probability that the model fits our observations?
    - This is the most common use of HMMs in bioinformatics: determining the probability that the query sequence fits the model
* The parameters that need to be estimated are the transition and emission probabilities. Since they must sum to 1, our St. Paul weather HMM needs only 2 transition probabilities and 2 emission probabilities estimated.
  + Sequence models have very large numbers of parameters to estimate, which requires large amounts of data.
* If you have training data with known states (e.g. you know when it is sunny or rainy, plus your friend’s activities, for a period of time: parameters come from simply calculating the proportion of each transition or emission.
  + Example: on rainy days, Carol took 6 walks, cleaned 30 times, and shopped 24 times. Emission probability estimates are 6/60 = 0.1, 30/60 =0.5, 24/60 = 0.4. (very frequentist!)
  + problem of 0 counts. Add a “pseudocount” so this probabilty isn’t zero (because it might actually happen). Typically add 1 to numerator and total number of possibilites to denominator.
    - e.g. for walks, emission prob = (6 +1) / (60 + 3) = 7/63 = 0.111

## How to determine the parameters

* Start with a good multiple alignment, which generally involves hand refinement. This is often refered to as “curated”.
* Most of the columns will become match states.
  + Pseudocounts are needed to make sure that each match state has a non-zero probability of emitting all 20 amino acids.
* Insertion and deletion states are given some background low transition probabilities to start with.
  + columns that are mostly gaps are considered insertion states, and gaps in columns with mostly amino acids are modeled with deletion states.
  + The query sequence is used as a set of observations that are emitted from the HMM if the proper path is followed

EXPERIMENTAL RESULTS

We implemented object detection apart from HMM.

With the help of object detection only HMM get to know about different events

 Object detection and segmentation is the most important and challenging fundamental task of computer vision.  It is a critical part in many applications such as image search, scene understanding, etc. However it is still an open problem due to the variety and complexity of object classes and backgrounds.  
  
The easiest way to detect and segment an object from an image is the color based methods . The object and the background should have a significant color difference in order to successfully  segment objects using color based methods.

# *Simple Example of Detecting a Red Object*

**CODE**

**#include <iostream>**  
**#include "opencv2/highgui/highgui.hpp"**  
**#include "opencv2/imgproc/imgproc.hpp"**  
 **using namespace cv;**  
**using namespace std;**  
 **int main( int argc, char\*\* argv )**  
**{**  
**VideoCapture cap(0); //capture the video from web cam**  
 **if ( !cap.isOpened() )  // if not success, exit program**  
**{**  
**cout<< "Cannot open the web cam" <<endl;**  
**return -1;**  
**}**  
 **namedWindow("Control", CV\_WINDOW\_AUTOSIZE); //create a window called "Control"**  
 **int iLowH = 0;**  
**int iHighH = 179;**  
 **int iLowS = 0;**  
**int iHighS = 255;**  
 **int iLowV = 0;**  
**int iHighV = 255;**  
 **//Create trackbars in "Control" window**  
**cvCreateTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)**  
**cvCreateTrackbar("HighH", "Control", &iHighH, 179);**  
 **cvCreateTrackbar("LowS", "Control", &iLowS, 255); //Saturation (0 - 255)**  
**cvCreateTrackbar("HighS", "Control", &iHighS, 255);**  
 **cvCreateTrackbar("LowV", "Control", &iLowV, 255); //Value (0 - 255)**  
**cvCreateTrackbar("HighV", "Control", &iHighV, 255);**  
 **while (true)**  
**{**  
**Mat imgOriginal;**  
 **bool bSuccess = cap.read(imgOriginal); // read a new frame from video**  
 **if (!bSuccess) //if not success, break loop**  
**{**  
**cout<< "Cannot read a frame from video stream" <<endl;**  
**break;**  
**}**  
 **Mat imgHSV;**  
 **cvtColor(imgOriginal, imgHSV, COLOR\_BGR2HSV); //Convert the captured frame from BGR to HSV**  
  
**Mat imgThresholded;**  
 **inRange(imgHSV, Scalar(iLowH, iLowS, iLowV), Scalar(iHighH, iHighS, iHighV), imgThresholded); //Threshold the image**  
  
**//morphological opening (remove small objects from the foreground)**  
**erode(imgThresholded, imgThresholded, getStructuringElement(MORPH\_ELLIPSE, Size(5, 5)) );**  
**dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH\_ELLIPSE, Size(5, 5)) );**  
 **//morphological closing (fill small holes in the foreground)**  
**dilate( imgThresholded, imgThresholded, getStructuringElement(MORPH\_ELLIPSE, Size(5, 5)) );**  
**erode(imgThresholded, imgThresholded, getStructuringElement(MORPH\_ELLIPSE, Size(5, 5)) );**  
 **imshow("Thresholded Image", imgThresholded); //show the thresholded image**  
**imshow("Original", imgOriginal); //show the original image**  
 **if (waitKey(30) == 27) //wait for 'esc' key press for 30ms. If 'esc' key is pressed, break loop**  
**{**  
**cout<< "esc key is pressed by user" <<endl;**  
**break;**  
**}**  
**}**  
 **return 0;**  
  
**}**

# Explanation

OpenCV usually captures images and videos in 8-bit, unsigned integer, BGR format. In other words, captured images can be considered as 3 matrices; BLUE, GREEN and RED (hence the name BGR) with integer values ranges from 0 to 255.  
  
The following image shows how a color image is represented using 3 matrices.

|  |
| --- |
| How BGR image is formed using 3 matrices which represent blue, green and red planes |
| How BGR image is formed |

In the above image, each small box represents a pixel of the image. In real images, these pixels are so small that human eye cannot differentiate.  
  
Usually, one can think that BGR color space is more suitable for color based segmentation. But [HSV color space](http://en.wikipedia.org/wiki/HSL_and_HSV) is the most suitable color space for color based image segmentation. So, in the above application, I have converted the color space of original image of the video from BGR to HSV image.  
  
HSV color space is also consists of 3 matrices, HUE, SATURATION and VALUE. In OpenCV, value range for  HUE, SATURATION  and VALUE  are respectively 0-179, 0-255 and 0-255. HUE represents the color, SATURATION  represents the amount to which that respective color is mixed with white and VALUE  represents the  amount to which that respective color is mixed with black.  
　  
In the above application, I have considered that the red object has HUE, SATURATION and VALUE in between 170-180, 160-255, 60-255 respectively. Here the HUE is unique for that specific color distribution of that object. But SATURATION and VALUE may be vary according to the lighting condition of that environment.  
  
Hue values of basic colors

* + Orange  0-22
  + Yellow 22- 38
  + Green 38-75
  + Blue 75-130
  + Violet 130-160
  + Red 160-179

These are approximate values. You have to find the exact range of HUE values according to the color of the object. I found that the range of 170-179 is perfect for the range of hue values of my object. The SATURATION and VALUE is depend on the lighting condition of the environment as well as the surface of the object.   
  
  
  
After thresholding the image, you'll see small white isolated objects here and there. It may be because of noises in the image or the actual small objects which have the same color as our main object. These unnecessary small white patches can be eliminated by applying **morphological opening**. **Morphological opening** can be achieved by a erosion, followed by the dilation with the same structuring element.  
  
Thresholded image may also have small white holes in the main objects here and there. It may be because of noises in the image. These unnecessary small holes in the main object can be eliminated by applying **morphological closing**. **Morphological closing**can be achieved by a dilation, followed by the erosion with the same structuring element.  
  
Now let's discuss new OpenCV methods in the above application.

* **void inRange(InputArray src, InputArray lowerb, InputArray upperb, OutputArray dst);**

Checks that each element of 'src'  lies between 'lowerb' and 'upperb'. If so, that respective location of  'dst' is assigned '255' , otherwise '0'. (Pixels with value 255 is shown as white whereas pixels with value 0 is shown as black)

Arguments -

* + **InputArray src**- Source image
  + **InputArray lowerb** - Inclusive lower boundary (If **lowerb=**Scalar(x, y, z), pixels which have values lower than x, y and z for HUE, SATURATION and VALUE respectively is considered as black pixels in **dst**image)
  + **InputArray upperb** -  Exclusive upper boundary (If it is **upperb=**Scalar(x, y, z), pixels which have values greater or equal than x, y and z for HUE, SATURATION and VALUE respectively is considered as black pixels in **dst**image)
  + **OutputArray dst** -  Destination image (should have the same size as the **src** image and should be 8-bit unsigned integer, CV\_8U)
* **void erode( InputArray src, OutputArray dst, InputArray kernel, Point anchor=Point(-1,-1), int iterations=1, int borderType=BORDER\_CONSTANT, const Scalar& borderValue=morphologyDefaultBorderValue() )**

This function erodes the source image and stores the result in the destination image. In-place processing is supported. (which means you can use the same variable for the source and destination image). If the source image is multi-channel, all channels are processed independently and the result is stored in the destination image as separate channels.

Arguments -

* + **InputArray src -**Source image
  + **OutputArray dst**- Destination image (should have the same size and type as the source image)
  + **InputArray kernel** - Structuring element which is used to erode the source image
  + **Point anchor** - Position of the anchor within the kernel. If it is Point(-1, -1), the center of the kernel is taken as the position of anchor
  + **int iterations** - Number of times erosion is applied
  + **int borderType** - Pixel extrapolation method in a boundary condition
  + **const Scalar& borderValue** - Value of the pixels in a boundary condition if **borderType = BORDER\_CONSTANT**
* **void dilate( InputArray src, OutputArray dst, InputArray kernel,**
* **Point anchor=Point(-1,-1), int iterations=1,**

* **int borderType=BORDER\_CONSTANT,**

* **const Scalar& borderValue=morphologyDefaultBorderValue() );**

This function dilates the source image and stores the result in the destination image. In-place processing is supported. (Which means you can use the same variable for the source and destination image). If the source image is multi-channel, all channels are processed independently and the result is stored in the destination image as separate channels.

* + **InputArray src** - Source image
  + **OutputArray dst** - Destination image (should have the same size and the type as the source image)
  + **InputArray kernel** - Structuring element which is used to dilate the source image
  + **Point anchor** - Position of the anchor within the kernel. If it is Point(-1, -1), the center of the kernel is taken as the position of anchor
  + **int iterations**- Number of times dilation is applied
  + **int borderType**- Pixel extrapolation method in a boundary condition
  + **const Scalar& borderValue**- Value of the pixels in a boundary condition if **borderType = BORDER\_CONSTANT**
* **void cvtColor( InputArray src, OutputArray dst, int code, int dstCn=0 )**

This function converts a source image from one color space to another. In-place processing is supported. (Which means you can use the same variable for the source and destination image)

* + **InputArray src** - Source image
  + **OutputArray dst** - Destination image (should have the same size and the depth as the source image)
  + **int code** - Color space conversion code (e.g. - COLOR\_BGR2HSV, COLOR\_RGB2HSV, COLOR\_BGR2GRAY, COLOR\_BGR2YCrCb, COLOR\_BGR2BGRA, etc)
  + **int dstCn** - Number of channels in the destination image. If it is 0, number of channels is derived automatically from the source image and the color conversion code.

Event Modeling Based on Syntactic Representations

Overview – Representation using attribute grammar

• Extension of attribute grammar – Recognition by online parsing – Results

• Specific event recognition

• Anomaly detection

• Attribute grammar – Grammar: strings naturally correspond to sequences, compact – Attributes: describe general features, allow events with multiple objects & concurrent events

• Woo and Chellappa, 2007.

# Primitives

• Input symbols  correspond to  “primitive  events” – e.g., stop, disappear

• Attributes – Additional features associated with primitive event – Features that cannot be represented by (finite) input symbols – e.g., location, object id

# Attribute grammar

• Definition AG = (G, SD, AD, R, C) – G = (VN, VT, P, S): Context free grammar. Defines syntax. – SD : Semantic domain. Set of attribute types and functions. – AD : Set of attributes associated with each symbol in P

• Attributes  are  interpreted  as  “semantics” – R : Set of attribute evaluation rules

• Functions that determines attribute values – C : Set of semantic conditions (predicates) on attributes

• For each production

• The production can be used only if the conditions are satisfied

# Example

PARKING o CARPARK perapp disappear ( near(X1.loc,X2.loc)  near(X3,BldgEntrance) ) CARPARK o carapp carstart STOP ( inside(X3.loc, ParkingSpace) ) STOP o carstop carstart STOP X0.loc := f(X1.loc, X3.loc) STOP o carstop X0.loc := X1

# Recognition

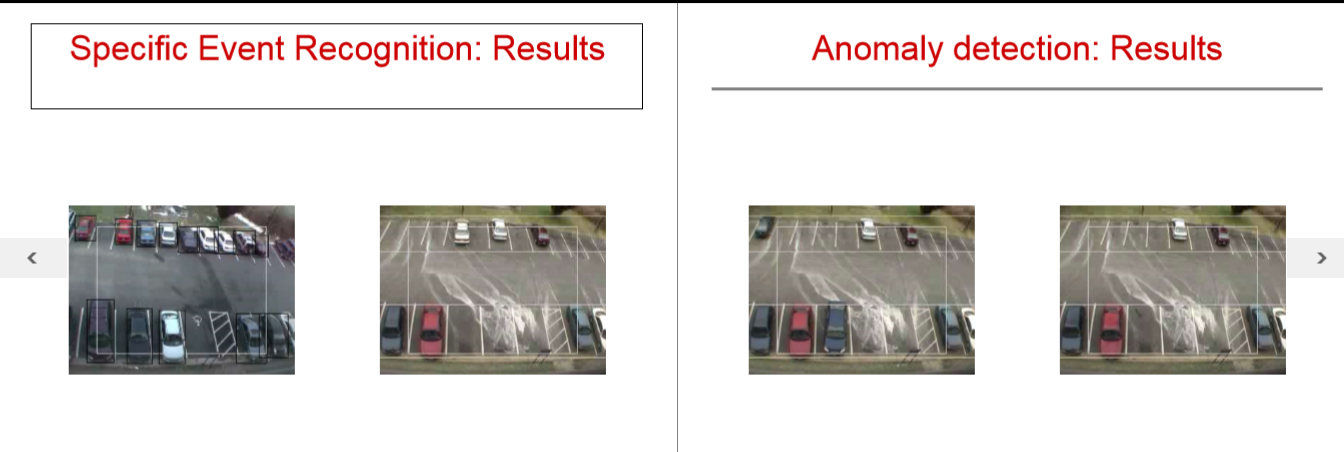
• Parsing  based  on  Earley’s  algorithm  (1970)

• Error correcting grammar for handling tracking error

• Two types of applications – Recognize specific events: Ignore negative patterns – Detect anomalies: Classify into positive & negative patterns

• Event recognition system – Track moving objects, classify into person / vehicle, generate primitive events – Positive recognition

• Successfully parsed (syntactic) with high confidence (in attribute condition) – Anomaly detection

 • Parse failed or parsed with low confidence

Conclusion

In this project, we have shown that how an activity is recognized via several models (HMM) on the observed sequence of events. By using different approaches and making probability sequence of an event of objects taking place and determining their probabilities for happening and for not happening in an event. Based on this observation HMM recognize the activity.

We have also implemented object detection here in this project through which HMM get to knows about the behavior of an Object.

References

* + <http://libra.msra.cn/Publication/4413038/activity-modeling-using-event-probability-sequences>
  + <https://www.shodor.org/interactivate/activities/>
  + <http://www.researchgate.net/publication/5462942_Activity_modeling_using_event_probability_sequences>