Machine Learning and Robot Controlling

Final Report

for

Intelligent Robotics ECE 579
Portland State University, spring 2004

Normen Giesecke

1. Introduction

This Report contains 2 different parts of my project work in spring 2004. The first part is an extension of a project in Robotics 578 in winter 2004. I refer to this report to get the basic information of the machine learning part and the meaning of the OF-Table format as well as MVSIS. In this report I will explain how the "interpreting" of the data works and I will present the source that implements all required functions. The other part of the report covers, how to control the robot and program different behaviors as well as saving the programmed behaviors and movements. The structure is designed as simple as possible to have a program that even high school students can use to provide and create new movements from the servo layer upwards.

2. Machine Learning

2.1 Converter from OF format to MVSIS format

2.1.1 Overview

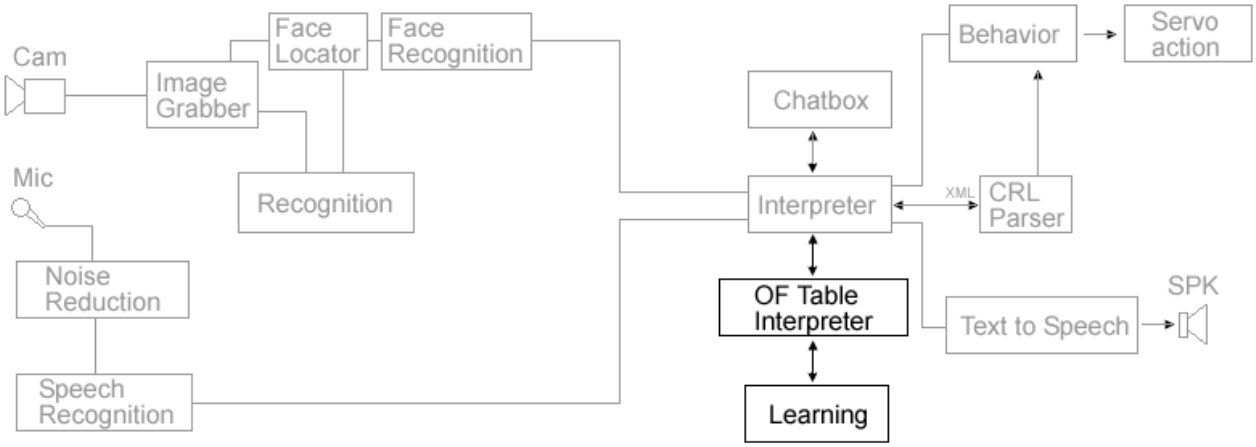

Fig. 2.1 Workflow of the robot

The figure above shows the structure and workflow of our robot. My part is the learning part and contains an “Interpreter” and the “Learning” itself. For more information I refer again to the previous report that describes the work flow more in detail. The learning part is subdivided in 2 steps. The first step is the *OF-Interpreter* and the second step is the *MVSIS decomposition program.* The *OF-Interpreter* is basically a parser that goes through the code and then recognizes particular words, saves the information and writes a new file that is in the way MVSIS can understands it. The program is a consol application since it provides only a function of the top level program, so the program is only called using threads and when the program terminates the thread is terminated. In this way resources are only used when needed. After interpreting and converting the OF-Table file, MVSIS will be started and decomposes the MVSIS file. The decomposed file goes now through the Interpreter and now the system is ready to use.

2.1.2. OF Table file structure

In order to implement learning in the robots “mind” and not only predictable behavior we invented the Object feature – Table (OF). The idea was to create a, for human readable, file so that we are able to understand the robots decisions. Other important reasons for this format are that we can control and if necessary debug the function and way of working of the robot.

Example of a *.of file

```
#-------------------------------------------------------------------------
# this is an example file as an OF Table for the age example
# filename : file.of
#-------------------------------------------------------------------------

.mvinputs a=pitch 4 low middle high very_high
.mvinputs b=height 4 short middle tall very_tall
.mvinputs c=haircolor 4 blonde brown black grey

.mvoutputs d=age 4 young middle old very_old

.table
Joan a=3 b=0 c=0 | d=0
Mike a=2 b=1 c=1 | d=1
Joe a=1 b=2 c=2 | d=2
Frank a=0 b=1 c=3 | d=3
.EOT
#End Of Table
#-------------------------------------------------------------------------
# that means
# joan has very high pitch(a=3), is short(b=0) and has blonde hair(c=0).
#-------------------------------------------------------------------------
```

This is the structure of the OF Table File and now I will show the structure of the converted file.

```
#-----------------------------------------------#
# MVSIS FILE create from a Object Feature table #
#                                               #
#-----------------------------------------------#
.model example

#definitons of inputvariables
.inputs a
.inputs b
.inputs c

#definition of outputvariable
.outputs d

#declaration of multivalued variables
.mv a,b,c 4
.mv d 4

.table a b c -> d
0 0 0 0
1 0 0 -
2 0 0 -
3 0 0 0
0 1 0 -
1 1 0 -
2 1 0 -
3 1 0 -
0 2 0 -
1 2 0 -
2 2 0 -
3 2 0 -
0 3 0 -
1 3 0 -
2 3 0 -
3 3 0 -
0 0 1 -
1 0 1 -
2 0 1 -
3 0 1 -
0 1 1 -
1 1 1 -
2 1 1 1
3 1 1 -
0 2 1 -
1 2 1 -
2 2 1 -
3 2 1 -
0 3 1 -
1 3 1 -
2 3 1 -
3 3 1 -
0 0 2 -
1 0 2 -
2 0 2 -
3 0 2 -
0 1 2 -
1 1 2 -
2 1 2 -
3 1 2 -
0 2 2 -
1 2 2 2
2 2 2 -
3 2 2 -
0 3 2 -
```

```
1 3 2 -
2 3 2 -
3 3 2 2
0 0 3 -
1 0 3 -
2 0 3 -
3 0 3 -
0 1 3 3
1 1 3 -
2 1 3 2
3 1 3 -
0 2 3 -
1 2 3 -
2 2 3 -
3 2 3 -
0 3 3 -
1 3 3 -
2 3 3 -
3 3 3 2
.end
```

As one can see in the file only the combination of the features in the OF-Table give an output in the MVSIS file. Everything else got a “don’t care”. These “don’t cares” are later subject of the decomposition in the MVSIS interpreter.

2.2 MVSIS Interpreter

After the OF file is converted a function calls MVSIS and decomposes the MVSIS file with looks like this.

```
.model example
.spec age.mv
.inputs a b c
.outputs d
.mv a 4
.mv b 4
.mv c 4
.mv d 4
.table a b c d
- 0 (0,1)  0
(0,1,3) 1 2  0
- (1,2,3) (0,1)  1
- (0,2,3) (2,3)  2
2 - (2,3)  2
(0,1,3) (0,1,2) 3  3
.end
```

First the input variables are declared (a, b, c) and then the output variable is declared (d). Afterwards the combinations of inputs and the interpreted outputs are shown. That means for example a= “don’t care”, b= 0 and c= generalized “don’t care” (0,1) gives at the current situation d=0. In this example of the age, either pitch with a low height and a hair color that “blonde” or “brown” gives us as result a “young person” which is actually true. MVSIS is called with a script file that is generated by the program in order to use the program universal. The script file looks like as follows:

Structure of the script file needed to decompose the K-Map properly.

```
read_blif_mv age.mv
strash
mfs
collapse
write_blif_mv age_guess.mv
quit
```

It is very important to mention that *"strash"*, *"mfs"* and *"collapse"* are the only combination of commands that return the required structure for the Interpreter. This structure is disjoint. To call MVSIS, the *mvsis.exe* is necessary. After MVSIS is done it is terminated with "quit" at the end of the script file.

Since the top level knows which thoughts are thought and games are played the Interpreter returns just a value and the top level architecture knows the information. In this case of the age guessing, the top level knows that 0= young, 1= middle age, 2= old and 3 means very old. The program is called with 2 parameters. The first parameter is the given .of file and the second is the name of the converted, not yet existing, MVSIS file. All data from these files are sufficient to the 2 procedures. From this point of view the program is universal applicable to different examples.

Calling the program with: "[path] of.exe age.of age.mv"

3. Robot Behavior and Controlling Tool

3.1 Intention and Structure

The Intention of the program is to provide an easy way to program the robot. The program offers to control the robot from the servo layer, over movement layer to the more complex behavior. First of all we have actual 13 servos in use. Each servo can be controlled separately in its own range. Each servo has different ranges and all these ranges are saved in the servo.ini file. The file is loaded at the beginning and gives us the capability to have always exact movements. A complex behavior consists of movements and each movement contains of several servo movements. Each movement has its own, per user adjustable, delay. The delay gives us the possibility to wait till the particular movement is done or processed by the servos, since it takes a while till each servo reaches its own position.

3.2 Description of the Program

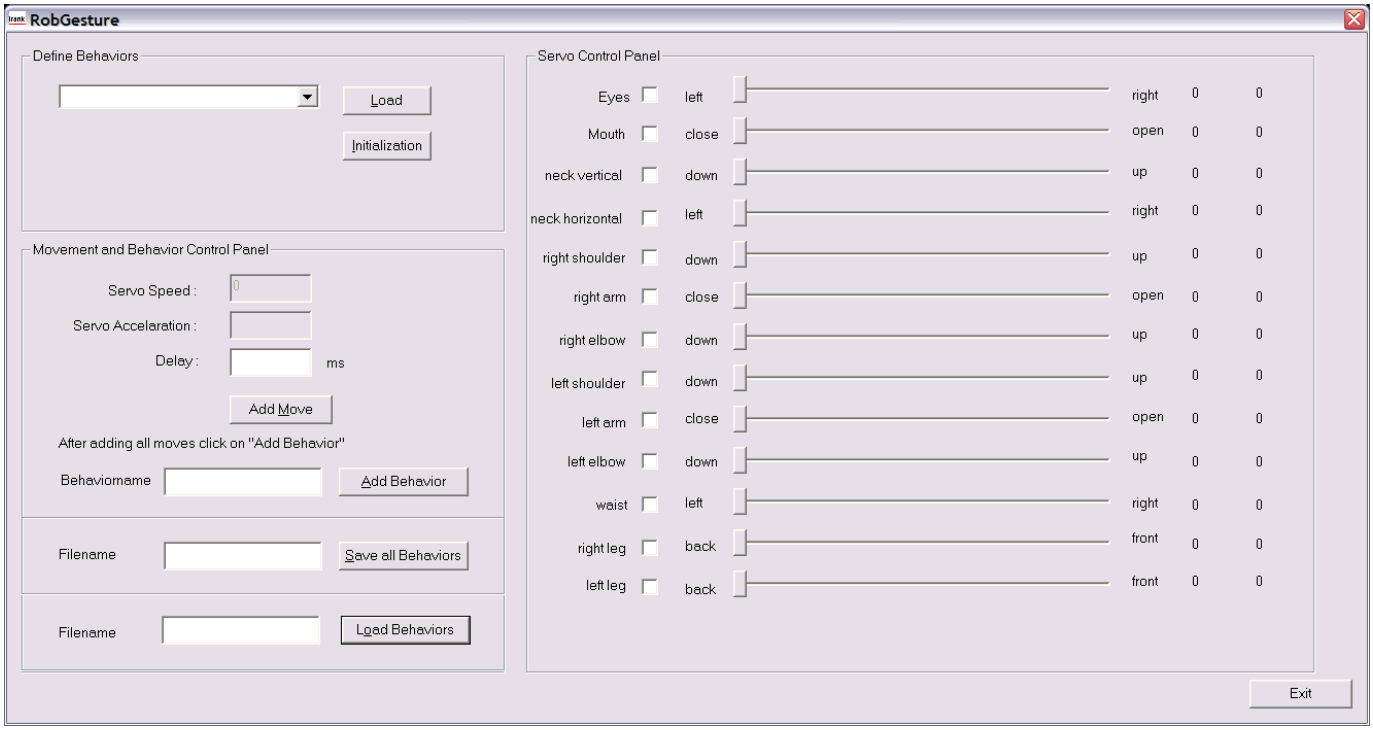

Fig 3.2. Appearance of the Robot Control Tool

Figure 3.2. shows the appearance of the Robot control tool. On the right you can see the “Servo Control Panel”. Each servo has its own slide bar. The slide bar is internal normalized from 0 to 1000. As you can see, the servo for the eyes reaches from left to right and the two number next to each slide bar is the position of the slider (e.g. half way slider is always 500) and the next number on the most right is the position of the servo. These values are different for every servo and are just for control, that the servo is in its range and works properly. In addition to that you have a checkbox on each slide to select the servo for a movement. How to program a move, will be subject of the next chapter. On the left side you see the “Define Behaviors” box in which you can select a particular movement and then load it to the servos. The “Initialization” button must be pressed at the beginning to make sure that all servos are in their initial position. The initial positions are given in the servo.ini file. The file looks like this:

```
//Servo.ini
eyes:2600 1690 780
mouth:2000 2000 3200
neck_vertical:600 2000 3300
neck_horizontal:2500 1000 -700
right_shoulder:3700 2900 -400
right_arm:3400 1720 45
right_elbow:-700 -700 2800
left_shoulder:60 650 4000
left_arm:500 1300 3800
left_elbow:3200 3200 -500
waist:1100 2320 3400
right_leg:-300 1500 3200
left_leg:3200 1700 -100
```

This file provides all servo information that is necessary. The first line shows the details for servo #1. The first number is the most left value, the second is the initialization value and the last value is the most right value for the servo.

Below the “Define Behaviors” field we have the “Movement and Behavior Control Panel”. In the first field you enter the delay for the particular movement in milliseconds. Then you have the “Add Movement” Button to simply add a movement to a behavior. When you are done with defining all movement for a behavior you enter a name for the behavior and click the “Add Behavior” Button. With the edit fields “Load Behaviors” and “Save Behaviors” you can load and save programmed behaviors.

3.3 How to program a behavior

First you start the program by clicking on “RobGesture.exe” and then click the “Initialization” Button to initialize the servos of the robot. Now you think of the movement you want to program and you can play around with the slide bars as you want to find out how the robot reacts to the changes of the slide bar. By moving the slide bar the robot will immediately react by moving its servos. If you want the initial position of all servos just hit “Initialization” again. In order to program a behavior move the slide bar of the servo you want to move. If you reach the desired position of the servo select the checkbox for this servo. You can move several servos at a time, but you have always to select the servo you want to use and to save for a particular movement. After moving and selecting the servos, enter a delay the robot should wait till the movement is done. The delay is in milliseconds and you should play around with it to figure how long it takes to move the servo in the desired position. After entering the delay you click on the “Add Movement”-Button and then you change the slide bars and check the servos as you want to program the next movement. When you are done you only have to select the servos that you are using for this particular movement. After that enter a delay and hit the “Add Movement” button again. You can repeat this as often as you want. After creating all movement, enter a behavior name in the edit box next to the “Add Behavior”- button and then click on the “Add Behavior”- button to save the behavior temporally. After hitting the button you can load the programmed behavior. Go to the “Define Behaviors” field, click on the Combo Box to select the behavior you want to see or check. After selecting the behavior hit the “Load”-button. Now the behavior is applied on the robot and the robot moves in the way you programmed him.

One can continue to create new behavior by repeating the procedure described above as often as you want. Please keep in mind to change the name of the behavior, otherwise you have two behaviors and you can not distinguish between them. After creating the behaviors you can save them by entering a file name in the edit box next to the “Save Behavior” button. After entering a name, hit the “Save Behavior” button and the programmed behaviors are saved in that file.

When you start the RobGesture.exe again and you want to continue programming new behaviors but you want to load the previous behaviors then enter the name of the previous edited file and hit the “Load Behavior” button. Afterwards all behaviors are available in the Combo Box and by simply select one and click on “Load” the behavior is applied to the robot/servos.

4. Conclusion

I finished now the learning part of the robot and made it applicable for universal use. Stefan can now use the program to implement learning. On the other hand I gave now the opportunity to control the robot in a very easy way and makes it now possible to generate behaviors for all different situations and reaction of the robot. This program might be very useful for the robot camp in summer to let people invent new behavior and help us to speed up the process of creating new behaviors.

5. Source Code of the OF_Table_to_MVSIS program

Comments are usually in English, if there are some German comments then these are inserted by the Visual Studio 6.
Due the fact that this program is called from the main application of the robot, I decided to program just a consol application.
To run the program you have to enter “OF_Table_to_MVSIS.exe file.of mvsis.mv”. Where “file.of” is the name of the “*.of file”, which is the “input file” to the program. As second parameter we have the name of the output filename which has to be a *.mv file. The extension *.mv is needed for further procession with MVSIS. There are several return values that are important if you run the program as a ‘called’ program from the main program.

```
// OF_Table_to_MVSIS.cpp : Definiert den Einsprungpunkt für die Konsolenanwendung.
//


//include files
#include "stdafx.h"
#include "string.h"
#include "process.h"
#include "windows.h"

// Overview return values
//-1 return value, if no filenames are specified.
//-2 return value, if *.of file not exists
//-3 return value, if fails to create mvsis.mv file


//global variables
char buffer[100];
FILE *of_file;
FILE *mvsis_file;
char of_filename[20]="";
char mvsis_filename[20]="";

//global input variables
char input_var_array[20][20];
char input_name_array[20][20];
char input_property_array[20][60];
char tmp[20];
int  input_var_value[20];
int  input_counter=0;
int  i=0,j=0;
int     temp=0;

//global output variables
char output_var[20];
char output_name[20];
```

```
char output_property[50];
int  output_var_value=0;

//global .table data
char names[20][20];
char table_var[20][20];
int  names_counter=0;
int  output_values[20];

//simulation input
int simul_input[20];


// help variable for the output in the blif file
int variables[100];

int read_in_OF_file()
{

	if((of_file=fopen(of_filename,"r")) == NULL)
	{
		cout << of_filename << " does not exist, please try again!" <<endl<<endl;
		return(-2);
	}



	while(!feof(of_file))
	{
		fgets(buffer,100,of_file);
		strlwr(buffer);				//convert buffer in lower case

		if(buffer[0]== '.')		//if reserved word(denoted by a point at the beginning)
		{
			//for the .mvinputs case
			if((strncmp(buffer,".mvinputs",9)==NULL))
			{
				for(i=9;i<100;i++)
				{
					if(buffer[i]!= ' ')
					{
						while(buffer[i]!='=')
						{
							input_var_array[input_counter][temp]=buffer[i];
							temp++;i++;
						}
						input_var_array[input_counter][temp]='\0';
						temp=0;

						//to jump over spaces
						while(buffer[i]==' ' && buffer[i]=='=')
						{
							i++;
						}
						//getting the name of the property
						while(buffer[i]!=' ')
						{
							input_name_array[input_counter][temp]=buffer[i];
							temp++;i++;
						}
						input_name_array[input_counter][temp]='\0';
						temp=0;

						//to jump over spaces
```

```
                                        while(buffer[i]==' ')
                                        {
                                                i++;
                                        }
                                        //getting the value of the property
                                        while(buffer[i]!=' ')
                                        {
                                                tmp[temp]=buffer[i];

                                                temp++;i++;
                                        }
                                        tmp[temp]='\0';
                                        input_var_value[input_counter] = atoi(tmp);
                                        temp=0;

                                        //now properties of the input variable

                                        while(buffer[i]==' ')
                                        {
                                                i++;
                                        }

                                        while(buffer[i]!='\0')
                                        {
                                                input_property_array[input_counter][temp]=buffer[i];

                                                temp++;i++;
                                        }

                                        input_property_array[input_counter][--temp]='\0';
                                        i=0;temp=0;

                                        break;
                                }
                        }


                        input_counter++;
        }
        //End of .mv case


        //begin of .mvoutputs case ------------------------------------------------
        if((strncmp(buffer,".mvoutputs",10)==NULL))
        {
                for(i=10;i<100;i++)
                {
                        if(buffer[i]!= ' ')
                        {
                                while(buffer[i]!='=' && buffer[i]!=' ')
                                {
                                        output_var[temp]=buffer[i];
                                        temp++;i++;
                                }
                                output_var[temp]='\0';
                                temp=0;

                                //to jump over spaces
                                while(buffer[i]==' ' || buffer[i]=='=')
                                {
                                        i++;
                                }
                                //getting the name of the property
                                while(buffer[i]!=' ')
```

```
                                        {
                                                output_name[temp]=buffer[i];
                                                temp++;i++;
                                        }
                                        output_name[temp]='\0';
                                        temp=0;

                                        //to jump over spaces
                                        while(buffer[i]==' ')
                                        {
                                                i++;
                                        }
                                        //getting the value of the property
                                        while(buffer[i]!=' ')
                                        {
                                                tmp[temp]=buffer[i];

                                                temp++;i++;
                                        }
                                        tmp[temp]='\0';
                                        output_var_value = atoi(tmp);
                                        temp=0;

                                        //now properties of the variable

                                        while(buffer[i]==' ')
                                        {
                                                i++;
                                        }

                                        while(buffer[i]!='\0')
                                        {
                                                output_property[temp]=buffer[i];

                                                temp++;i++;
                                        }

                                        output_property[--temp]='\0';
                                        i=0;temp=0;

                                        break;
                                }
                        }


}//End of mvoutputs case

//begin of the .table case
temp=0;
if((strncmp(buffer,".table",6)==NULL))
{
        fgets(buffer,100,of_file);
        while(strncmp(buffer,".EOT",4)!=NULL && strncmp(buffer,".eot",4)!=NULL)
        {
                if(buffer[0]!='#' && buffer[0]!=' ' && buffer[0]!='\0')
                {
                        for(i=0;i<100;i++)
                        {
                                //names[20][20];
                                //table_var[20][20];

                                while(buffer[i]==' ')
                                {
                                        i++;
```

```
                                        }

                                        while(buffer[i]!=' ' && buffer[i]!='=')
                                        {
                                                names[names_counter][temp]=buffer[i];

                                                temp++;i++;
                                        }
                                        names[names_counter][temp]='\0';
                                        temp=0;

                                        for(j=0;j<input_counter;j++)
                                        {
                                                while(buffer[i]==' ')
                                                        i++;

                                                while(buffer[i]!=' ' && buffer[i]!='=')
                                                {
                                                        table_var[names_counter][temp]=buffer[i];
                                                        temp++;i++;
                                                }

                                                while(buffer[i]==' ' || buffer[i]=='=')
                                                        i++;

                                                while(buffer[i]!=' ' && buffer[i]!='|')
                                                {
                                                        table_var[names_counter][temp]=buffer[i];
                                                        temp++;i++;
                                                }
                                        }
                                        while(buffer[i]==' ')
                                        {
                                                i++;
                                        }

                                        if(buffer[i]=='|')
                                        {
                                                i++;
                                                while(buffer[i]!='=')
                                                {
                                                        i++;
                                                }
                                                i++;
                                                temp=0;
                                                while(buffer[i]!=' ' && buffer[i]!='\0')
                                                {
                                                        tmp[temp]=buffer[i];

                                                        temp++;i++;
                                                }
                                                tmp[--temp]='\0';
                                                output_values[names_counter] = atoi(tmp);

                                        }

                                        break;
                                }
                        }
strcpy(buffer,"\0");
fgets(buffer,100,of_file);
names_counter++;
temp=0;
}
```

```
                                        break;
                                }
                                //end of the .table case


                        }               // End of reserved words cases
        } //parenthesis of the while-loop
        fclose(of_file);
        return 0;
}


int create_blif_file()
{

        if((mvsis_file=fopen(mvsis_filename,"w"))== NULL)
        {
                cout << "Fail to create " << mvsis_filename << ". Please try again!" <<endl<<endl;
                return(-3);
        }

        int i,j,help=0;
        char char_temp[20];
        char symbol[60];

        fputs("#-------------------------------------------------#\n",mvsis_file);
        fputs("# MVSIS FILE create from a Object Feature table #\n",mvsis_file);
        fputs("#                                                #\n",mvsis_file);
        fputs("#-------------------------------------------------#\n",mvsis_file);

        fputs(".model example\n\n",mvsis_file);

        fputs("#definitons of inputvariables\n",mvsis_file);



        for(i=0;i<input_counter;i++)
        {
                fputs(".inputs ",mvsis_file);
                fputs(input_var_array[i],mvsis_file);
                fputs("\n",mvsis_file);

        }

        /*
                sprintf(char_temp,"%d",input_var_value[i]);
                fputs(char_temp,mvsis_file);
                fputs(" ",mvsis_file);
                fputs(input_property_array[i],mvsis_file);
                fputs("\n",mvsis_file);
        }*/

        //outputvariables to file
        fputs("\n#definition of outputvariable\n",mvsis_file);
        fputs(".outputs ",mvsis_file);
        fputs(output_var,mvsis_file);
        fputs("\n\n",mvsis_file);



        //write .mv variables

        fputs("#declaration of multivalued variables\n.mv ",mvsis_file);
        for(i=0;i<input_counter;i++)
```

```
    {
            fputs(input_var_array[i],mvsis_file);
            if(i<input_counter-1)
                    fputs(",",mvsis_file);
    }
    fputs(" ",mvsis_file);
    sprintf(char_temp,"%d",input_var_value[0]);
    fputs(char_temp,mvsis_file);

    fputs("\n.mv ",mvsis_file);
    fputs(output_var,mvsis_file);
    fputs(" ",mvsis_file);
sprintf(char_temp,"%d",output_var_value);
    fputs(char_temp,mvsis_file);


    fputs("\n\n.table",mvsis_file);
    fputs(" ",mvsis_file);




    // inputs a b c -> d is written here
    for(i=0;i<input_counter;i++)
    {
            fputs(input_var_array[i],mvsis_file);
            fputs(" ",mvsis_file);
    }

    fputs("-> ",mvsis_file);
    fputs(output_var,mvsis_file);
    fputs("\n",mvsis_file);


    for(i=0;i<100;i++)
            variables[i]=0;



    for(i=0;i<(int(pow(float(input_var_value[0]),input_counter)));i++)
    {
            // ausgabe von inputvariablen like a b c
            for(j=0;j<input_counter;j++)
            {
                    sprintf(char_temp,"%d",variables[j]);
                    fputs(char_temp,mvsis_file);
                    fputs(" ",mvsis_file);
            }

            //ausgabe von outputvariablen like d
            strcpy(char_temp,"\0");
            strcpy(symbol,"\0");
            for(j=0;j<input_counter;j++)
            {
                    //creating a string like "a0b0c0"
                    if(j==0)
                    {
                            strcpy(symbol,input_var_array[j]);
                            sprintf(char_temp,"%d",variables[j]);
                            strcat(symbol,char_temp);
                    }
                    else
                    {
                            strcat(symbol,input_var_array[j]);
```

```
                                        sprintf(char_temp,"%d",variables[j]);
                                        strcat(symbol,char_temp);
                                }
                        }


                        help=0;
                        strcpy(char_temp,"\0");

                        for(j=0;j<names_counter;j++)
                        {
                                if((strcmp(symbol,table_var[j])==NULL))
                                {
                                        sprintf(char_temp,"%d",output_values[j]);
                                        fputs(char_temp,mvsis_file);
                                        help=1;
                                        break;
                                }
                        }
                        if(help==0)
                                fputs("-",mvsis_file);


                        fputs("\n",mvsis_file);

                        for(j=0;j<input_counter;j++)
                        {
                                if(j==0)
                                        variables[j]++;

                                if(variables[j]==input_var_value[0])
                                {
                                        variables[j]=0;
                                        variables[j+1]++;
                                }
                        }
        }
        fputs(".end",mvsis_file);
        fclose(mvsis_file);
        return 0;
}

/*
//------------------------------------------------------------------------
//------------------------------------------------------------------------
int interpret_mvsis_output()
{
        FILE *mv_file;

        char line_buffer[200];
        int i=0;

        if((mv_file=fopen("mvsis_output.mv","r")) == NULL)
        {
                cout << "mvsis_output.mv" << " does not exist, please try again!" <<endl<<endl;
                return(-2);
        }


        while(!feof(mv_file))
        {
                fgets(line_buffer,200,mv_file);

                if(strncmp(line_buffer,".table",6)==NULL)
```

```
		{
			fgets(line_buffer,200,mv_file);

			//check if end of decomposed MVSIS file is reached
			if(strncmp(line_buffer,".end",4)==NULL)
			{
				cout << "end of decomposed file reached." << endl;
				return 5;
			}
		}
	}
	return 0;
}


//---------------------------------------------------------------------
//---------------------------------------------------------------------
*/


int simulate()
{

	FILE *mv_file;

	char line_buffer[200];

	int i=0,j=0;
	int itmp=0;

	char tmp[2];
	char buffer[10];
	char default_value_buffer[10];
	int counter_for_correctness=0;
	int letter=0;
	char *pos;
	int default_value=-1,k;

	if((mv_file=fopen("age_guess.mv","r")) == NULL)
	{
		cout << "age_guess.mv" << " does not exist, please try again!" <<endl<<endl;
		return(-2);
	}

	cout << "Please enter " << input_counter << " values!"<< endl;
	for(i=0;i<input_counter;i++)
	{
		cout <<"Enter " << i+1 << ". value!";
		if(i==0)
			cout << " (pitch 0=low 1=middle 2=high 3=very_high)"<< endl;
		if(i==1)
			cout << " (height 0=short 1=middle 2=tall 3=very_tall)" << endl;
		if(i==2)
			cout << " (haircolor 0=blonde 1=brown 2=black 3=grey)" << endl;

		cin >> simul_input[i];
		cout << endl;
	}

	while(!feof(mv_file))
	{
		fgets(line_buffer,200,mv_file);

		if(strncmp(line_buffer,".table",6)==NULL)
```

```
{
	while(1)
	{
		fgets(line_buffer,200,mv_file);

		if(strncmp(line_buffer,".default",8)==NULL)
		{
			k=9;
			while(line_buffer[k]!=' ' && line_buffer[k]!='\n')
			{
				default_value_buffer[k-9]=line_buffer[k];
				k++;
			}
			default_value_buffer[k-9]='\0';
			default_value= atoi(default_value_buffer);
			fgets(line_buffer,200,mv_file);                    //first jump over the
.default line...later there will be a recognizing of the default value!
		}

		//check if end of decomposed MVSIS file is reached
		if(strncmp(line_buffer,".end",4)==NULL)
		{
			if(default_value==-1)
			{
				cout << "end of decomposed file reached....something is wrong in the state of
denmark..." << endl;
				return 5;
			}
			else {
				cout << "the result is: " << default_value ;
				if(default_value==0)
					cout << " You are young." << endl;
				if(default_value==1)
					cout << " You are middle age." << endl;
				if(default_value==2)
					cout << " You are old." << endl;
				if(default_value==3)
					cout << " You are very old." << endl;
				return default_value;
			}
		}

		letter=0;
		counter_for_correctness=0;
		for(i=0;i<input_counter+1;i++)                    //all inputs + output --> inputcounter+1
		{
			for(j=0;j<10;j++)
			{
				if(line_buffer[letter]!=' ')
					buffer[j]=line_buffer[letter];
				else
				{
					buffer[j]='\0';
					letter++;
					break;
				}
				letter++;
			}

			while(line_buffer[letter]==' ')
				letter++;

			if(strcmp(buffer,"-")!=NULL)
			{
```

```
                                        sprintf(tmp,"%d",simul_input[i]);

                                        pos= strstr(buffer,tmp);

                                        if(i<input_counter)
                                        {
                                                if(pos==NULL)
                                                {
                                                        //cout << "String not found!"<< endl;
                                                        break;
                                                }
                                                else
                                                {
                                                        //cout << "String found!"<< endl;
                                                        counter_for_correctness++;
                                                }

                                                pos=NULL;
                                        }
                                }
                                else
                                {
                                        //cout << "String found!"<< endl;
                                        counter_for_correctness++;
                                }

                                if(i==input_counter && counter_for_correctness == input_counter)
                                {
                                        cout << "the result is: " << buffer ;
                                        if(atoi(buffer)==0)
                                                cout << " You are young." << endl;
                                        if(atoi(buffer)==1)
                                                cout << " You are middle age." << endl;
                                        if(atoi(buffer)==2)
                                                cout << " You are old." << endl;
                                        if(atoi(buffer)==3)
                                                cout << " You are very old." << endl;

                                        cout << "\nfound in this line: " << line_buffer << endl;
                                        return atoi(buffer);
                                }
                        }
                }
        }

        //return 0;//just 0 only for testing
}


int insert_entry_in_of_file(char of_file_name[20],int simul_result)
{
        char name_of_test_person[50];
        FILE *of_file;
        char tmp[50];
        char all_lines[200][200];
        int i,line=0,k=0;


        if((of_file=fopen(of_file_name,"r")) == NULL)
        {
                cout << of_file_name << " does not exist, please try again!" <<endl;
                return(-2);
        }
```

```
	cout << "Please enter your name." << endl;
	cin >> name_of_test_person;

	while(!feof(of_file))
	{
		fgets(all_lines[line],200,of_file);
		line++;
	}
	fclose(of_file);

	//close and open file to renew it
	if((of_file=fopen(of_file_name,"w")) == NULL)
	{
		cout << of_file_name << " does not exist, please try again!" <<endl;
		return(-2);
	}


	while(strncmp(all_lines[k],".eot",4)!=NULL)
	{
		fputs(all_lines[k],of_file);
		k++;
		strlwr(all_lines[k]);
	}

	fputs(name_of_test_person,of_file);
	fputs(" ",of_file);
	for(i=0;i<input_counter;i++)
	{
		fputs(input_var_array[i],of_file);
		fputs("=",of_file);
		sprintf(tmp,"%d",simul_input[i]);
		fputs(tmp,of_file);
		fputs(" ",of_file);
	}
	fputs("| ",of_file);
	fputs(output_var,of_file);
	fputs("=",of_file);
	sprintf(tmp,"%d",simul_result);
	fputs(tmp,of_file);
	fputs("\n.eot",of_file);
	fclose(of_file);
	return 0;
}



//-------------|\--/|----/\-----||---|\--|-----------------------------------
//-------------|-\/-|---/  \----||---|-\-|-----------------------------------
//-------------|----|---|--|----||---|--\|-----------------------------------
//-------------|----|---|  |----||---|---|-----------------------------------
int main(int argc, char* argv[])
{

	//initializations


	char answer[20],answer2[20];
	int simulation_result=0;

	//if not 2 filenames given, then stop to run program
	if(argc<3)
	{
```

```
		cout << "please enter the filename of the OF-Table file" <<endl ;
		cout << "and the destination-filename of the MVSIS file" << endl << endl;
		return(-1);
	}

	strcpy(of_filename,argv[1]);
	strcpy(mvsis_filename,argv[2]);


	//read in Object Feature file (OF-File)
	read_in_OF_file();

	create_blif_file();

	system("mvsis.exe -f script");


	while(1)
	{
		cout << "do you really want to simulate?"<<endl;
		cin >> answer;

		if(strcmp(answer,"y")==NULL)
		{
			simulation_result=simulate();
			cout << "Is it result correct? ('n' for no or 'y' for yes)" << endl;
			cin >> answer2;
			if(strcmp(answer2,"y")==NULL)
				insert_entry_in_of_file(argv[1],simulation_result);
			if(strcmp(answer2,"n")==NULL)
			{
				cout << "Please enter right answer: " << endl;
				cin >>  simulation_result;
				insert_entry_in_of_file(argv[1],simulation_result);
			}

			// After adding a new guy, we need to decompose it again
			//-------------------------------------------------------
			input_counter=0;
			names_counter=0;
			read_in_OF_file();
			create_blif_file();

			system("mvsis.exe -f script");
			//-------------------------------------------------------


		}

		if(strcmp(answer,"n")==NULL)
		{
			cout << "program terminated."<< endl;
			break;
		}







	}

	return simulation_result;
}
```

6. Source Code of the Robot Control Tool

```
// RobGestureDlg.cpp : Implementierungsdatei
//

#include "stdafx.h"
#include "RobGesture.h"
#include "RobGestureDlg.h"
#include "RobotCtrl.h"                                  // for the SendByte function
#include "windows.h"
#include <time.h>
#include <conio.h>
#include "XSleep.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif



// Global Variables --------

//Structure for Servo.ini
struct servo_values
{
    int servo_num;
    int servo_pos_low;
    int servo_pos_high;
    int servo_init;
};
servo_values servo_val[16];


//Saves the number of variables
int behav_cnt=0;



struct movement
{
    int servo_array[16];          //int array with the moved servo number
    int servo_pos_array[16];      //int array with desired servo position info
    int speed;                                  //speed of movement
    int accel;                                  //accelaration of movement
```

```
    int delay;                                  //delay in ms after movement
    int servo_num;                              //saves the number of servos that are in use for a
particular movement
};


struct behavior
{
    char name[30];
    movement mov[20];           //array of max 20 movements in on behavior
    int mov_cnt;
};
//behavoirs[30];

behavior behaviors[30];

// Global Variables --- END -------




/////////////////////////////////////////////////////////////////////////////
// CRobGestureDlg Dialogfeld

CRobGestureDlg::CRobGestureDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CRobGestureDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CRobGestureDlg)
    m_nSpeed = 0;
    //}}AFX_DATA_INIT
    // Beachten Sie, dass LoadIcon unter Win32 keinen nachfolgenden DestroyIcon-Aufruf benötigt
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CRobGestureDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CRobGestureDlg)
    DDX_Control(pDX, IDC_COMBO2, m_combo);
    DDX_Control(pDX, IDC_CHECK9, m_checkbox9);
    DDX_Control(pDX, IDC_CHECK8, m_checkbox8);
    DDX_Control(pDX, IDC_CHECK7, m_checkbox7);
    DDX_Control(pDX, IDC_CHECK6, m_checkbox6);
    DDX_Control(pDX, IDC_CHECK5, m_checkbox5);
    DDX_Control(pDX, IDC_CHECK4, m_checkbox4);
```

```
    DDX_Control(pDX, IDC_CHECK3, m_checkbox3);
    DDX_Control(pDX, IDC_CHECK2, m_checkbox2);
    DDX_Control(pDX, IDC_CHECK13, m_checkbox13);
    DDX_Control(pDX, IDC_CHECK12, m_checkbox12);
    DDX_Control(pDX, IDC_CHECK11, m_checkbox11);
    DDX_Control(pDX, IDC_CHECK10, m_checkbox10);
    DDX_Control(pDX, IDC_CHECK1, m_checkbox1);
    DDX_Control(pDX, IDC_SLIDER9, m_slidectrl9);
    DDX_Control(pDX, IDC_SLIDER8, m_slidectrl8);
    DDX_Control(pDX, IDC_SLIDER7, m_slidectrl7);
    DDX_Control(pDX, IDC_SLIDER6, m_slidectrl6);
    DDX_Control(pDX, IDC_SLIDER5, m_slidectrl5);
    DDX_Control(pDX, IDC_SLIDER4, m_slidectrl4);
    DDX_Control(pDX, IDC_SLIDER3, m_slidectrl3);
    DDX_Control(pDX, IDC_SLIDER2, m_slidectrl2);
    DDX_Control(pDX, IDC_SLIDER13, m_slidectrl13);
    DDX_Control(pDX, IDC_SLIDER12, m_slidectrl12);
    DDX_Control(pDX, IDC_SLIDER11, m_slidectrl11);
    DDX_Control(pDX, IDC_SLIDER10, m_slidectrl10);
    DDX_Control(pDX, IDC_SLIDER1, m_slidectrl1);
    DDX_Text(pDX, IDC_SPEED, m_nSpeed);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CRobGestureDlg, CDialog)
    //{{AFX_MSG_MAP(CRobGestureDlg)
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_SEND, OnSend)
    ON_CBN_EDITCHANGE(IDC_COMBO2, OnEditchangeCombo2)
    ON_BN_CLICKED(IDC_INIT, OnInitialization)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER1, OnChangeSlider1)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER10, OnChangeSlider10)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER11, OnChangeSlider11)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER12, OnChangeSlider12)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER13, OnChangeSlider13)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER2, OnChangeSlider2)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER3, OnChangeSlider3)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER4, OnChangeSlider4)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER5, OnChangeSlider5)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER6, OnChangeSlider6)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER7, OnChangeSlider7)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER8, OnChangeSlider8)
    ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER9, OnChangeSlider9)
    ON_BN_CLICKED(IDC_ADDMOVE, OnAddmove)
    ON_BN_CLICKED(IDC_ADDBEHAVIOR, OnAddbehavior)
    ON_BN_CLICKED(IDC_LOAD, OnLoadBehavior)
    ON_BN_CLICKED(IDC_SAVEBEHAVIOR, OnSavebehavior)
    ON_BN_CLICKED(IDC_LOADBEHAVIORS, OnLoadbehaviors)
```

```
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CRobGestureDlg Nachrichten-Handler

BOOL CRobGestureDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Symbol für dieses Dialogfeld festlegen. Wird automatisch erledigt
    //  wenn das Hauptfenster der Anwendung kein Dialogfeld ist
    SetIcon(m_hIcon, TRUE);                    // Großes Symbol verwenden
    SetIcon(m_hIcon, FALSE);          // Kleines Symbol verwenden

    // ZU ERLEDIGEN: Hier zusätzliche Initialisierung einfügen
    m_slidectrl1.SetRange(0,1000);
    m_slidectrl2.SetRange(0,1000);
    m_slidectrl3.SetRange(0,1000);
    m_slidectrl4.SetRange(0,1000);
    m_slidectrl5.SetRange(0,1000);
    m_slidectrl6.SetRange(0,1000);
    m_slidectrl7.SetRange(0,1000);
    m_slidectrl8.SetRange(0,1000);
    m_slidectrl9.SetRange(0,1000);
    m_slidectrl10.SetRange(0,1000);
    m_slidectrl11.SetRange(0,1000);
    m_slidectrl12.SetRange(0,1000);
    m_slidectrl13.SetRange(0,1000);


    //Check if servo.ini is available
    ReadInServoValues();

    return TRUE;  // Geben Sie TRUE zurück, außer ein Steuerelement soll den Fokus erhalten
}

// Wollen Sie Ihrem Dialogfeld eine Schaltfläche "Minimieren" hinzufügen, benötigen Sie
//  den nachstehenden Code, um das Symbol zu zeichnen. Für MFC-Anwendungen, die das
//  Dokument/Ansicht-Modell verwenden, wird dies automatisch für Sie erledigt.


// Die Systemaufrufe fragen den Cursorform ab, die angezeigt werden soll, während der Benutzer
//  das zum Symbol verkleinerte Fenster mit der Maus zieht.
HCURSOR CRobGestureDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}
```

```
void CRobGestureDlg::OnSend()
{
    // This function is invoked, if the "Send" Button was pressed.

    CRobotCtrl COM1Ctrl;                    // create an object for the serial port

    short ntemp=0;
    short nInteger;                         // thats the Byte we wanna send to the ASC16 Board
    short nRemainder;
    const short divisor = 256;          // we need this to split the position into two pieces
                                                                    // look below for more information

    UpdateData(true);

    // check, if a value for speed was set, if not,
    // then set a default value

    // check, if a value for speed was set, if not,
    // then set a default value

    if (!m_nSpeed)
    {
            m_nSpeed = 100;
    }

//  ntemp = _ttoi(m_strSend);     // conversation String2Number

    /*

      Ok, lets exlain how the positioning of the servo works.
      We got the position number in the variable ntemp. It is a
      number, now.
      Since we can't handle (I mean we can't send throughout the
      com1 interface) numbers that are bigger than 255, we gotta
      cut the big number down.
      I do it just by dividing the position number (ntemp) by 256.
      The solution is stored in nInteger for the integer part and
      in nRemainder for the remainder part. Both together form a
      absolute servo position. But, we dont have to deal with it.
      It's done in the servo. We just have to send both parts.

    */

    nInteger = ntemp/divisor;
    nRemainder = ntemp%divisor;

    // this is the pattern that has to send each time to the board
```

```
    COM1Ctrl.SendByte(61);                      // we gotta ask alfred what this means
    COM1Ctrl.SendByte(m_nSpeed);            // set speed
    COM1Ctrl.SendByte(0);                       // garbage Bit
//  COM1Ctrl.SendByte(m_nServoNumber);  // we call the SendByte function
    COM1Ctrl.SendByte(nInteger);            // position component 1
    COM1Ctrl.SendByte(nRemainder);          // position component 2
//  m_strSend = "";                             // reset position string
    UpdateData(false);



}

void CRobGestureDlg::Move(int servo, int position)
{
    //variables
    int nInteger;
    int nRemainder;
    int divisor = 256;
    int ntemp;
    CRobotCtrl COM1Ctrl;                    // create an object for the serial port


    // this function takes the servonumber and the desired position and executes it.
    COM1Ctrl.SendByte(servo);               // we call the SendByte function

    ntemp = position;
    nInteger = ntemp/divisor;
    nRemainder = ntemp%divisor;

    COM1Ctrl.SendByte(nInteger);            // position component 1
    COM1Ctrl.SendByte(nRemainder);          // position component 2
}




void CRobGestureDlg::OnEditchangeCombo2()
{
    // TODO: Add your control notification handler code here

}



void CRobGestureDlg::HelpFunctionReadInServoValues(char line[200],int input_count)
{
```

```
	int i=0,j=0;
	char temp[6];

	servo_val[input_count].servo_num=input_count+1;				//saves servonumber, to
make avaiable in the hole program

	while(line[i]!=':')
		i++;
	i++;

	while(line[i]==' ')
		i++;

	while(line[i]!=' ')
	{
		temp[j]=line[i];
		j++;i++;
	}
	temp[j]='\0';j=0;
	servo_val[input_count].servo_pos_low= atoi(temp);

	while(line[i]==' ')
	i++;

	while(line[i]!=' ')
	{
		temp[j]=line[i];
		j++;i++;
	}
	temp[j]='\0';j=0;
	servo_val[input_count].servo_init= atoi(temp);
		while(line[i]==' ')
		i++;

	while(line[i]!=' ' && line[i]!='\0')
	{
		temp[j]=line[i];
		j++;i++;
	}
	temp[j]='\0';j=0;
	servo_val[input_count].servo_pos_high= atoi(temp);
	i=0;

}


void CRobGestureDlg::ReadInServoValues()
{
```

```
FILE *servo_file;
char line_buffer[200];
int i=0,j=0;
int input_cnt=0;


if((servo_file=fopen("servo.ini","r"))==NULL)
{
	AfxMessageBox("Can't open 'servo.ini' !",MB_OK);
	exit(0);
}

while(!feof(servo_file))
{

	fgets(line_buffer,199,servo_file);
	strlwr(line_buffer);

	if((strncmp(line_buffer,"eyes",4))==NULL)
		HelpFunctionReadInServoValues(line_buffer,0);

	if((strncmp(line_buffer,"mouth",5))==NULL)
		HelpFunctionReadInServoValues(line_buffer,1);

	if((strncmp(line_buffer,"neck_vertical",13))==NULL)
		HelpFunctionReadInServoValues(line_buffer,2);

	if((strncmp(line_buffer,"neck_horizontal",15))==NULL)
		HelpFunctionReadInServoValues(line_buffer,3);

	if((strncmp(line_buffer,"right_shoulder",14))==NULL)
		HelpFunctionReadInServoValues(line_buffer,4);

	if((strncmp(line_buffer,"right_arm",9))==NULL)
		HelpFunctionReadInServoValues(line_buffer,5);

	if((strncmp(line_buffer,"right_elbow",11))==NULL)
		HelpFunctionReadInServoValues(line_buffer,6);

	if((strncmp(line_buffer,"left_shoulder",13))==NULL)
		HelpFunctionReadInServoValues(line_buffer,7);

	if((strncmp(line_buffer,"left_arm",8))==NULL)
		HelpFunctionReadInServoValues(line_buffer,8);

	if((strncmp(line_buffer,"left_elbow",10))==NULL)
		HelpFunctionReadInServoValues(line_buffer,9);
```

```
		if((strncmp(line_buffer,"waist",5))==NULL)
			HelpFunctionReadInServoValues(line_buffer,10);

		if((strncmp(line_buffer,"right_leg",9))==NULL)
			HelpFunctionReadInServoValues(line_buffer,11);

		if((strncmp(line_buffer,"left_leg",8))==NULL)
			HelpFunctionReadInServoValues(line_buffer,12);

		i++;
	}

	fclose(servo_file);


}


int CRobGestureDlg::ServoToSlide(int servo_pos,int servo_num)
{
	int result;
	servo_num--;

//	(servo_pos- servo_val[servo_num].servo_pos_low)/(servo_val[servo_num].

	if(servo_val[servo_num].servo_pos_low < servo_val[servo_num].servo_pos_high)
		result = 1000*(servo_pos -
servo_val[servo_num].servo_pos_low)/(servo_val[servo_num].servo_pos_high -
servo_val[servo_num].servo_pos_low);
	else
		result = 1000*(servo_val[servo_num].servo_pos_low -
servo_pos)/(servo_val[servo_num].servo_pos_low - servo_val[servo_num].servo_pos_high);
	return result;

}


int CRobGestureDlg::SlideToServo(int slide_pos,int servo_num)
{
	int result;
	servo_num--;

//	(servo_pos- servo_val[servo_num].servo_pos_low)/(servo_val[servo_num].

	if(servo_val[servo_num].servo_pos_low < servo_val[servo_num].servo_pos_high)
		result = servo_val[servo_num].servo_pos_low+ ((servo_val[servo_num].servo_pos_high -
servo_val[servo_num].servo_pos_low) * slide_pos/1000);
```

```
    return result;

}


void CRobGestureDlg::OnInitialization()
{
    int temp=0;
    int i=0;

    for(i=0;i<13;i++)
            Move(i+1,servo_val[i].servo_init);

/*  Move(1,1690);       //eyes
    Move(2,2000);       //mouth
    Move(3,2000);       //neck vertical
    Move(4,1000);       //neck horizontal
    Move(5,3100);       //right shoulder
    Move(6,1720);       //right arm
    Move(7,-700); //right elbow
    Move(8,550);  //left shoulder
    Move(9,2400);       //left arm
    Move(10,3200);      //left elbow
    Move(11,2025);      //center waist
    Move(12,1500);      //right leg
    Move(13,1700);      //left leg
*/

    temp = ServoToSlide(servo_val[0].servo_init,1);
    m_slidectrl1.SetPos(temp);
    SetDlgItemInt(IDC_SERVO1,servo_val[0].servo_init);
    SetDlgItemInt(IDC_SLIDE1,temp);

    temp= ServoToSlide(servo_val[1].servo_init,2);
    m_slidectrl2.SetPos(temp);
    SetDlgItemInt(IDC_SERVO2,servo_val[1].servo_init);
    SetDlgItemInt(IDC_SLIDE2,temp);

    temp = ServoToSlide(servo_val[2].servo_init,3);
    m_slidectrl3.SetPos(temp);
    SetDlgItemInt(IDC_SERVO3,servo_val[2].servo_init);
    SetDlgItemInt(IDC_SLIDE3,temp);

    temp= ServoToSlide(servo_val[3].servo_init,4);
    m_slidectrl4.SetPos(temp);
    SetDlgItemInt(IDC_SERVO4,servo_val[3].servo_init);
    SetDlgItemInt(IDC_SLIDE4,temp);
```

```
    temp= ServoToSlide(servo_val[4].servo_init,5);
    m_slidectrl5.SetPos(temp);
    SetDlgItemInt(IDC_SERVO5,servo_val[4].servo_init);
    SetDlgItemInt(IDC_SLIDE5,temp);

    temp= ServoToSlide(servo_val[5].servo_init,6);
    m_slidectrl6.SetPos(temp);
    SetDlgItemInt(IDC_SERVO6,servo_val[5].servo_init);
    SetDlgItemInt(IDC_SLIDE6,temp);

    temp= ServoToSlide(servo_val[6].servo_init,7);
    m_slidectrl7.SetPos(temp);
    SetDlgItemInt(IDC_SERVO7,servo_val[6].servo_init);
    SetDlgItemInt(IDC_SLIDE7,temp);

    temp= ServoToSlide(servo_val[7].servo_init,8);
    m_slidectrl8.SetPos(temp);
    SetDlgItemInt(IDC_SERVO8,servo_val[7].servo_init);
    SetDlgItemInt(IDC_SLIDE8,temp);

    temp= ServoToSlide(servo_val[8].servo_init,9);
    m_slidectrl9.SetPos(temp);
    SetDlgItemInt(IDC_SERVO9,servo_val[8].servo_init);
    SetDlgItemInt(IDC_SLIDE9,temp);

    temp= ServoToSlide(servo_val[9].servo_init,10);
    m_slidectrl10.SetPos(temp);
    SetDlgItemInt(IDC_SERVO10,servo_val[9].servo_init);
    SetDlgItemInt(IDC_SLIDE10,temp);

    temp= ServoToSlide(servo_val[10].servo_init,11);
    m_slidectrl11.SetPos(temp);
    SetDlgItemInt(IDC_SERVO11,servo_val[10].servo_init);
    SetDlgItemInt(IDC_SLIDE11,temp);

    temp= ServoToSlide(servo_val[11].servo_init,12);
    m_slidectrl12.SetPos(temp);
    SetDlgItemInt(IDC_SERVO12,servo_val[11].servo_init);
    SetDlgItemInt(IDC_SLIDE12,temp);

    temp= ServoToSlide(servo_val[12].servo_init,13);
    m_slidectrl13.SetPos(temp);
    SetDlgItemInt(IDC_SERVO13,servo_val[12].servo_init);
    SetDlgItemInt(IDC_SLIDE13,temp);

}
```

```
void CRobGestureDlg::OnChangeSlider1(NMHDR* pNMHDR, LRESULT* pResult)
{
    //Eyes
    int slider_value=0;
    int slider_min=0,slider_max=0;
    int servo_range=0;
    int servo_min=0,servo_max=0;
    int converted_value=0;


    slider_min = m_slidectrl1.GetRangeMin();
    slider_max = m_slidectrl1.GetRangeMax();
    slider_value= m_slidectrl1.GetPos();
    servo_min = servo_val[0].servo_pos_low;                                    //these
values have to read-in at the beginning at the program
    servo_max = servo_val[0].servo_pos_high;                                   //these
values have to read-in at the beginning at the program
    servo_range = servo_max - servo_min;

    converted_value = (servo_range*slider_value/slider_max) + servo_min;

    SetDlgItemInt(IDC_SLIDE1,slider_value,true);
    SetDlgItemInt(IDC_SERVO1,converted_value,true);

    Move(1,converted_value);

    *pResult = 0;
}



void CRobGestureDlg::OnChangeSlider2(NMHDR* pNMHDR, LRESULT* pResult)
{
    //Mouth
    int slider_value=0;
    int slider_min=0,slider_max=0;
    int servo_range=0;
    int servo_min=0,servo_max=0;
    int converted_value=0;


    slider_min = m_slidectrl2.GetRangeMin();
    slider_max = m_slidectrl2.GetRangeMax();
    slider_value= m_slidectrl2.GetPos();
    servo_min = servo_val[1].servo_pos_low;                                    //these
values have to read-in at the beginning at the program
```

```
    servo_max = servo_val[1].servo_pos_high;                                    //these
values have to read-in at the beginning at the program
    servo_range = servo_max - servo_min;

    converted_value = (servo_range*slider_value/slider_max) + servo_min;

    SetDlgItemInt(IDC_SLIDE2,slider_value,true);
    SetDlgItemInt(IDC_SERVO2,converted_value,true);

    Move(2,converted_value);

    *pResult = 0;
}

void CRobGestureDlg::OnChangeSlider3(NMHDR* pNMHDR, LRESULT* pResult)
{
    //Neck vertical
    int slider_value=0;
    int slider_min=0,slider_max=0;
    int servo_range=0;
    int servo_min=0,servo_max=0;
    int converted_value=0;


    slider_min = m_slidectrl3.GetRangeMin();
    slider_max = m_slidectrl3.GetRangeMax();
    slider_value= m_slidectrl3.GetPos();
    servo_min = servo_val[2].servo_pos_low;                                     //these
values have to read-in at the beginning at the program
    servo_max = servo_val[2].servo_pos_high;                                    //these
values have to read-in at the beginning at the program
    servo_range = servo_max - servo_min;

    converted_value = (servo_range*slider_value/slider_max) + servo_min;

    SetDlgItemInt(IDC_SLIDE3,slider_value,true);
    SetDlgItemInt(IDC_SERVO3,converted_value,true);

    Move(3,converted_value);

    *pResult = 0;
}

void CRobGestureDlg::OnChangeSlider4(NMHDR* pNMHDR, LRESULT* pResult)
{
    //Neck horizontal
    int slider_value=0;
    int slider_min=0,slider_max=0;
```

```
	int servo_range=0;
	int servo_min=0,servo_max=0;
	int converted_value=0;


	slider_min = m_slidectrl4.GetRangeMin();
	slider_max = m_slidectrl4.GetRangeMax();
	slider_value= m_slidectrl4.GetPos();
	servo_min = servo_val[3].servo_pos_low;                                    //these
values have to read-in at the beginning at the program
	servo_max = servo_val[3].servo_pos_high;                                   //these
values have to read-in at the beginning at the program
	servo_range = servo_max - servo_min;

	converted_value = (servo_range*slider_value/slider_max) + servo_min;

	SetDlgItemInt(IDC_SLIDE4,slider_value,true);
	SetDlgItemInt(IDC_SERVO4,converted_value,true);

	Move(4,converted_value);

	*pResult = 0;
}

void CRobGestureDlg::OnChangeSlider5(NMHDR* pNMHDR, LRESULT* pResult)
{
	//right shoulder
	int slider_value=0;
	int slider_min=0,slider_max=0;
	int servo_range=0;
	int servo_min=0,servo_max=0;
	int converted_value=0;


	slider_min = m_slidectrl5.GetRangeMin();
	slider_max = m_slidectrl5.GetRangeMax();
	slider_value= m_slidectrl5.GetPos();
	servo_min = servo_val[4].servo_pos_low;                                    //these
values have to read-in at the beginning at the program
	servo_max = servo_val[4].servo_pos_high;                                   //these
values have to read-in at the beginning at the program
	servo_range = servo_max - servo_min;

	converted_value = (servo_range*slider_value/slider_max) + servo_min;

	SetDlgItemInt(IDC_SLIDE5,slider_value,true);
	SetDlgItemInt(IDC_SERVO5,converted_value,true);
```

```
	Move(5,converted_value);

	*pResult = 0;
}

void CRobGestureDlg::OnChangeSlider6(NMHDR* pNMHDR, LRESULT* pResult)
{
	//right arm
	int slider_value=0;
	int slider_min=0,slider_max=0;
	int servo_range=0;
	int servo_min=0,servo_max=0;
	int converted_value=0;


	slider_min = m_slidectrl6.GetRangeMin();
	slider_max = m_slidectrl6.GetRangeMax();
	slider_value= m_slidectrl6.GetPos();
	servo_min = servo_val[5].servo_pos_low;
	//these values have to read-in at the beginning at the program
	servo_max = servo_val[5].servo_pos_high;                                    //these
values have to read-in at the beginning at the program
	servo_range = servo_max - servo_min;

	converted_value = (servo_range*slider_value/slider_max) + servo_min;

	SetDlgItemInt(IDC_SLIDE6,slider_value,true);
	SetDlgItemInt(IDC_SERVO6,converted_value,true);

	Move(6,converted_value);

	*pResult = 0;
}

void CRobGestureDlg::OnChangeSlider7(NMHDR* pNMHDR, LRESULT* pResult)
{

	//right elbow
	int slider_value=0;
	int slider_min=0,slider_max=0;
	int servo_range=0;
	int servo_min=0,servo_max=0;
	int converted_value=0;


	slider_min = m_slidectrl7.GetRangeMin();
	slider_max = m_slidectrl7.GetRangeMax();
	slider_value= m_slidectrl7.GetPos();
```

```
	servo_min = servo_val[6].servo_pos_low;
	//these values have to read-in at the beginning at the program
	servo_max = servo_val[6].servo_pos_high;                                    //these
values have to read-in at the beginning at the program
	servo_range = servo_max - servo_min;

	converted_value = (servo_range*slider_value/slider_max) + servo_min;

	SetDlgItemInt(IDC_SLIDE7,slider_value,true);
	SetDlgItemInt(IDC_SERVO7,converted_value,true);

	Move(7,converted_value);

	*pResult = 0;
}

void CRobGestureDlg::OnChangeSlider8(NMHDR* pNMHDR, LRESULT* pResult)
{
	//left shoulder
	int slider_value=0;
	int slider_min=0,slider_max=0;
	int servo_range=0;
	int servo_min=0,servo_max=0;
	int converted_value=0;


	slider_min = m_slidectrl8.GetRangeMin();
	slider_max = m_slidectrl8.GetRangeMax();
	slider_value= m_slidectrl8.GetPos();
	servo_min = servo_val[7].servo_pos_low;
	//these values have to read-in at the beginning at the program
	servo_max = servo_val[7].servo_pos_high;                                    //these
values have to read-in at the beginning at the program
	servo_range = servo_max - servo_min;

	converted_value = (servo_range*slider_value/slider_max) + servo_min;

	SetDlgItemInt(IDC_SLIDE8,slider_value,true);
	SetDlgItemInt(IDC_SERVO8,converted_value,true);

	Move(8,converted_value);

	*pResult = 0;
}

void CRobGestureDlg::OnChangeSlider9(NMHDR* pNMHDR, LRESULT* pResult)
{
	//left arm
```

```
	int slider_value=0;
	int slider_min=0,slider_max=0;
	int servo_range=0;
	int servo_min=0,servo_max=0;
	int converted_value=0;


	slider_min = m_slidectrl9.GetRangeMin();
	slider_max = m_slidectrl9.GetRangeMax();
	slider_value= m_slidectrl9.GetPos();
	servo_min = servo_val[8].servo_pos_low;
	//these values have to read-in at the beginning at the program
	servo_max = servo_val[8].servo_pos_high;                                    //these
values have to read-in at the beginning at the program
	servo_range = servo_max - servo_min;

	converted_value = (servo_range*slider_value/slider_max) + servo_min;

	SetDlgItemInt(IDC_SLIDE9,slider_value,true);
	SetDlgItemInt(IDC_SERVO9,converted_value,true);

	Move(9,converted_value);

	*pResult = 0;
}

void CRobGestureDlg::OnChangeSlider10(NMHDR* pNMHDR, LRESULT* pResult)
{
	//left elbow
	int slider_value=0;
	int slider_min=0,slider_max=0;
	int servo_range=0;
	int servo_min=0,servo_max=0;
	int converted_value=0;


	slider_min = m_slidectrl10.GetRangeMin();
	slider_max = m_slidectrl10.GetRangeMax();
	slider_value= m_slidectrl10.GetPos();
	servo_min = servo_val[9].servo_pos_low;
	//these values have to read-in at the beginning at the program
	servo_max = servo_val[9].servo_pos_high;                                    //these
values have to read-in at the beginning at the program
	servo_range = servo_max - servo_min;

	converted_value = (servo_range*slider_value/slider_max) + servo_min;

	SetDlgItemInt(IDC_SLIDE10,slider_value,true);
```

```
	SetDlgItemInt(IDC_SERVO10,converted_value,true);

	Move(10,converted_value);

	*pResult = 0;
}

void CRobGestureDlg::OnChangeSlider11(NMHDR* pNMHDR, LRESULT* pResult)
{
	//left elbow
	int slider_value=0;
	int slider_min=0,slider_max=0;
	int servo_range=0;
	int servo_min=0,servo_max=0;
	int converted_value=0;


	slider_min = m_slidectrl11.GetRangeMin();
	slider_max = m_slidectrl11.GetRangeMax();
	slider_value= m_slidectrl11.GetPos();
	servo_min = servo_val[10].servo_pos_low;
	//these values have to read-in at the beginning at the program
	servo_max = servo_val[10].servo_pos_high;                                    //these
values have to read-in at the beginning at the program
	servo_range = servo_max - servo_min;

	converted_value = (servo_range*slider_value/slider_max) + servo_min;

	SetDlgItemInt(IDC_SLIDE11,slider_value,true);
	SetDlgItemInt(IDC_SERVO11,converted_value,true);

	Move(11,converted_value);

	*pResult = 0;
}

void CRobGestureDlg::OnChangeSlider12(NMHDR* pNMHDR, LRESULT* pResult)
{
	//left elbow
	int slider_value=0;
	int slider_min=0,slider_max=0;
	int servo_range=0;
	int servo_min=0,servo_max=0;
	int converted_value=0;


	slider_min = m_slidectrl12.GetRangeMin();
	slider_max = m_slidectrl12.GetRangeMax();
```

```
    slider_value= m_slidectrl12.GetPos();
    servo_min = servo_val[11].servo_pos_low;
    //these values have to read-in at the beginning at the program
    servo_max = servo_val[11].servo_pos_high;                                    //these
values have to read-in at the beginning at the program
    servo_range = servo_max - servo_min;

    converted_value = (servo_range*slider_value/slider_max) + servo_min;

    SetDlgItemInt(IDC_SLIDE12,slider_value,true);
    SetDlgItemInt(IDC_SERVO12,converted_value,true);

    Move(12,converted_value);

    *pResult = 0;
}

void CRobGestureDlg::OnChangeSlider13(NMHDR* pNMHDR, LRESULT* pResult)
{
    //left elbow
    int slider_value=0;
    int slider_min=0,slider_max=0;
    int servo_range=0;
    int servo_min=0,servo_max=0;
    int converted_value=0;


    slider_min = m_slidectrl13.GetRangeMin();
    slider_max = m_slidectrl13.GetRangeMax();
    slider_value= m_slidectrl13.GetPos();
    servo_min = servo_val[12].servo_pos_low;
    //these values have to read-in at the beginning at the program
    servo_max = servo_val[12].servo_pos_high;                                    //these
values have to read-in at the beginning at the program
    servo_range = servo_max - servo_min;

    converted_value = (servo_range*slider_value/slider_max) + servo_min;

    SetDlgItemInt(IDC_SLIDE13,slider_value,true);
    SetDlgItemInt(IDC_SERVO13,converted_value,true);

    Move(13,converted_value);

    *pResult = 0;
}

void CRobGestureDlg::OnAddmove()
{
```

```
	int i=0;
	int speed=0;
	int accel=0;
	int delay=0;
	char buffer[40]={0};
	int servo_number=0;

	if(m_checkbox1.GetCheck() == 1)
	{
		GetDlgItemText(IDC_SERVO1,buffer,10);

	behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
		behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=1;
		servo_number++;
	}

	if(m_checkbox2.GetCheck() == 1)
	{
		GetDlgItemText(IDC_SERVO2,buffer,10);

	behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
		behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=2;
		servo_number++;
	}
	if(m_checkbox3.GetCheck() == 1)
	{
		GetDlgItemText(IDC_SERVO3,buffer,10);

	behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
		behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=3;
		servo_number++;
	}
	if(m_checkbox4.GetCheck() == 1)
	{
		GetDlgItemText(IDC_SERVO4,buffer,10);

	behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
		behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=4;
		servo_number++;
	}
	if(m_checkbox5.GetCheck() == 1)
	{
		GetDlgItemText(IDC_SERVO5,buffer,10);
```

```
    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
            behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=5;
            servo_number++;
    }
    if(m_checkbox6.GetCheck() == 1)
    {
            GetDlgItemText(IDC_SERVO6,buffer,10);

    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
            behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=6;
            servo_number++;
    }
    if(m_checkbox7.GetCheck() == 1)
    {
            GetDlgItemText(IDC_SERVO7,buffer,10);

    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
            behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=7;
            servo_number++;
    }
    if(m_checkbox8.GetCheck() == 1)
    {
            GetDlgItemText(IDC_SERVO8,buffer,10);

    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
            behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=8;
            servo_number++;
    }
    if(m_checkbox9.GetCheck() == 1)
    {
            GetDlgItemText(IDC_SERVO9,buffer,10);

    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
            behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=9;
            servo_number++;
    }
    if(m_checkbox10.GetCheck() == 1)
    {
            GetDlgItemText(IDC_SERVO10,buffer,10);

    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
```

```
            behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=10;
            servo_number++;
    }
    if(m_checkbox11.GetCheck() == 1)
    {
            GetDlgItemText(IDC_SERVO11,buffer,10);

    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
            behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=11;
            servo_number++;
    }
    if(m_checkbox12.GetCheck() == 1)
    {
            GetDlgItemText(IDC_SERVO12,buffer,10);

    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
            behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=12;
            servo_number++;
    }
    if(m_checkbox13.GetCheck() == 1)
    {
            GetDlgItemText(IDC_SERVO13,buffer,10);

    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[servo_number]=atoi(buf
fer);
            behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[servo_number]=13;
            servo_number++;
    }

    GetDlgItemText(IDC_SPEED,buffer,10);
    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].speed= atoi(buffer);
    GetDlgItemText(IDC_ACCEL,buffer,10);
    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].accel= atoi(buffer);
    GetDlgItemText(IDC_DELAY,buffer,10);
    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].delay= atoi(buffer);
    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_num = servo_number;

    behaviors[behav_cnt].mov_cnt++;
}

void CRobGestureDlg::OnAddbehavior()
{

    char buffer[40]={0};

    GetDlgItemText(IDC_BEHAVIOR,buffer,30);
```

```
    strcpy (behaviors[behav_cnt].name, buffer);

    m_combo.AddString(buffer);
    behav_cnt++;


}

void CRobGestureDlg::OnLoadBehavior()
{
    char select[40];
    char temp[40];
    int i=0,j=0,k=0;


    GetDlgItemText(IDC_COMBO2,select,40);

    for(i=0;i<behav_cnt;i++)
    {
            strcpy(temp,behaviors[i].name);
            strlwr(temp);
            strlwr(select);

            if((strcmp(temp,select))==NULL)
            {
                    for(j=0;j< behaviors[i].mov_cnt; j++)                   //amount of movements
                    {

                            for(k=0; k<behaviors[i].mov[j].servo_num; k++)      //amount of servos used in
the movement
                            {

    Move(behaviors[i].mov[j].servo_array[k],behaviors[i].mov[j].servo_pos_array[k]);

                            }
                            XSleep(behaviors[i].mov[j].delay);

                    }
            }
    }
    OnInitialization();
}


void CRobGestureDlg::OnSavebehavior()
{
    FILE *save;
    char filename[40];
```

```
    char temp[20];
    int i=0,j=0,k=0;


    GetDlgItemText(IDC_EDIT1,filename,40);

    if((save=fopen(filename,"w"))==NULL)
    {
            AfxMessageBox("Can't create File !",MB_OK);
            exit(0);
    }

    for(i=0;i<behav_cnt;i++)
    {
            fputs("behavior: ",save);
            fputs(behaviors[i].name,save);
            fputs("\n",save);

            for(j=0;j< behaviors[i].mov_cnt; j++)                   //amount of movements
            {
                    fputs("move\n",save);


                    for(k=0; k<behaviors[i].mov[j].servo_num; k++)     //amount of servos used in the
movement
                    {
                            sprintf(temp,"%d",behaviors[i].mov[j].servo_array[k]);
                            fputs(temp,save);
                            fputs(" ",save);
                            sprintf(temp,"%d",behaviors[i].mov[j].servo_pos_array[k]);
                            fputs(temp,save);
                            fputs("\n",save);
                    }
                    fputs("delay ",save);
                    sprintf(temp,"%d",behaviors[i].mov[j].delay);
                    fputs(temp,save);
                    fputs("\n",save);
            }
            fputs("end_behavior\n",save);

    }

    fclose(save);
}

void CRobGestureDlg::OnLoadbehaviors()
{
    char buffer[40];
```

```
FILE *load_file;
char line_buffer[200]={0};
char temp[20]={0};
int i=0;
int j=0;
int k=0;


GetDlgItemText(IDC_EDIT2,buffer,40);

if((load_file=fopen(buffer,"r"))==NULL)
{
	AfxMessageBox("Can't open the file!",MB_OK);
	exit(0);
}

else
{


while(!feof(load_file))
{

	fgets(line_buffer,199,load_file);
	strlwr(line_buffer);

	if((strncmp(line_buffer,"behavior",8))==NULL)
	{


		while(line_buffer[i]!=':')
			i++;
		i++;
		while(line_buffer[i]==' ')
			i++;

		j=0;
		while(line_buffer[i]!=' ' && line_buffer[i]!='\0')
		{
			temp[j]=line_buffer[i];
			j++;i++;
		}
		j--;
		temp[j]='\0';

		strcpy(behaviors[behav_cnt].name, temp);
		m_combo.AddString(temp);
```

```
i=0;j=0;
fgets(line_buffer,199,load_file);
strlwr(line_buffer);

while((strncmp("end_behavior",line_buffer,12))!=NULL)
{
	if((strncmp(line_buffer,"move",4))==NULL)
	{
		fgets(line_buffer,199,load_file);
		strlwr(line_buffer);
		i=0;

		while((strncmp(line_buffer,"delay",5))!=NULL)
		{
			j=0;
			while(line_buffer[i]!=' ' && line_buffer[i]!='\0')
			{
				temp[j]=line_buffer[i];
				j++;i++;
			}
			i++;
			temp[j]='\0';
			j=0;


behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_array[k]=atoi(temp);

			while(line_buffer[i]!=' ' && line_buffer[i]!='\0')
			{
				temp[j]=line_buffer[i];
				j++;i++;
			}
			j--;
			temp[j]='\0';
			j=0;

behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_pos_array[k]=atoi(temp);

			k++;

behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].servo_num=k;


			fgets(line_buffer,199,load_file);
			strlwr(line_buffer);
			i=0;
```

```
                                        }
                                        k=0;

                                        if((strncmp(line_buffer,"delay",5))==NULL)
                                        {
                                                while(line_buffer[i]!=' ')
                                                        i++;
                                                i++;

                                                j=0;
                                                while(line_buffer[i]!=' ' && line_buffer[i]!='\0')
                                                {
                                                        temp[j]=line_buffer[i];
                                                        j++;i++;
                                                }
                                                j--;
                                                temp[j]='\0';


    behaviors[behav_cnt].mov[behaviors[behav_cnt].mov_cnt].delay= atoi(temp);

                                                fgets(line_buffer,199,load_file);
                                                strlwr(line_buffer);
                                                i=0;
                                        }

                                                behaviors[behav_cnt].mov_cnt++;
                                }
                        }

                        behav_cnt++;
                }

    }
    }

    fclose(load_file);

}
```