HUMAN ACTION RECOGNITION SYSTEM

A PROJECT REPORT P-2175

Submitted by

JAGAN.D (71204205012)
MADHAVARAJ.N (71204205018)

In partial fulfillment for the award of the degree
Of

BACHELOR OF TECHNOLOGY
in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2008

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "HUMAN ACTION RECOGNITION SYSTEM" is the bonafide work of Jagan.D and Madhavaraj.N who carried out the project work under my supervision.

SIGNATURE
Dr. S. Thangasamy
Dean
Computer Science & Engineering
Kumaraguru college of Tech.
Chinnavedampatti post
Coimbatore-641006

SIGNATURE
Ms. N. Suganthi
Senior Lecturer
Information Technology
Kumaraguru college of Tech.
Chinnavedampatti post
Coimbatore-641006

The candidates with University Register Nos. **71204205012, 71204205018** examined by us in the project viva-voce examination held on .24.04.08.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our sincere thanks to our chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc, F.I.E** and correspondent **Prof. K. Arumugam B.E., M.S., M.I.E.,** for all their support and ray of strengthening hope extended. We are immensely grateful to our principal **Dr. Joseph V. Thanikal M.E., Ph.D., PDF., CEPIT.,** for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr.S.Thangasamy**, Head of the Department of Computer Science and Engineering for his valuable guidance and useful suggestions during course of this project.

We also extend our heartfelt thanks to our project coordinator **Prof.K.R.Baskaran B.E., M.S.**, Department of Information Technology for providing us his support which really helped us.

We are indebted to our project guide **Ms. N. Suganthi M.E.,** Department of Information Technology for her helpful guidance and valuable support given to us throughout this project.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support during the course of this project. We also thank all of our friends who helped us to complete this project successfully.

DECLARATION

We,

JAGAN .D	**71204205012**
MADHAVARAJ .N	**71204205018**

hereby declare that the project entitled **"HUMAN ACTION RECOGNITION SYSTEM"**, submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru college of Technology, Coimbatore.

Place: Coimbatore

Date: 22.04.08

(Jagan .D)

(Madhavaraj .N)

TABLE OF CONTENTS

ABSTRACT

We learn explicit representations for dynamic shape manifolds of moving humans for the task of action recognition. We exploit locality preserving projections (LPP) for dimensionality reduction, leading to a low-dimensional embedding of human movements.

Given a sequence of moving silhouettes associated to an action video, by LPP, we project them into a low-dimensional space (LDS) to characterize the spatiotemporal property of the action, as well as to preserve much of the geometric structure.

Action classification is then achieved in a nearest neighbor framework. The experimental results show that the proposed method is able to not only recognize human actions effectively, but also considerably tolerate some challenging conditions, eg: low-quality videos, changes in viewpoints, scales, and clothes within-class variations caused by different subjects with different physical build; styles of motion.

LIST OF FIGURES

LIST OF ABBREVIATIONS

LPP	-	Locality Preserving Projections
CED	-	Canny Edge Detection
LDS	-	Low Dimensional Space
HMM	-	Hidden Markove Model

INTRODUCTION

GENERAL

PROBLEM DEFINITION

OBJECTIVE OF THE PROJECT

CHAPTER 1

INTRODUCTION

1.1 GENERAL:

Visual analysis of human movements concerns the detection, tracking and recognition of people, and, more generally, the understanding of human activities, from image sequences. In particular, the recognition of human activities has a wide range of promising applications such as smart surveillance, perceptual interfaces, interpretation of sport events, etc. Although there has been much work on human motion analysis over the past two decades, activity understanding still remains challenging.

In terms of higher-level analysis, previous studies generally fall under two major categories of approaches, i.e., template matching based approaches and state-space approaches. The former usually characterizes the spatiotemporal distribution generated by the motion in its continuum, e.g., Bobick and Davis proposed temporal templates for the representation and recognition of aerobics actions.

The latter generally defines each static posture in the action as a state, and recognizes it through considering temporal variations of those poses using state-space models, e.g., Yamato *et al.* combined the mesh features of 2-D human blobs with the HMMs to identify tennis behaviors. Our approach is a form of matching and is, thus, closer to the first category.

1.2 PROBLEM DEFINITION:

Given a video as input, we identify the action of it. The input video is converted in to images and then extracted in to frames. Edges can be detected from the frames. For detection, we convert the image in to silhouettes which is the dark outline of the image which is then projected in to a low dimensional space to characterize the spatio temporal property of the action, as well as to preserve much of the geometric structure.

To match the embedded action trajectories, the median hausdorff distance or normalized spatiotemporal correlation is used for similarity measures. We have to store the edges into the Files. The stored file is compared with the input video.

A message "action found" will be displayed if any match has been found. If the stored file is then compared with an unrecognized action a message "action does not match" will be displayed. A database is maintained to track the result. Classification of the video gives the final output, which is the specific action of the video.

1.3 OBJECTIVE OF THE PROJECT:

The main objective of the project is to establish an effective action recognition method using analysis of spatiotemporal silhouettes measured during the activities, based on the idea that spatiotemporal variations of human silhouettes encode not only spatial information about body poses at certain instants, but also dynamic information about global body motion and the motions of local body parts.

LITERATURE REVIEW

FEASIBILITY STUDY

CURRENT STATUS OF THE PROBLEM

CHAPTER 2

LITERATURE REVIEW

2.1 FEASIBILITY STUDY:

2.1.1 CURRENT STATUS OF THE PROBLEM:

The recognition of human activities has a wide range of promising applications such as smart surveillance, perceptual interfaces, interpretation of sport events, etc. Although there has been much work on human motion analysis over the past two decades activity understanding still remains challenging and also it is available only for high quality video.

Problem exists in extracting useful information from raw video data. Methods like key frame extraction, the computation of optical flow, space time gradients and local descriptors does not provide for useful extraction because of lack of motion information's, intensity based features unreliable in case of low quality video, smooth surfaces motion discontinuities and singularities.

Shape and kinematics are two important cues in human movements. It is difficult to accurately extract kinematics from real videos using current imperfect vision techniques.

PROPOSED SYSTEM AND ADVANTAGES

2.1.2 PROPOSED SYSTEM AND ITS ADVANTAGES:

The proposed method has several desirable properties: a) It is easier to comprehend and implement, without the requirements of explicit feature tracking and complex probabilistic modeling of motion patterns. b) Being based on binary silhouette analysis, it naturally avoids some problems arising in most previous methods. c) It obtains good results on a large and challenging database and exhibits considerable robustness.

This method is able to not only recognize human actions effectively, but also considerably tolerate some challenging conditions, e.g., partial occlusion, low quality videos, changes in view points, scales and clothes; within class variations caused by different subjects with different physical build ; styles of motions; etc. It also derives a compact trajectory description to reflect the characteristics of motion patterns.

Performance evaluation is carried out on a recently reported database, with size similar or larger to those of most action databases currently in use, in terms of the number of actions, subjects and videos, and good results with considerable roubustness are obtained which demonstrates that silhouettes are indeed informative for characterization and recognition of human motions.

HARDWARE REQUIREMENTS

SOFTWARE REQUIREMENTS

SOFTWARE OVERVIEW

2.2 HARDWARE REQUIREMENTS:

System Type	: X86-based PC
Physical Memory	: 256 MB
Processor	: Pentium IV
Hard Disk	: 40 Gb
Monitor	: 15 VGA colour

2.3 SOFTWARE REQUIREMENTS:

Operating System	: Windows XP
Environment	: Visual Studio .Net 2005
Language	: C#

2.4 SOFTWARE OVERVIEW:

VISUAL STUDIO .NET INTRODUCTION:

Microsoft Visual Studio.Net used as front end tool.

The reason for selecting Visual Studio dot Net as front end tool as follows:

- Visual Studio .Net has flexibility, allowing one or more language to interoperate to provide the solution. This Cross Language Compatibility allows to do project at faster rate.
- Visual Studio. Net has Common Language Runtime that allows the entire component to converge into one intermediate format and then can interact.

- Visual Studio. Net has provide excellent security when your application is executed in the system
- Visual Studio.Net has flexibility, allowing us to configure the working environment to best suit our individual style. We can choose between a single and multiple document interfaces, and we can adjust the size and positioning of the various IDE elements.
- Visual Studio. Net has Intelligence feature that make the coding easy and also dynamic help provides very less coding time.
- The working environment in Visual Studio.Net is often referred to as Integrated Development Environment because it integrates many different functions such as design, editing, compiling and debugging within a common environment. In most traditional development tools, each of separate program, each with its own interface.
- The Visual Studio.Net language is quite powerful – if we can imagine a programming task and accomplished using Visual Basic .Net.
- After creating a Visual Studio. Net application, if we want to distribute it to others we can freely distribute any application to anyone who uses Microsoft windows.
- Toolbars provide quick access to commonly used commands in the programming environment. We click a button on the toolbar once to carry out the action represented by that button. By default, the standard toolbar is displayed when we start Visual Basic. Additional toolbars for editing, form design, and debugging can be toggled on or off from the toolbars command on the view menu.
- Many parts of Visual Studio are context sensitive. Context sensitive means we can get help on these parts directly without having to go

through the help menu. For example, to get help on any keyword in the Visual Basic language, place the insertion point on that keyword in the code window and press F1.

- Visual Studio interprets our code as we enter it, catching and highlighting most syntax or spelling errors on the fly.

C# INTRODUCTION:

C# is an object-oriented programming language developed by Microsoft as part of the .NET initiative and later approved as a standard by ECMA (**ECMA-334**) and ISO (**ISO/IEC 23270**).C# has a procedural, object-oriented syntax based on C++ and includes influences from aspects of several other programming languages (most notably Delphi and Java) with a particular emphasis on simplification.

COMMON TYPE SYSTEM:

C# has a unified type system. This unified type system is called Common Type System.A unified type system implies that all types, including primitives such as integers, are subclasses of the System.Object class. For example, every type inherits a ToString() method.

CATEGORIES OF DATA TYPES:

CTS separates datatypes into two categories:

- Value Type
- Reference Type

While value types are those in which the value itself is stored by allocating memory on the stack, reference types are those in which only the address to the location where the value is present, is stored. Value types include integers (short, long), floating-point numbers (float, double), decimal (a base 10 number used for financial calculations), structures, enumerators, booleans and characters while reference types include objects, strings, classes, interfaces and delegates.

USER DEFINED DATA TYPES:

C# also allows the programmer to create user-defined value types, using the struct keyword. From the programmer's perspective, they can be seen as lightweight classes. Unlike regular classes, and like the standard primitives, such value types are allocated on the stack rather than on the heap. They can also be part of an object (either as a field or boxed), or stored in an array, without the memory indirection that normally exists for class types.

Structs also come with a number of limitations. Because structs have no notion of a *null* value and can be used in arrays without initialization, they are implicitly initialized to default values (normally by filling the struct memory space with zeroes, but the programmer can specify explicit default values to override this).

The programmer can define additional constructors with one or more arguments. This also means that structs lack a virtual method table, and because of that (and the fixed memory footprint), they cannot allow inheritance (but can implement interfaces).

FEATURES OF C#:

- Partial classes allow class implementation across more than one source file. This permits splitting up very large classes, and is also useful if some parts of a class are automatically generated.

- Generics or parameterized types. This is a .NET 2.0 feature supported by C#. Unlike C++ templates, .NET parameterized types are instantiated at runtime rather than by the compiler; hence they can be cross-language whereas C++ templates cannot.
- Static classes that cannot be instantiated, and that only allow static members. This is similar to the concept of module in many procedural languages.
- A new form of iterator that provides generator functionality, using a `yield return` construct similar to `yield` in Python.
- Anonymous delegates providing closure functionality.
- Covariance and contravariance for signatures of delegates
- The accessibility of property accessors can be set independently.
- Nullable value types (denoted by a question mark, e.g. `int i = null;`) which add `null` to the set of allowed values for any value type. This provides improved interaction with SQL databases, which can have nullable columns of types corresponding to C# primitive types: an SQL `INTEGER NULL` column type directly translates to the C# `int`.

MERITS OF C#:

- XML documentation generated from source code comments.
- Operator overloading .
- Language support for unsigned types (you can use them from VB.NET, but they aren't in the language itself). Again, support for these is coming to VB.NET in Whidbey.
- The `using` statement, which makes unmanaged resource disposal simple.
- Explicit interface implementation, where an interface which is already implemented in a base class can be reimplemented separately in a derived class.

DETAILS OF METHODOLOGY EMPLOYED

CHAPTER 3

DETAILS OF METHODOLOGY EMPLOYED

MODULES OF THE PROJECT:

The project has been mainly analyzed into five different modules which have to be implemented and integrated .The various modules are as follows:

1. Pre-processing Module
2. Conversion Module
3. Action Recognition Module
4. Action Detection Module
5. Output Module

Now let's go into the detail analysis of the various modules involved in the project

1. PRE-PROCESSING MODULE:

This is the first module. This module is to convert the input video to images. And we have to do the image enhancement (i.e.: Noise removal etc...). Image got from the video is extracted into frames. We exploit locality preserving projections(LPP) for dimensionality reduction, leading to a low dimensional embedding of human movements.

2. CONVERSION MODULE:

Edges can be detected from the frames. For detection, we convert the image to black and white using grayscale .For edge detection we have to use canny edge detection algorithm. Given a sequence of moving silhouettes associated to an action video, by LPP we project them in to a low dimensional space to characterize the spatiotemporal property of the action, as well as to preserve much of the geometric structure.

3. ACTION RECOGNITION MODULE:

We have to store the edges into the Files. This module is to find the human action with the use of stored files. The human action can be recognized by the methods like the feature tracking based methods, intensity or gradient based methods and the methods of dimensionality reduction. The action recognition can be solved through measuring motion similarities between the reference motion patterns and test samples in the low dimensional embedding space.

4. ACTION DETECTION MODULE:

Comparison with the files in this section with the input video. This module will do the main processing in this project. A message "action found" will be displayed if any match has been found. If the stored file is then compared with an unrecognized action a message "action does not match" will be displayed. A database is maintained to track the result.

5. OUTPUT MODULE:

The action for the given input video is displayed as the final output. Action classification is then achieved in a nearest neighbor framework. The experimental results show that the proposed method is able to not only recognize human actions effectively, but also considerably tolerate some challenging conditions, eg: low-quality videos, changes in viewpoints, scales, and clothes within-class variations caused by different subjects with different physical build; styles of motion.

DATA FLOW DIAGRAM:

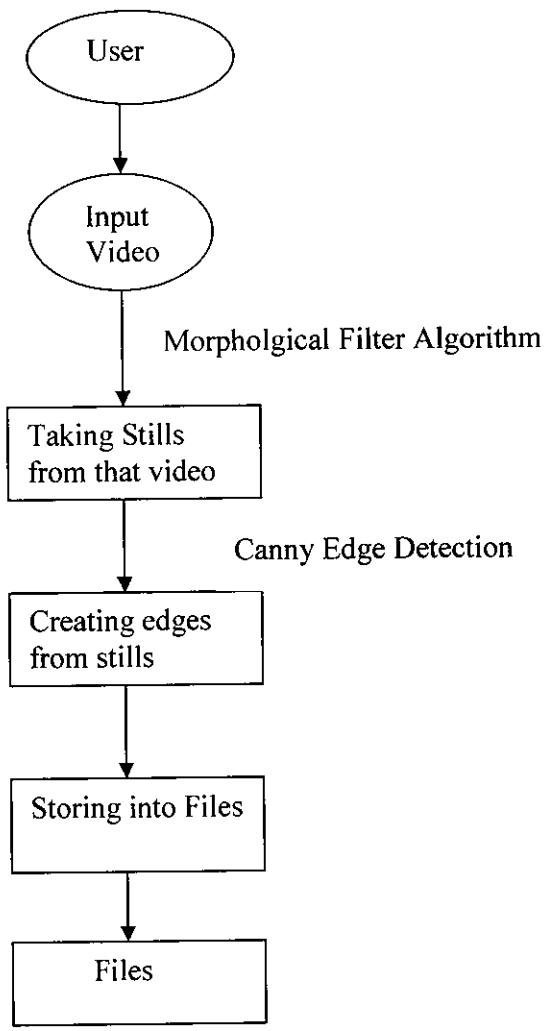


Figure:1

USER DIAGRAM:

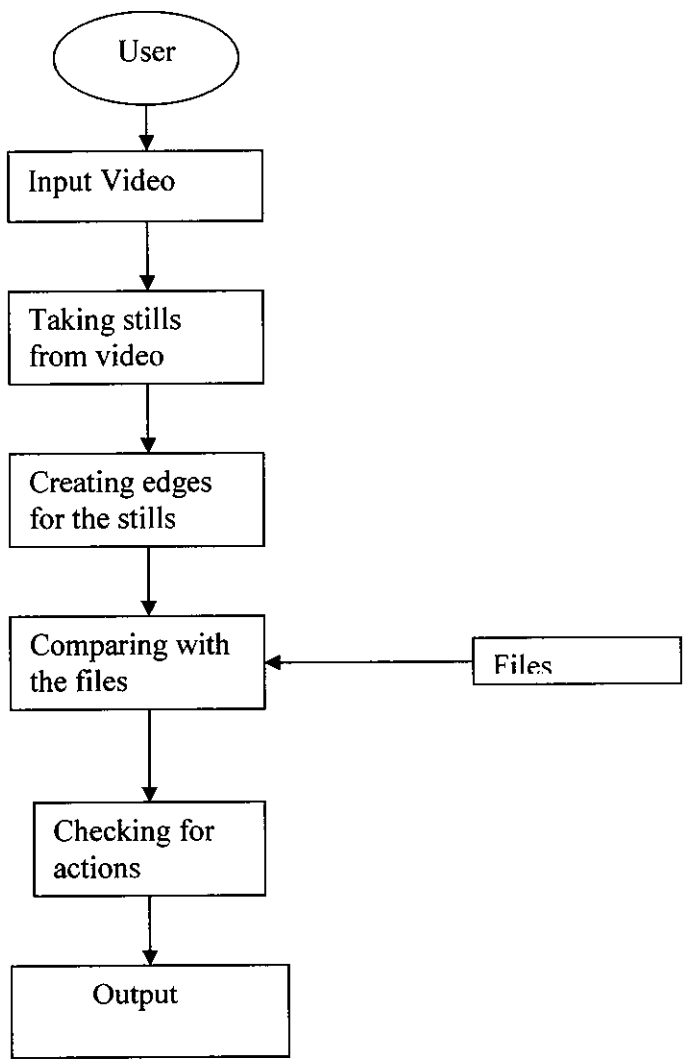


Figure:2

SYSTEM FLOW DIAGRAM:

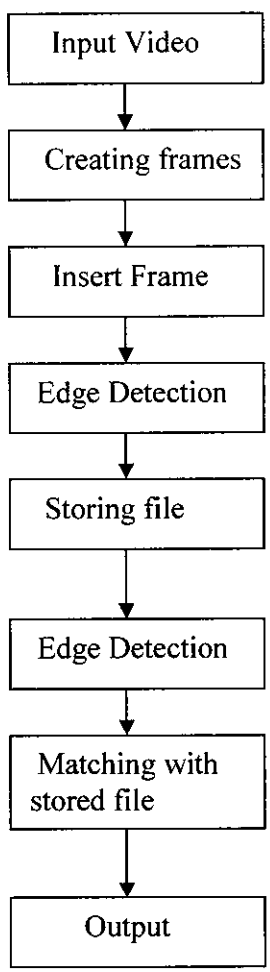


Figure :3

CONCLUSION

CHAPTER 4

CONCLUSION

Thus the project "HUMAN ACTION RECOGNITION SYSTEM" has established an effective method for action recognition using analysis of spatiotemporal silhouettes measured during the activities, based on the idea that spatiotemporal variations of human silhouettes encode not only spatial information about body poses at certain instants, but also dynamic information about global body motion and the motions of local body parts.

The method has tolerated some challenging conditions like partial occlusion, low quality videos, changes in view points, scales and clothes, within class variations caused by different subjects with different physical build; styles of motions. It derives a compact trajectory description to reflect the characteristics of motion patterns.

Considerable robustness is also obtained using this method. It became easier to comprehend and implement, without the requirements of explicit feature tracking.

It also avoided problems arising in most previous methods,e.g. unreliable 2-D or 3-D tracking, expensive and sensitive optical flow computations etc., it obtained good results on a large and challenging database and exhibits considerable robustness.

FUTURE ENHANCEMENT

CHAPTER 5
FUTURE ENHANCEMENTS

Human action recognition has gained increasing interest in the computer vision community. Compared with other extensively studied topics such as human detection, tracking and recognition, human activity understanding is in its infancy due to the complexity and variety of actions.

This paper has proposed a simple but effective method for human action recognition. The major core is based on low-dimensional embedding Representations of dynamic silhouettes obtained from the action videos. Extensive experimental results have validated the powerful abilities of the proposed method.

Although the experiments have demonstrated that our methodology works effectively, further evaluation on a larger database, with multivariate actions, subjects and scenarios, needs to be carried out. Both shape and kinematics information derived from actions play important roles in human motion analysis.

Fusion of two cues is thus, preferable for improving the accuracy and reliability. Most current work on action and motion interpretation remains rooted in view dependent representations.

Although there have been some attempts for this problem, they usually use the epipolar geometry between the views of two or more cameras to perform view invariant recognition. How to extract view information features still remains challenging. We also plan to test our algorithm with a spatiotemporal extension to isomap or to augment other dimension reduction methods with temporal relation for dealing with the problem of manifold learning of dynamic data.

APPENDICES

CHAPTER 6

APPENDICES

6.1 SAMPLE CODE:

VIDEO TO IMAGE CONVERSION:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Runtime.InteropServices;

namespace JockerSoft.Media
{
   public class Form1 : System.Windows.Forms.Form
   {
      FrameGrabber frmg = new FrameGrabber();

      public static int bsize=10;
      private System.Windows.Forms.TextBox textBox1;
      private System.Windows.Forms.TextBox textBox2;
      private System.Windows.Forms.Label label1;
      private System.Windows.Forms.Label label2;
      private System.Windows.Forms.Button button1;
      private System.Windows.Forms.OpenFileDialog openFileDialog1;
      private System.Windows.Forms.Button button3;
      private GroupBox groupBox1;
      private TrackBar trackBar1;
      private PictureBox pictureBox1;
      private Label label3;
      private TextBox textBox3;
      private Button button2;
      private ProgressBar progressBar1;
```

```
private System.ComponentModel.Container components = null;

public Form1()
{

   InitializeComponent();

}

protected override void Dispose(bool disposing)
{
   if (disposing)
   {
      if (components != null)
      {
         components.Dispose();
      }
   }
   base.Dispose(disposing);
}

#region Windows Form Designer generated code
private void InitializeComponent()
{
   this.textBox1 = new System.Windows.Forms.TextBox();
   this.textBox2 = new System.Windows.Forms.TextBox();
   this.label1 = new System.Windows.Forms.Label();
   this.label2 = new System.Windows.Forms.Label();
   this.button1 = new System.Windows.Forms.Button();
   this.openFileDialog1 = new
System.Windows.Forms.OpenFileDialog();
   this.button3 = new System.Windows.Forms.Button();



   this.groupBox1 = new System.Windows.Forms.GroupBox();
```

```
        this.trackBar1 = new System.Windows.Forms.TrackBar();
        this.pictureBox1 = new System.Windows.Forms.PictureBox();
        this.label3 = new System.Windows.Forms.Label();
        this.textBox3 = new System.Windows.Forms.TextBox();
        this.button2 = new System.Windows.Forms.Button();
        this.progressBar1 = new System.Windows.Forms.ProgressBar();
        this.groupBox1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit(
);
        this.SuspendLayout();
          // textBox1
        this.textBox1.Location = new System.Drawing.Point(8, 94);
        this.textBox1.Name = "textBox1";
        this.textBox1.Size = new System.Drawing.Size(156, 20);
        this.textBox1.TabIndex = 7;
        this.textBox1.Enter += new
 System.EventHandler(this.textBox1_Enter);
        this.textBox1.MouseUp += new
 System.Windows.Forms.MouseEventHandler(this.textBox1_MouseUp);

        this.textBox2.Location = new System.Drawing.Point(8, 162);
        this.textBox2.Name = "textBox2";
        this.textBox2.Size = new System.Drawing.Size(160, 20);
        this.textBox2.TabIndex = 8;




        this.textBox2.Text = "c:\\frame.bmp";
        //
        // label1
        //
        this.label1.Location = new System.Drawing.Point(9, 75);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(100, 16);
        this.label1.TabIndex = 9;
```

```
this.label1.Text = "Video Path";
//
// label2
//
this.label2.Location = new System.Drawing.Point(9, 143);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(100, 16);
this.label2.TabIndex = 10;
this.label2.Text = "Image Path";
//
// button1
//
this.button1.Location = new System.Drawing.Point(140, 70);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(24, 23);
this.button1.TabIndex = 11;
this.button1.Text = "...";
this.button1.Click += new System.EventHandler(this.button1_Click);
//
// button3
//
this.button3.Location = new System.Drawing.Point(49, 264);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(75, 23);




this.button3.TabIndex = 13;
this.button3.Text = "Extract";
this.button3.Click += new System.EventHandler(this.button3_Click);
//
// groupBox1
//
this.groupBox1.Controls.Add(this.trackBar1);
this.groupBox1.Controls.Add(this.pictureBox1);
this.groupBox1.Location = new System.Drawing.Point(182, 7);
this.groupBox1.Name = "groupBox1";
this.groupBox1.Size = new System.Drawing.Size(631, 424);
```

```
        this.groupBox1.TabIndex = 14;
        this.groupBox1.TabStop = false;
        this.groupBox1.Enter += new
System.EventHandler(this.groupBox1_Enter);
        //
        // trackBar1
        //
        this.trackBar1.Anchor =
((System.Windows.Forms.AnchorStyles)(((System.Windows.Forms.Anchor
Styles.Bottom | System.Windows.Forms.AnchorStyles.Left)
                | System.Windows.Forms.AnchorStyles.Right)));
        this.trackBar1.Location = new System.Drawing.Point(9, 376);
        this.trackBar1.Maximum = 1000;
        this.trackBar1.Name = "trackBar1";
        this.trackBar1.Size = new System.Drawing.Size(593, 45);
        this.trackBar1.TabIndex = 8;
        this.trackBar1.TickFrequency = 5;
        this.trackBar1.TickStyle = System.Windows.Forms.TickStyle.Both;
        //
        // pictureBox1
        //
        this.pictureBox1.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.Ancho
rStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)
                | System.Windows.Forms.AnchorStyles.Left)
                | System.Windows.Forms.AnchorStyles.Right)));
        this.pictureBox1.BackColor =
System.Drawing.SystemColors.ControlText;
        this.pictureBox1.Location = new System.Drawing.Point(9, 21);
        this.pictureBox1.Name = "pictureBox1";
        this.pictureBox1.Size = new System.Drawing.Size(593, 351);
        this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.CenterImage;
        this.pictureBox1.TabIndex = 7;
        this.pictureBox1.TabStop = false;
```

```
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(9, 206);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(115, 13);
this.label3.TabIndex = 15;
this.label3.Text = "Frame  Extraction Rate";
//
// textBox3
//
this.textBox3.Location = new System.Drawing.Point(12, 222);
this.textBox3.Name = "textBox3";
this.textBox3.Size = new System.Drawing.Size(156, 20);




this.textBox3.TabIndex = 16;
this.textBox3.Text = "10";
//
// button2
//
this.button2.Location = new System.Drawing.Point(144, 138);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(24, 23);
this.button2.TabIndex = 17;
this.button2.Text = "...";
this.button2.Click += new System.EventHandler(this.button2_Click);
//
// progressBar1
//
this.progressBar1.Location = new System.Drawing.Point(182, 435);
this.progressBar1.Name = "progressBar1";
this.progressBar1.Size = new System.Drawing.Size(631, 18);
this.progressBar1.TabIndex = 18;
//
// Form1
```

```
        //
        this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
        this.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(192)))),
((int)(((byte)(192)))), ((int)(((byte)(255)))));
        this.ClientSize = new System.Drawing.Size(825, 465);
        this.Controls.Add(this.progressBar1);
        this.Controls.Add(this.button2);
        this.Controls.Add(this.textBox3);
        this.Controls.Add(this.label3);
        this.Controls.Add(this.groupBox1);
        this.Controls.Add(this.button3);
        this.Controls.Add(this.button1);




        this.Controls.Add(this.label2);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.textBox2);
        this.Controls.Add(this.textBox1);
        this.Name = "Form1";
        this.Text = "FRAME EXTRACTION ";
        this.Load += new System.EventHandler(this.Form1_Load);
        this.groupBox1.ResumeLayout(false);
        this.groupBox1.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();

    }
    #endregion


    [STAThread]
```

```
    private void trackBar1_Scroll(object sender, System.EventArgs e)
    {
      try
      {
        this.pictureBox1.Image =
FrameGrabber.GetFrameFromVideo(this.textBox1.Text,
((double)this.trackBar1.Value) / this.trackBar1.Maximum);
      }
      catch (JockerSoft.Media.InvalidVideoFileException ex)
      {
        MessageBox.Show(ex.Message, "Extraction failed");




      }
    }

    private void textBox1_Enter(object sender, System.EventArgs e)
    {
      this.textBox1.SelectAll();
    }

    private void textBox1_MouseUp(object sender,
System.Windows.Forms.MouseEventArgs e)
    {
      this.textBox1.SelectAll();
    }

    private void button1_Click(object sender, System.EventArgs e)
    {
      if (this.openFileDialog1.ShowDialog() == DialogResult.OK)
        this.textBox1.Text = this.openFileDialog1.FileName;
    }

    private void button3_Click(object sender, System.EventArgs e)
    {
      int i;
```

```
String path;
String pth = Application.StartupPath;
Bitmap bmp;
double v;
int size;
size = Convert.ToInt32(textBox3.Text);
bsize = size;
textBox2.Text = pth.ToString();
progressBar1.Minimum = 0;
progressBar1.Maximum = size -1;

for (i = 1; i <= size; i++)
{




    progressBar1.Value = i-1;
    trackBar1.Value = trackBar1.Value + Convert.ToInt32 (1000/size)
;
    this.pictureBox1.Image =
FrameGrabber.GetFrameFromVideo(this.textBox1.Text,
((double)this.trackBar1.Value) / this.trackBar1.Maximum);



    path = @pth + "\\outputimage\\" + i + ".bmp";
    //v =  (double)( (double) 1.0 / (double) trackBar1.Value);
    v = ((double)this.trackBar1.Value) / this.trackBar1.Maximum;

    //pictureBox1.Image.Save(path);
    bmp = new Bitmap(pictureBox1.Image, pictureBox1.Width,
pictureBox1.Height);
    //  bmp.Save(path);

    try
    {
      textBox2.Text = path;
```

```
            FrameGrabber.SaveFrameFromVideo(this.textBox1.Text, v,
this.textBox2.Text);

          }
          catch (JockerSoft.Media.InvalidVideoFileException ex)
          {
            MessageBox.Show(ex.Message, "Extraction failed");
          }
        }
        MessageBox.Show("Frame extracted to "+ pth.ToString());

      }



      private void Form1_Load(object sender, EventArgs e)
      {
        textBox1.Text = Application.StartupPath + "\\outputimage";

      }

      private void button2_Click(object sender, EventArgs e)
      {
        if (this.openFileDialog1.ShowDialog() == DialogResult.OK)
          this.textBox1.Text = this.openFileDialog1.FileName;
      }

      private void groupBox1_Enter(object sender, EventArgs e)
      {

      }
    }
}
```

CANNY EDGE DETECTOR:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

using ImageLib.Imaging.Filters;

namespace HAR
{
	public class CannyDetectorForm : System.Windows.Forms.Form
	{
```

```
		private CannyEdgeDetector filter = new CannyEdgeDetector();
		private System.Windows.Forms.TextBox sigmaBox;
		private System.Windows.Forms.GroupBox groupBox1;
		private System.Windows.Forms.TrackBar sigmaTrackBar;
		private System.Windows.Forms.GroupBox groupBox2;
		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.Label label2;
		private HAR.ColorSlider thresholdSlider;
		private System.Windows.Forms.GroupBox groupBox3;
		private HAR.FilterPreview filterPreview;
		private System.Windows.Forms.Button cancelButton;
		private System.Windows.Forms.Button okButton;
		private System.Windows.Forms.TextBox highThresholdBox;
		private System.Windows.Forms.TextBox lowThresholdBox;
		private System.ComponentModel.Container components = null;
		public Bitmap Image
		{
			set { filterPreview.Image = value; }
		}
		public IFilter Filter
```

```
        {
            get { return filter; }
        }
        public CannyDetectorForm()
        {
            InitializeComponent();

            sigmaBox.Text = filter.GaussianSigma.ToString();




            lowThresholdBox.Text =
filter.LowThreshold.ToString();
            highThresholdBox.Text =
filter.HighThreshold.ToString();

            // set sliders
            sigmaTrackBar.Value = (int)((filter.GaussianSigma - 1.0)
* 20);
            thresholdSlider.Min = filter.LowThreshold;
            thresholdSlider.Max = filter.HighThreshold;

            filterPreview.Filter = filter;
        }

        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if(components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }
```

```
        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
                this.sigmaBox = new
System.Windows.Forms.TextBox();
                this.groupBox1 = new
System.Windows.Forms.GroupBox();
                this.sigmaTrackBar = new
System.Windows.Forms.TrackBar();
                this.groupBox2 = new
System.Windows.Forms.GroupBox();




                this.thresholdSlider = new HAR.ColorSlider();
                this.highThresholdBox = new
System.Windows.Forms.TextBox();
                this.label2 = new System.Windows.Forms.Label();
                this.lowThresholdBox = new
System.Windows.Forms.TextBox();
                this.label1 = new System.Windows.Forms.Label();
                this.groupBox3 = new
System.Windows.Forms.GroupBox();
                this.filterPreview = new HAR.FilterPreview();
                this.cancelButton = new
System.Windows.Forms.Button();
                this.okButton = new System.Windows.Forms.Button();
                this.groupBox1.SuspendLayout();

        ((System.ComponentModel.ISupportInitialize)(this.sigmaTrackBar)).
BeginInit();
                this.groupBox2.SuspendLayout();
                this.groupBox3.SuspendLayout();
                this.SuspendLayout();
                //
                // sigmaBox
                //
```

```
                this.sigmaBox.Location = new
System.Drawing.Point(10, 20);
                this.sigmaBox.Name = "sigmaBox";
                this.sigmaBox.Size = new System.Drawing.Size(60, 20);
                this.sigmaBox.TabIndex = 1;
                this.sigmaBox.Text = "";
                this.sigmaBox.TextChanged += new
System.EventHandler(this.sigmaBox_TextChanged);
                //
                // groupBox1
                //




                this.groupBox1.Controls.Add(this.sigmaTrackBar);
                this.groupBox1.Controls.Add(this.sigmaBox);
                this.groupBox1.Location = new
System.Drawing.Point(10, 5);
                this.groupBox1.Name = "groupBox1";
                this.groupBox1.Size = new System.Drawing.Size(280,
95);
                this.groupBox1.TabIndex = 2;
                this.groupBox1.TabStop = false;
                this.groupBox1.Text = "Gaussian Sigma";
                //
                // sigmaTrackBar
                //
                this.sigmaTrackBar.Location = new
System.Drawing.Point(10, 45);
                this.sigmaTrackBar.Maximum = 40;
                this.sigmaTrackBar.Name = "sigmaTrackBar";
                this.sigmaTrackBar.Size = new
System.Drawing.Size(260, 45);
                this.sigmaTrackBar.TabIndex = 2;
                this.sigmaTrackBar.TickFrequency = 2;
                this.sigmaTrackBar.ValueChanged += new
System.EventHandler(this.sigmaTrackBar_ValueChanged);
                //
```

```
                // groupBox2
                //
                this.groupBox2.Controls.Add(this.thresholdSlider);
                this.groupBox2.Controls.Add(this.highThresholdBox);
                this.groupBox2.Controls.Add(this.label2);




                this.groupBox2.Controls.Add(this.lowThresholdBox);
                this.groupBox2.Controls.Add(this.label1);
                this.groupBox2.Location = new
System.Drawing.Point(10, 110);
                this.groupBox2.Name = "groupBox2";
                this.groupBox2.Size = new System.Drawing.Size(280,
80);
                this.groupBox2.TabIndex = 3;
                this.groupBox2.TabStop = false;
                this.groupBox2.Text = "Threshold Values";
                //
                // thresholdSlider
                //
                this.thresholdSlider.Location = new
 System.Drawing.Point(8, 50);
                this.thresholdSlider.Name = "thresholdSlider";
                this.thresholdSlider.Size = new
 System.Drawing.Size(262, 23);
                this.thresholdSlider.TabIndex = 4;
                this.thresholdSlider.Text = "colorSlider1";
                this.thresholdSlider.ValuesChanged += new
 System.EventHandler(this.thresholdSlider_ValuesChanged);
                //
                // highThresholdBox
                //
                this.highThresholdBox.Location = new
 System.Drawing.Point(155, 20);
                this.highThresholdBox.Name = "highThresholdBox";
```

```
                this.highThresholdBox.Size = new
System.Drawing.Size(50, 20);
                this.highThresholdBox.TabIndex = 3;
                this.highThresholdBox.Text = "";




                this.highThresholdBox.TextChanged += new
System.EventHandler(this.highThresholdBox_TextChanged);
                //
                // label2
                //
                this.label2.Location = new System.Drawing.Point(120,
23);
                this.label2.Name = "label2";
                this.label2.Size = new System.Drawing.Size(35, 14);
                this.label2.TabIndex = 2;
                this.label2.Text = "High:";
                //
                // lowThresholdBox
                //
                this.lowThresholdBox.Location = new
System.Drawing.Point(45, 20);
                this.lowThresholdBox.Name = "lowThresholdBox";
                this.lowThresholdBox.Size = new
System.Drawing.Size(50, 20);
                this.lowThresholdBox.TabIndex = 1;
                this.lowThresholdBox.Text = "";
                this.lowThresholdBox.TextChanged += new
System.EventHandler(this.lowThresholdBox_TextChanged);
                //
                // label1
                //
                this.label1.Location = new System.Drawing.Point(10,
23);
                this.label1.Name = "label1";
                this.label1.Size = new System.Drawing.Size(35, 14);
```

```
            this.label1.TabIndex = 0;
            this.label1.Text = "Low:";
            //
            // groupBox3



            //
            this.groupBox3.Controls.Add(this.filterPreview);
            this.groupBox3.Location = new
System.Drawing.Point(300, 5);
            this.groupBox3.Name = "groupBox3";
            this.groupBox3.Size = new System.Drawing.Size(170,
185);
            this.groupBox3.TabIndex = 9;
            this.groupBox3.TabStop = false;
            this.groupBox3.Text = "Preview";
            //
            // filterPreview
            //
            this.filterPreview.Image = null;
            this.filterPreview.Location = new
System.Drawing.Point(10, 15);
            this.filterPreview.Name = "filterPreview";
            this.filterPreview.Size = new System.Drawing.Size(150,
150);
            this.filterPreview.TabIndex = 13;
            this.cancelButton.DialogResult =
System.Windows.Forms.DialogResult.Cancel;
            this.cancelButton.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
            this.cancelButton.Location = new
System.Drawing.Point(244, 205);
            this.cancelButton.Name = "cancelButton";
            this.cancelButton.TabIndex = 13;
            this.cancelButton.Text = "Cancel";
            //
            // okButton
            //
```

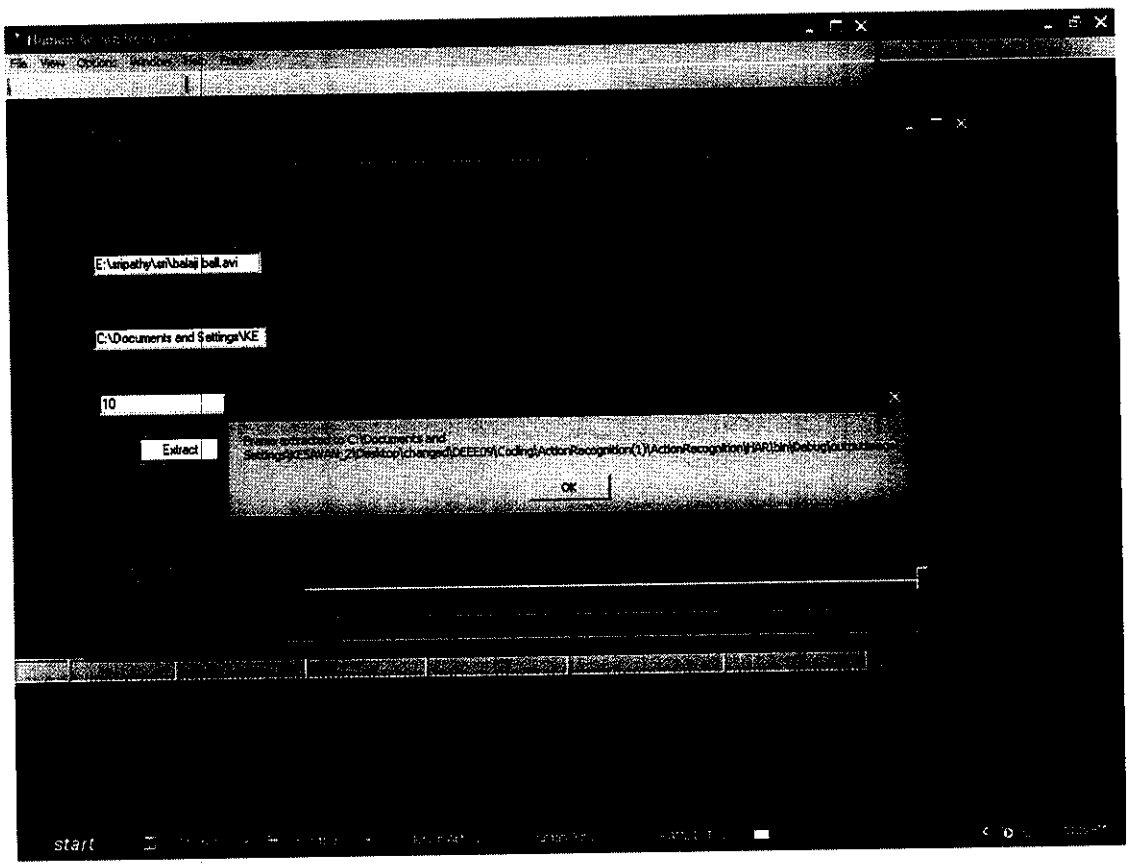
```
                this.okButton.DialogResult =
System.Windows.Forms.DialogResult.OK;
                this.okButton.FlatStyle =
System.Windows.Forms.FlatStyle.Flat;
                this.okButton.Location = new
System.Drawing.Point(159, 205);




                this.okButton.Name = "okButton";
                this.okButton.TabIndex = 12;
                this.okButton.Text = "Ok";
                this.AcceptButton = this.okButton;
                this.AutoScaleBaseSize = new System.Drawing.Size(5,
13);
                this.CancelButton = this.cancelButton;
                this.ClientSize = new System.Drawing.Size(479, 236);
                this.Controls.Add(this.cancelButton);
                this.Controls.Add(this.okButton);
                this.Controls.Add(this.groupBox3);
                this.Controls.Add(this.groupBox2);
                this.Controls.Add(this.groupBox1);
                this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedToolWindow;
                this.MaximizeBox = false;
                this.MinimizeBox = false;
                this.Name = "CannyDetectorForm";
                this.ShowInTaskbar = false;
                this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
                this.Text = "Canny Edge Detector";
                this.groupBox1.ResumeLayout(false);

        ((System.ComponentModel.ISupportInitialize)(this.sigmaTrackBar)).
EndInit();
                this.groupBox2.ResumeLayout(false);
                this.groupBox3.ResumeLayout(false);
                this.ResumeLayout(false);
```

```
        }
        #endregion

        private void sigmaTrackBar_ValueChanged(object sender,
System.EventArgs e)
        {




            double v = (double) sigmaTrackBar.Value / 20 + 1.0;

            sigmaBox.Text = v.ToString();
        }
        private void thresholdSlider_ValuesChanged(object sender,
System.EventArgs e)
        {
            lowThresholdBox.Text =
thresholdSlider.Min.ToString();
            highThresholdBox.Text =
thresholdSlider.Max.ToString();
        }

        // Sigma changed
        private void sigmaBox_TextChanged(object sender,
System.EventArgs e)
        {
            try
            {
                filter.GaussianSigma =
double.Parse(sigmaBox.Text);

                filterPreview.RefreshFilter();
            }
            catch (Exception)
            {
            }
```

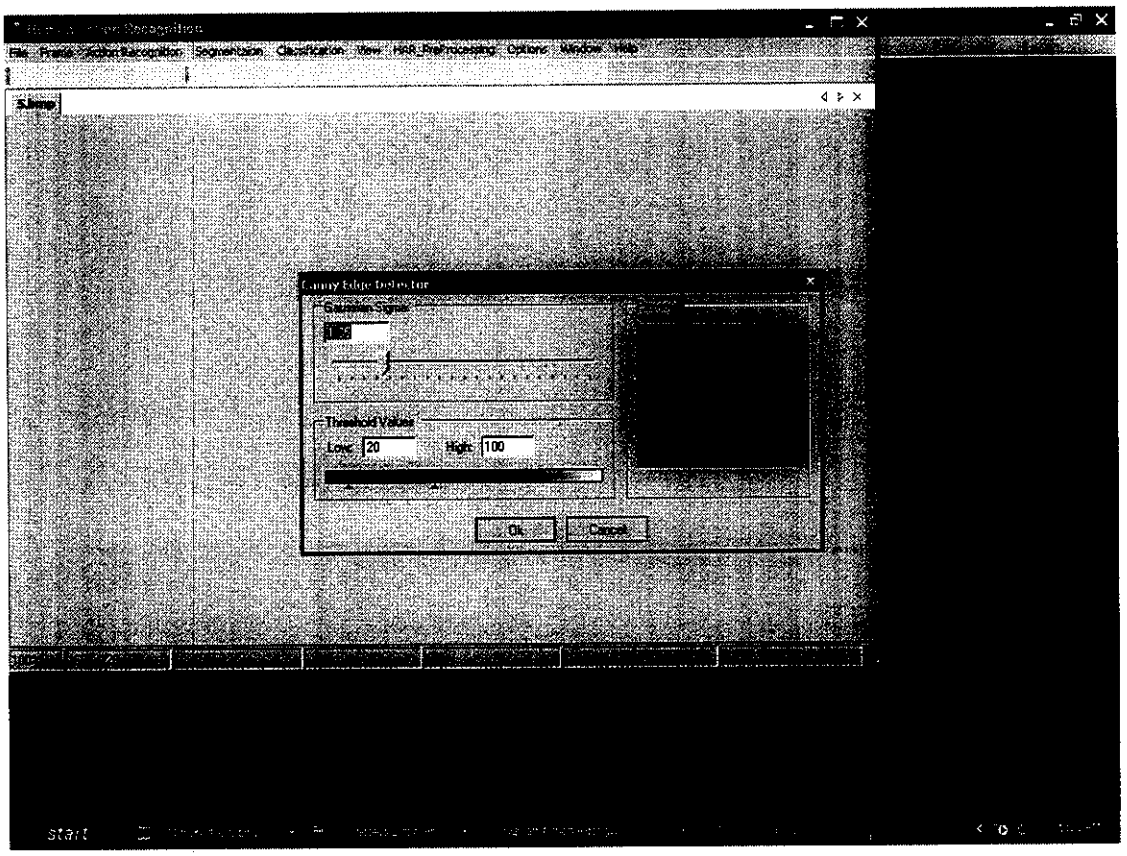
```
            }

            // Low threshold value changed
            private void lowThresholdBox_TextChanged(object sender,
System.EventArgs e)
            {
                  try
                  {
                        filter.LowThreshold =
byte.Parse(lowThresholdBox.Text);

                        filterPreview.RefreshFilter();
                  }
                  catch (Exception)
                  {
                  }
            }

            // Hight threshold value changed
            private void highThresholdBox_TextChanged(object sender,
System.EventArgs e)
            {
                  try
                  {
                        filter.HighThreshold =
byte.Parse(highThresholdBox.Text);

                        filterPreview.RefreshFilter();
                  }
                  catch (Exception)
                  {
                  }
            }
      }
 }
```

6.2 SCREEN SHOTS:

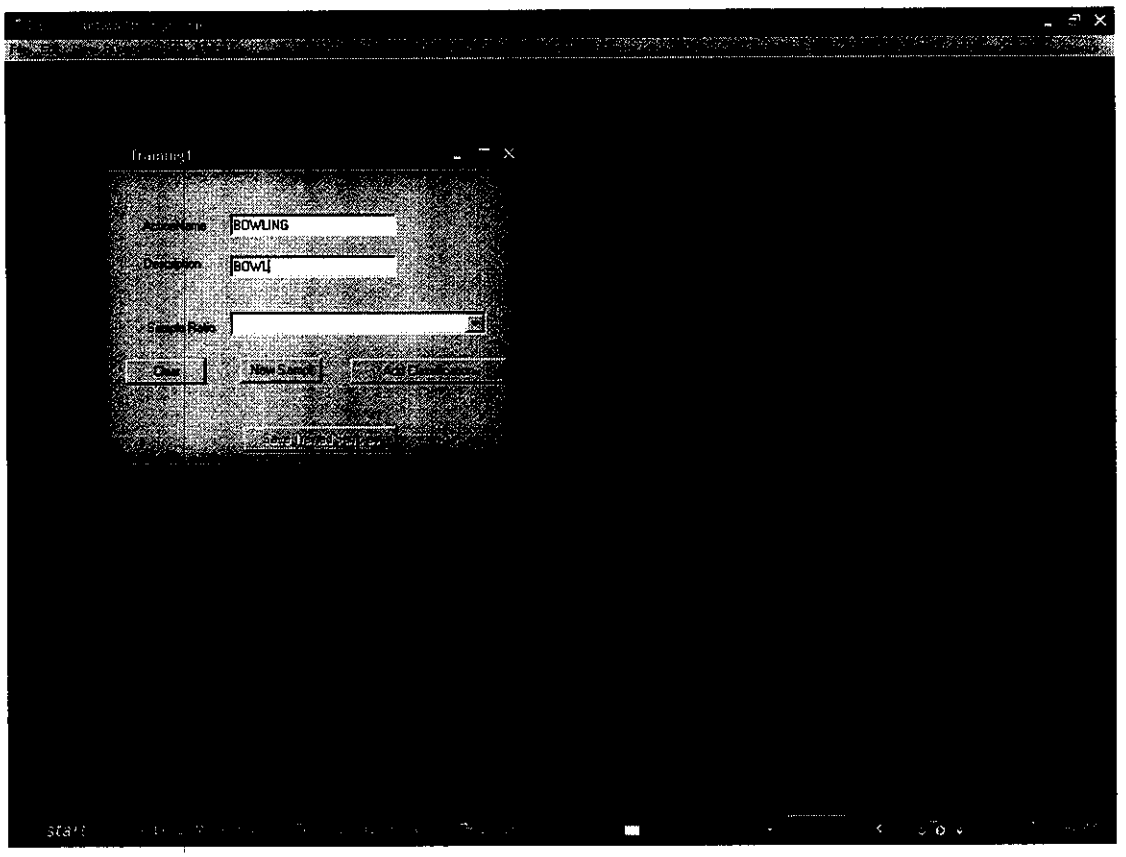
TAKING IMAGES FROM VIDEO:

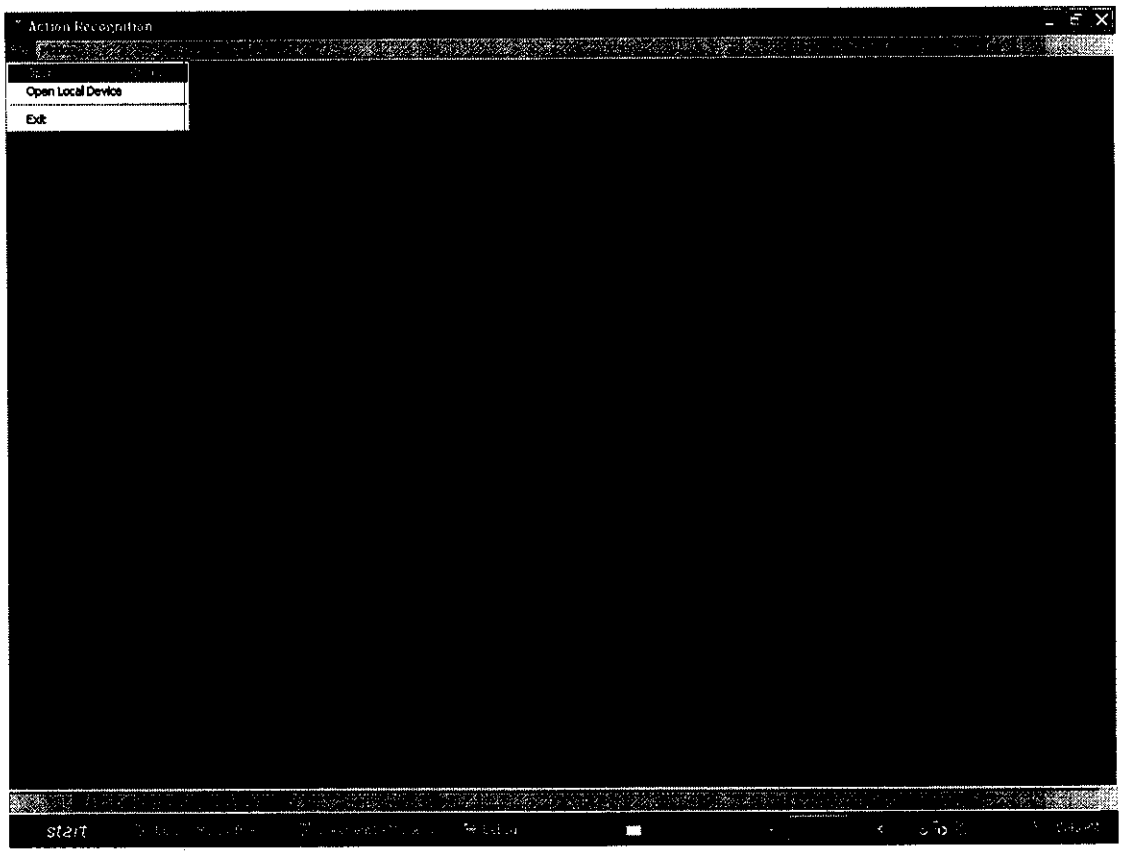


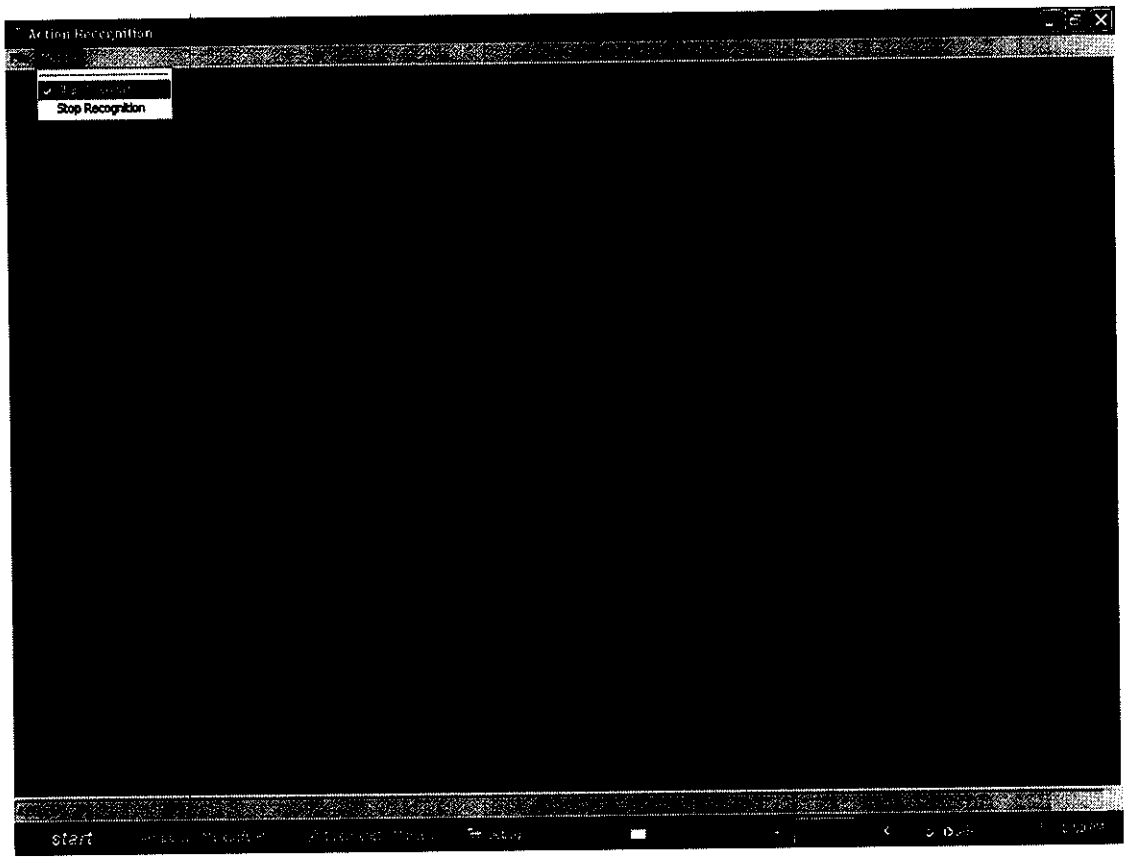
CANNY EDGE DETECTOR:

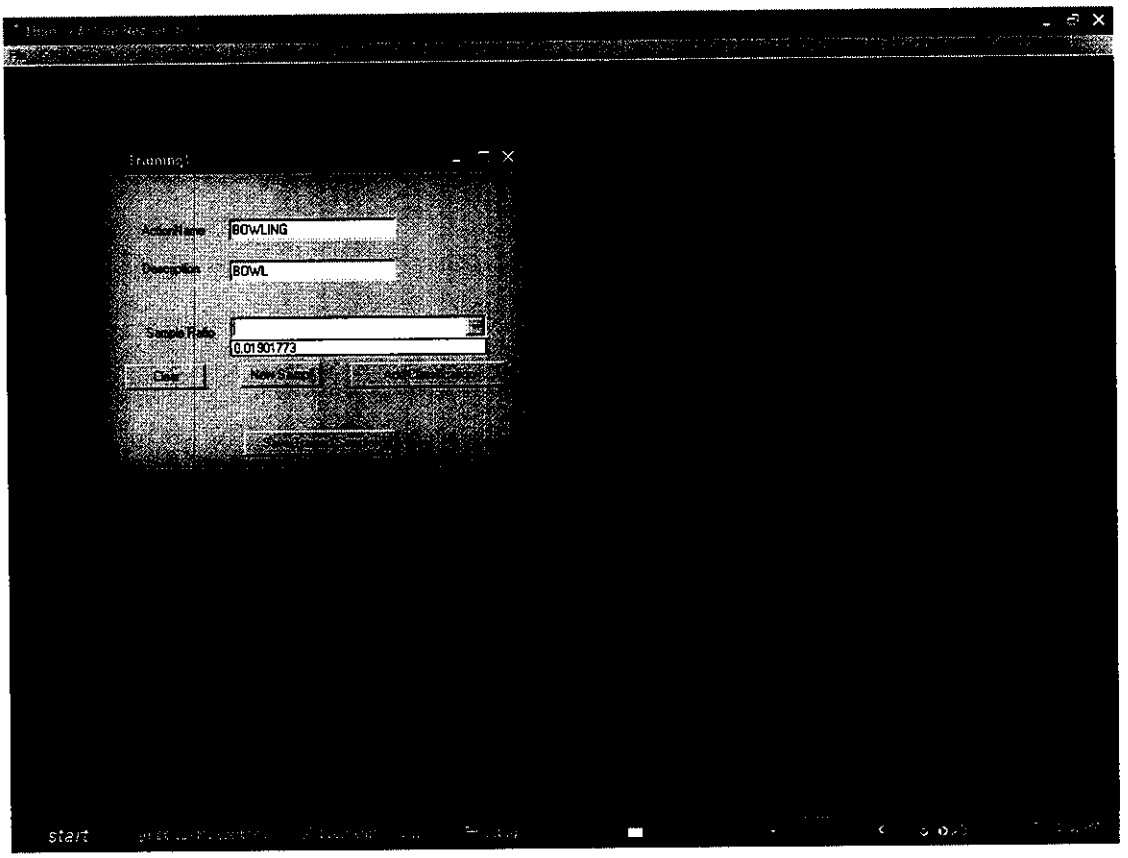


TRAINING:

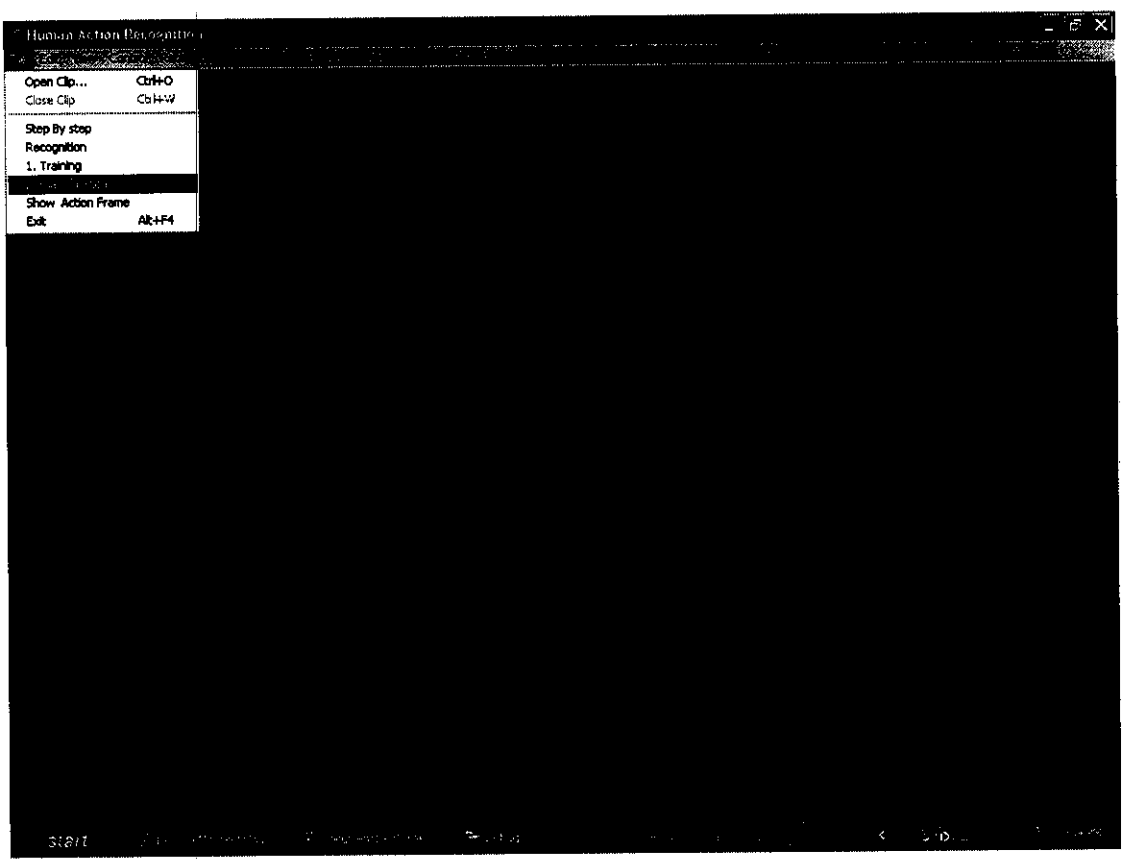


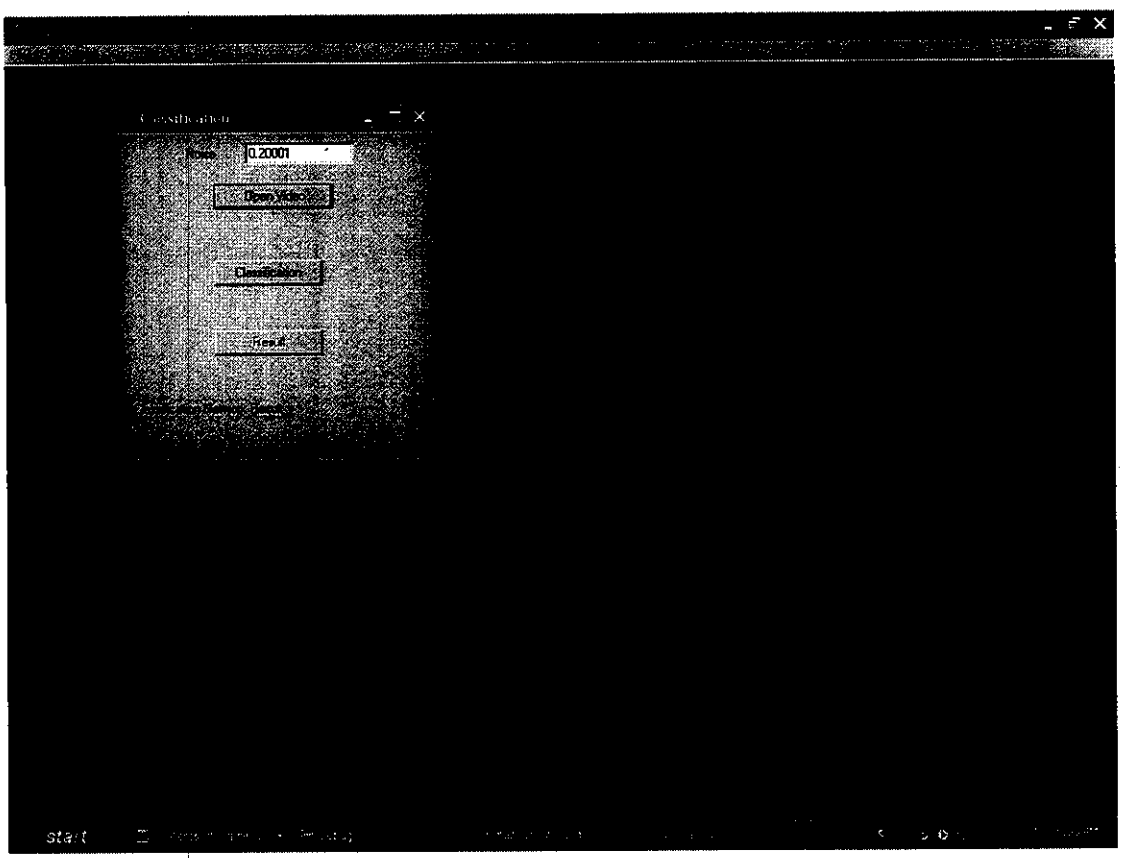

Action Recognition
Open Local Device
Exit
start

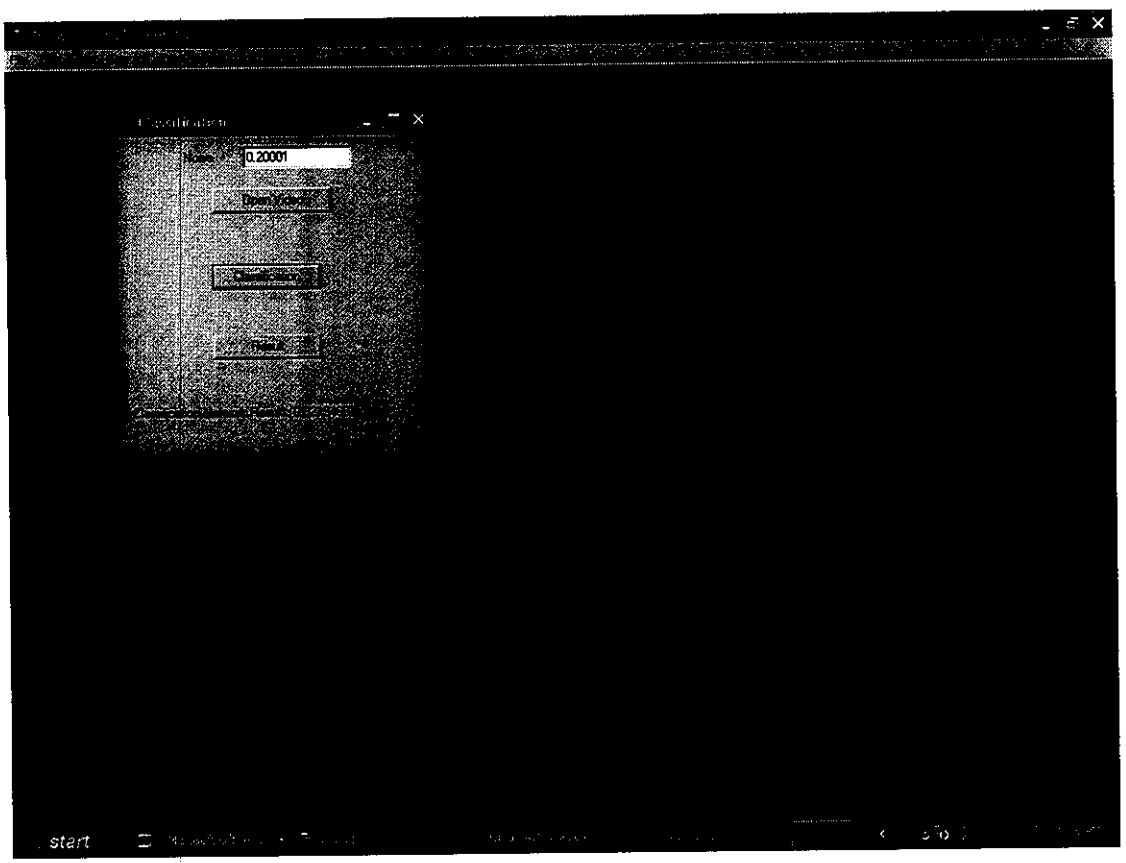

Action Recognition
Stop Recognition
start

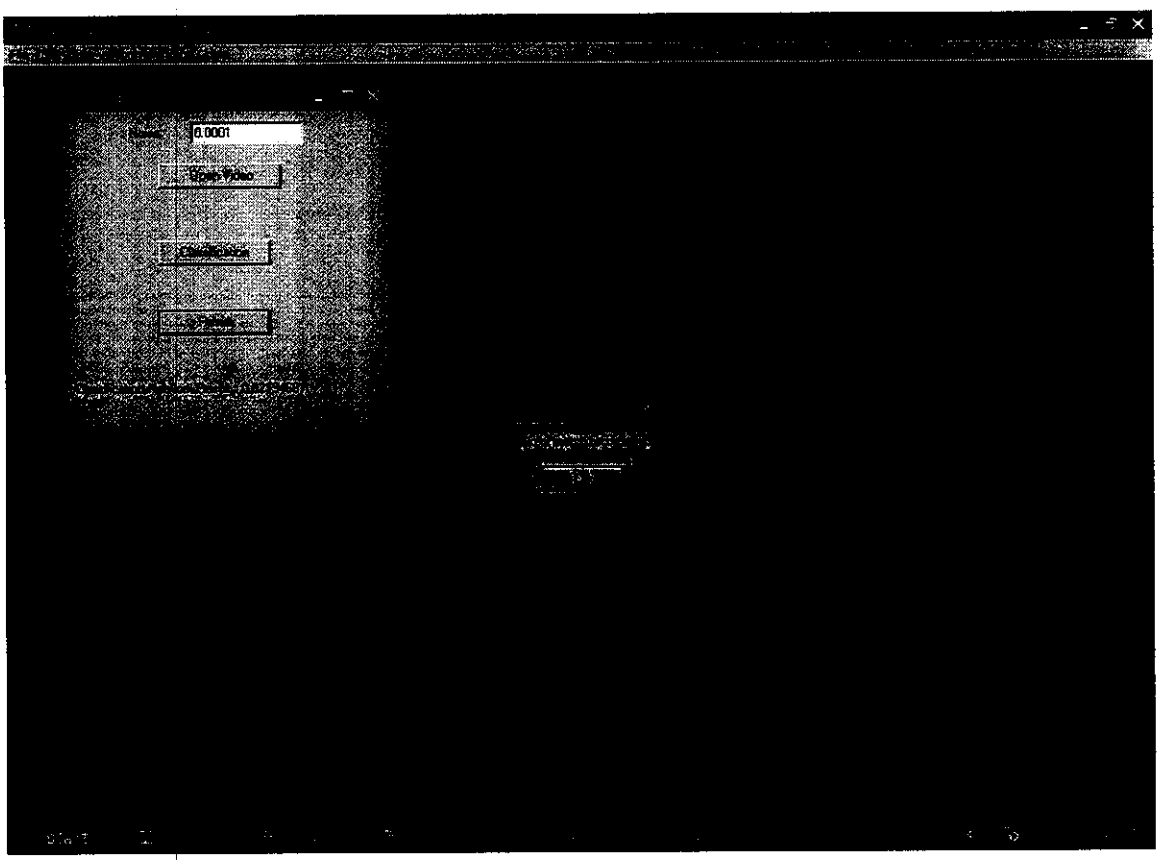

BOWLING
BOWL
0.01901773
start

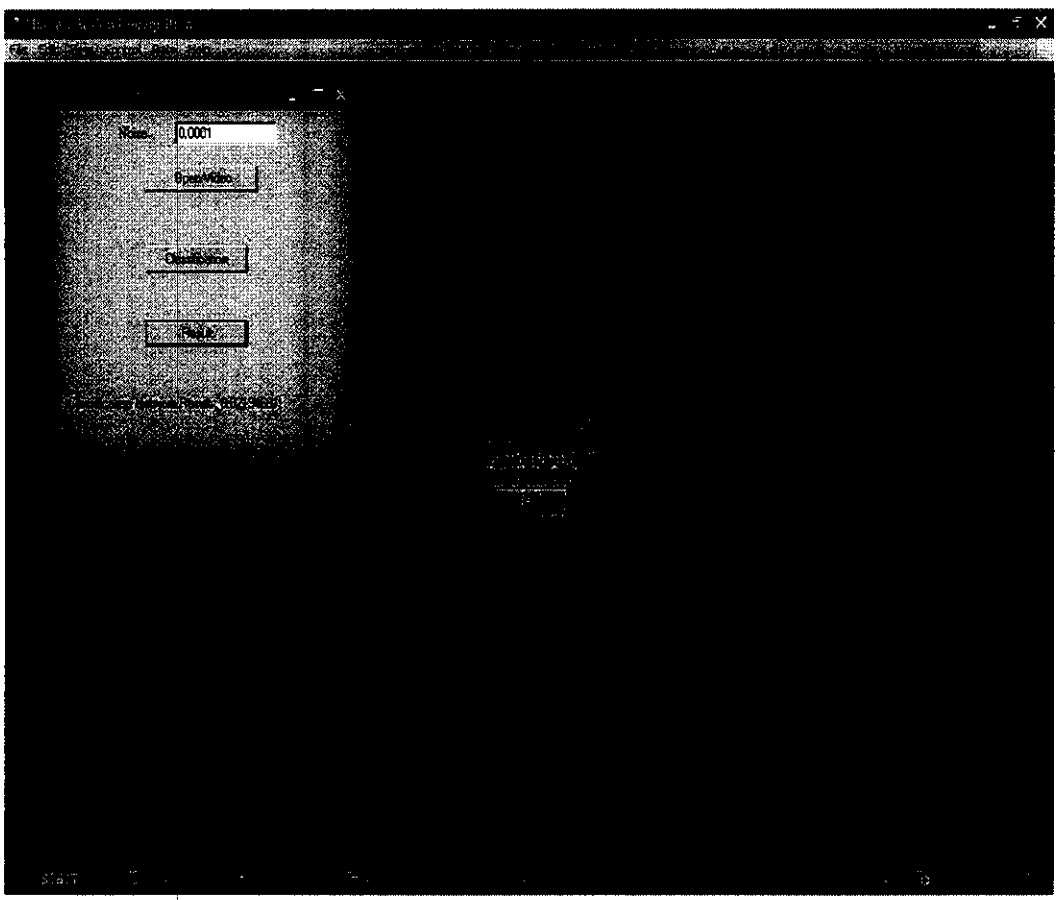
CLASSIFICATION:



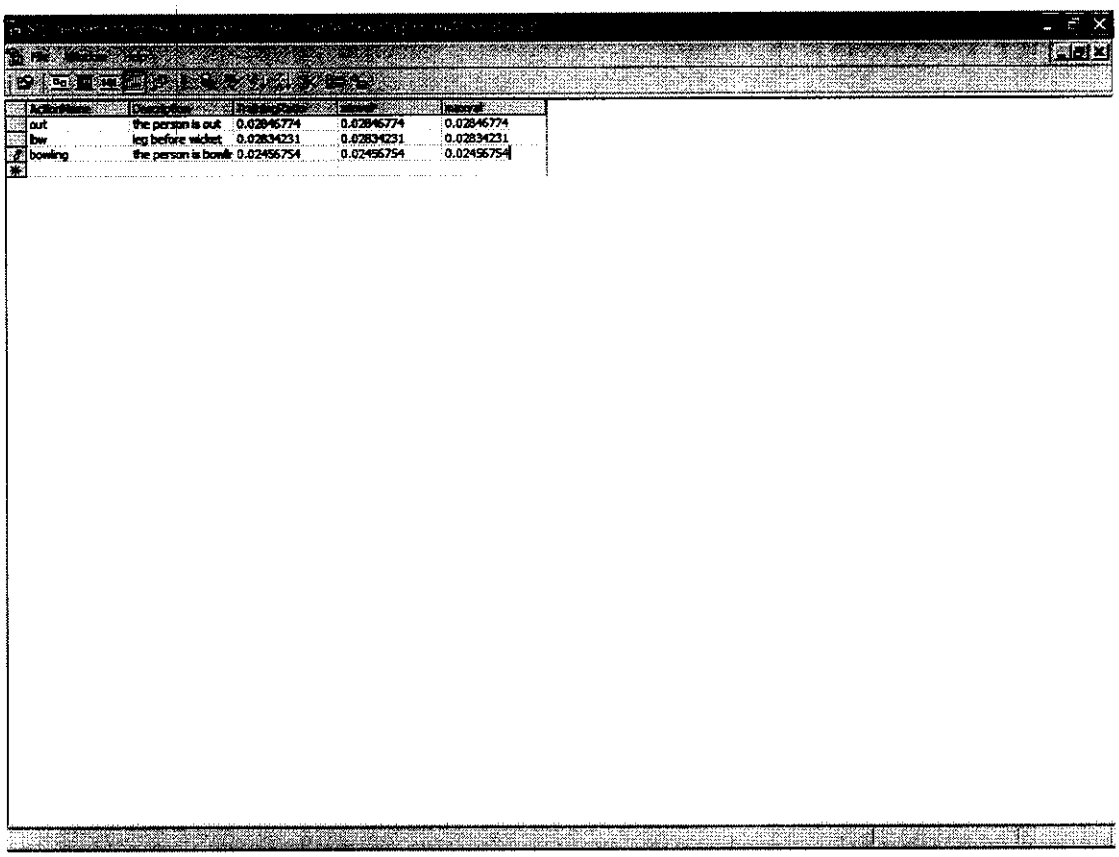

0.20001
Start


0.20001
start


0.0001


0.0001

DATABASE:



out	the person is out	0.02846774	0.02846774	0.02846774
lbw	leg before wicket	0.02834231	0.02834231	0.02834231
bowling	the person is bowle	0.02456754	0.02456754	0.02456754

REFERENCES

CHAPTER 7
REFERENCES

[1] **M. Black**, "Explaining optical flow events with parameterized spatiotemporal models," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1999, vol. 1, pp. 1326–1332.

[2] **C. Bregler**, "Learning and recognizing human dynamics in video sequences,"in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*,1997, pp. 568–574.

[3] **C. Cedras** and M. Shah, "Motion-based recognition: A survey," *Image Vis. Comput.*, vol. 13, no. 2, pp. 129–155, 1995.

[4] **A. Efros**, A. Berg, G. Mori, and J. Malik, "Recognizing action at a distance,"in *Proc. Int. Conf. Computer Vision*, 2003, vol. 2, pp. 726–733.

[5] **X. Feng and P. Perona,** "Human action recognition by sequence of movelet codewords," in *Proc. Int. Symp. 3D Data Processing Visualization and Transmission*, 2002, pp. 717–723.

[6] **D. Gavrila**, "The visual analysis of human movement: A survey," *Comput. Vis. Image Understand.*, vol. 73, no. 1, pp. 82–98, 1999.

[7] **R. Green and L. Guan**, "Quantifying and recognizing human movement patterns from monocular video images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 2, pp. 179–190, Feb. 2004.

[8] **R. Polana and R. Nelson**, "Detection and recognition of periodic, nonrigid motion," *Int. J. Comput. Vis.*, vol. 23, no. 3, pp. 261–282, 1997.

[9] **Y. Sheikh and M. Shah**, "Exploring the space of an action for human action recognition," in *Proc. Int. Conf. Computer Vision*, 2005, vol. 1, pp. 144–149.