Using Speech Recognition for controlling a Pan-Tilt-Zoom Network Camera

Enrique Garcia
Department of Computer Science
University of Lund
Lund, Sweden
enriqueg@axis.com

Sven Grönquist
Department of Computer Science
University of Lund
Lund, Sweden

Abstract

In this paper we describe a prototype component for using speech to control a remote controllable a web camera.

We investigate some situations where speech control might be useful, and describe the results of testing the component.

Our conclusion is that speech control can be a useful complement to the traditional traditional point and click interface.

1 Introduction and Background

In this project we developed a Speech Recognition component for controlling a Pan-Tilt-Zoom Network Camera. This report describes the speech recognition background, the details of our implementation and its evaluation.

1.1 Speech Recognition

Speech Recognition is a technology that allows a computer to identify the words that a person speaks. Traditionally, the input devices to the system are microphone and/or telephone.

The history of speech recognition goes back to the 19th century when Alexander Graham Bell tried to build a machine that could recognize the human voice [1]. In 1950 Bell Laboratories could build a machine that recognized the ten digits. Later in 1954 MIT developed the first hardware for identifying vowels. In the 1970s DARPA established a program for understanding continuous speech.

In the last years, dramatic improvement in speech recognition technology has taken place. This is due to new research and better algorithms, as well as computers with more processing power.

Today several methods and technologies are involved in speech recognition. The Hidden Markov Model is one example of a widely used statistical method that is based on the idea that the speech signal can be characterized as a parametric random process. Template methods use average procedures to derive words and a local spectral distance measure to compare patterns.

Despite the dramatic improvement in technology, further research is still needed to cope with the limitations of speech recognition. Some of the most important limitations are: Speech recognition systems are easy to confuse when using a large vocabulary, they find it difficult to differentiate between short words, and a high accuracy (up to 95%) is only achieved in quiet environments. Today the best way to handle some of these limitations is to train the system and learn the user to operate it.

There are many advantages using speech recognition, besides of the obvious reason that it is a more natural way to interface numerous computer applications than using a mouse and a keyboard, speech recognition allows faster input, and offers the users great freedom of mobility; hands and eyes are free.

Users who could benefit from speech recognition are people who for some reason are unable to (or find it difficult to) use a normal keyboard or mouse (*e.g.* people with hand or eye problems). It might also be useful to professionals producing written reports, but who traditionally don't type themselves,

like doctors, psychologists and lawyers. On the other hand using speech recognition for such reports requires an extremely high accuracy.

Additionally speech recognition technology is applied in telephone networks to automate operators' services and it is found in Personal Digital Assistants (PDAs), mobile phones and other client devices where keyboard input is difficult.

During the last years speech recognition has begun to be used for Internet applications. VoiceXML [2] and Speech Application Language Tags (SALT) [3] are two standards for navigating web pages and access remote database by speech only. Companies such as Microsoft, IBM, Philips and ScanSoft have presented working solutions based on these standards.

Especially Microsoft has introduced support for speech recognition in the .NET platform and a plug-in for Internet Explorer [4] that allows users to browse SALT enhanced web pages. Microsoft has also for years offered a freely downloadable Speech SDK that includes a very good speech recognition engine, components, samples and tutorials.

1.2 Pan-Tilt-Zoom (PTZ) Network Camera

A network camera can be described as a camera and a computer combined. It is a camera connected directly to the network. Inside the camera it is a CPU, Linux OS, a web server etc. A network camera has its own IP address and built-in functions to handle network communication. [5]

Everything needed for viewing images over the network is built into the unit. An embedded web server manages web pages for displaying the video and handles different HTTP requests for controlling the camera and displaying the video via Internet/Intranet, see figure 1.

Network cameras are used in professional security systems for surveillance of sensitive areas, such as buildings, casinos, banks and shops. Video of those areas can be monitored from control rooms, at police stations and by security managers from a variety of locations.

A Pan-Tilt-Zoom Network Camera is a network camera that allows the users to rotate it horizontally and vertically (Pan, Tilt) and also to adjust its zoom level (Zoom).

An example of a PTZ Network Camera is the

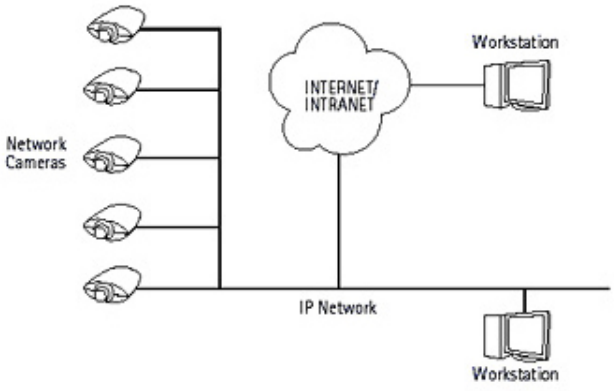

Figure 1: Network Cameras are able to transmit video and being controlled through Internet.

Figure 2: AXIS 2130 PTZ Network Camera. Here showing three different positions

AXIS 2130 (figure 2), developed by Axis Communications AB, located in Lund, Sweden. More about this PTZ Network Camera can be found at [6].

The camera is controlled by client software using `HTTP GET` requests that are sent to the camera's web server. These requests are defined in an application programming interface (API) which is independent of the kind of network camera.

For example for turning the camera with IP address `10.13.5.35`, five degrees to the left, the client software issues following request to the camera:

```
HTTP GET http://10.13.5.35/axis-cgi/
ptz.cgi?move=left
```

More information about the API and a link to the cammand details can be found in Appendix A – Axis PTZ API.

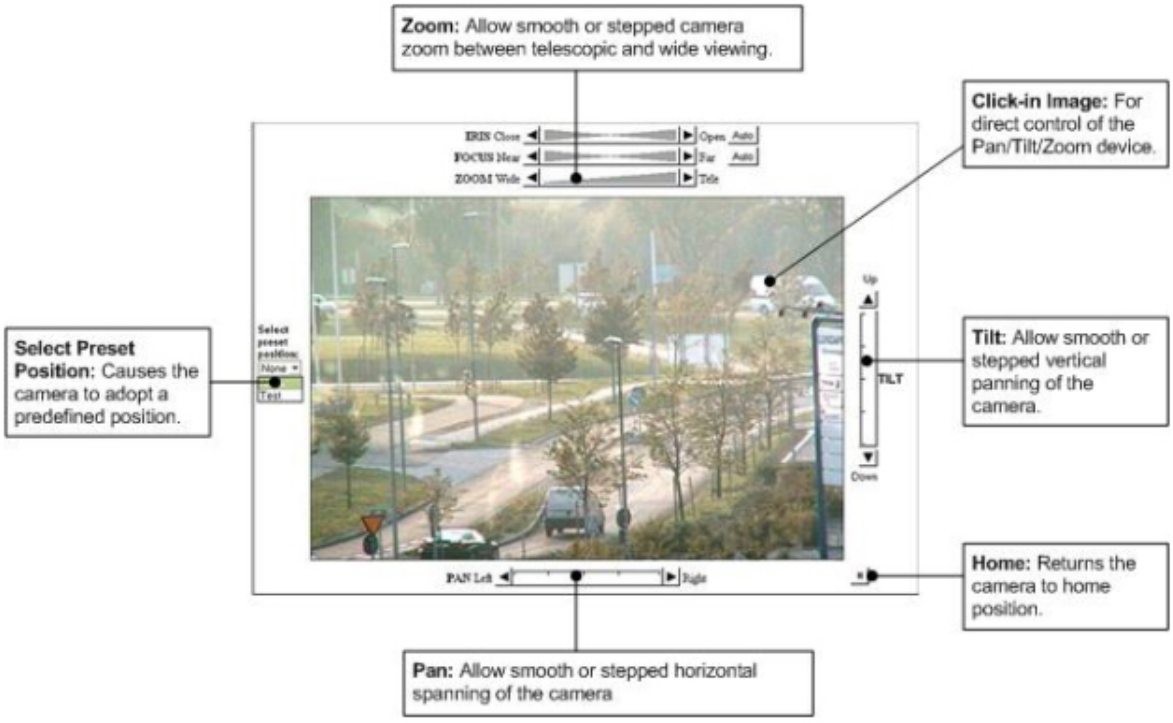

Figure 3: AXIS 2130 Client GUI

The AXIS 2130 client's graphic user interface (GUI) for controlling the PTZ camera is based on mouse control and shown in figure 3.

The users click the directions buttons with the mouse and these clicks are translated to the corresponding requests using the PTZ API.

2 Problem Statement

The project has as objective to implement and evaluate a speech recognition component for controlling the AXIS 2130 PTZ Network Camera.

Using speech recognition, the component will translate voice commands to PTZ requests managing the camera's positioning and zoom level.

We would like to test and compare this approach with the standard PTZ GUI, driven by mouse clicking in a gradient bar and directions buttons and see if the speech driven user interface could offer any advantages like natural user interaction, lower learning factor, commodity and quicker reactions of the users controlling the camera.

We believe that the results could motivate further development of new products that combine speech recognition and network camera technologies for enhancing the user experience.

3 Material and Methods

3.1 Defining Use Cases

We have identified the following scenarios. Here we don't look at a particular camera's capabilities, but rather what one might want to achieve.

General commands

The most obvious set of basic commands that we need to implement are the simple movement commands like '`up`', '`down`', '`left`', '`right`', '`undo`' etc. More complicated commands could be for moving at a certain speed (in pan and tilt directions), and even to 'lock on' a specific (moving) object. The more specific scenarios can then use these as primitives.

Another set of commands could be for handling preset positions such as '`record position N`' and '`go to position N`'. Ideally the they should be named (gate, coffee machine, etc), but we could use numbered presets.

For debugging purposes we might want additional commands such as '`reload grammar`' or '`start`

`logging`'.

Surveillance

One operator uses one or more cameras, and should be able to control any one camera at a given time. Here we have the following problems:
– More than one camera: We must give each a unique name, or, probably easier, a number. The operator shouldn't need to address a camera with its IP number. Using names is preferable, but names must not conflict with the commands (i.e. cameras should not be named '*left*' and '*right*').
– Only one camera can be operated at a given time. We need commands like '`activate camera N`' and perhaps '`deactivate camera N`' (but activating one camera should deactivate the others). These cammands could be mouse/keyboard-operated.
– We should consider the difference between a system where a camera only has a number of fixed presets (show gate, show machine A, show machine B etc), and a more *ad hoc* system where the operator can do anything.
– Sequences of operations, letting the user (operator?, manager?, *cf* web camera below) define a number of 'targets' (gate, machine A etc) in terms of pan-tilt-zoom, and letting the camera 'visit' each target. In this way one camera can work in a semi-autonomous mode, while the operator works with other things. This might get quite complex, the operator should be able to stop the camera, look at something particular, and then letting it resume.
– We must have a useful and consistent interface for working when speech control is not possible (e.g. when the operator is using the telephone). Sometimes it might be better *not* to use speech control.

Also: For a system like this it is acceptable to train the system for the operators voice (altough our solution doesn't do that).

Exhibitions

Additional issues for using the system at an exhibition:
– Working in a noisy environment (specifically lots of talk around),
– handling unknown voices (again, we don't use a training system, so this is not a problem here).

Web camera (or Web Attraction)

Controlling a web camera raises two new problems:
– Setting up more than one role. The first role is the *owner*, the person who sets the various targets (like '`coffee machine`', '`window view`' etc). The other is the *user* who should be able to say '`show coffee machine`' etc, but not much more.
– handling more than one user. When more than one user tries to control the camera at the same time ther might be synchronizing problems.
We don't investigate this any further.

3.2 Using the Microsoft Speech SDK

In this project we have chosen to use the Microsoft Speech Software Development Kit (SDK) because of the simplicity of the application programming interface (API) and the good support that can be found in this free package.

We avoided using the most recent Microsoft .NET SDK, because of the required upgrades in many components and other unavailable tools. (such as Microsoft Visual Studio .NET 2003).

The Speech SDK v 5.1 comes with COM, Active-X, VB and VC++ interfaces for speech recognition and speech synthesis for Windows. It also includes freely distributable text-to-speech engine (TTS) and speech recognition engine for U.S. English.

Using the Speech SDK is straightforward, a number of tutorials explain how to initialize the speech recognition engine and how to define the grammars for using it.

More information about the Microsoft Speech SDK is available at [6].

4 Prototype

4.1 A Speech Recognition Active-X Control

We developed an Active-X control which can recognize the general commands for controlling the camera. Building the prototype as an Active-X control makes it possible to embedded it on a web page below the Axis client that displays the video.

The GUI has been kept simple showing the available commands. When a command is identified that command's text color changes to red (see figure 4).

We wrote a simple grammar defining the following commands:

```
[Show me|Go to] (the) [left|down|right|up|home]
```

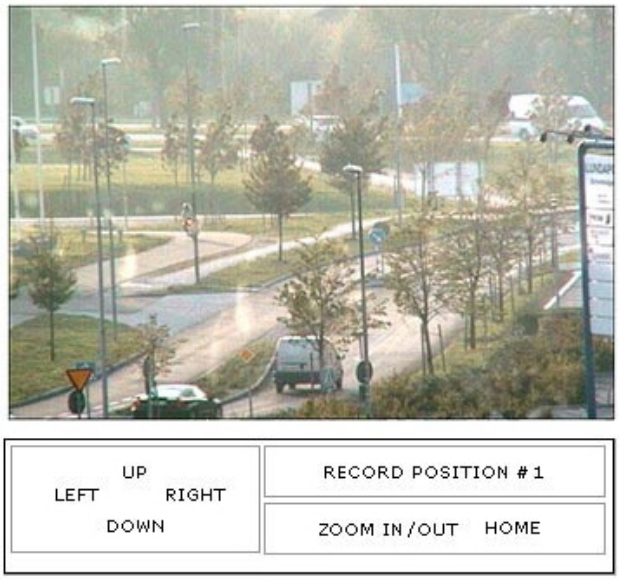

Figure 4: The prototype GUI

```
[Zoom] [in|out]
[Record|Show me] [one|two|three]
```

The last one is for recording and displaying the preset positions. In that way the prototype can handle the use cases for simple commands and the surveillance case for predefined presets.

5 Results and Discussion

We have tested the prototype with three subjects. We had prepared the following tests:

1. Rotate the camera so that a predefined object will be displayed in the center of the video (pan and tilt operations were necessary to center it)
2. Follow the trajectory of an object (a person walking).
3. Zoom in to a distant object so that it covers the entire display.
4. Record and view three different preset positions.

The tests raise a number of difficulties: using simple commands are not enough for exactly positioning the camera, it was also necessary in some cases to repeat the command many times and did show that others words like '`more`', '`little more`', '`stop`', '`back`' and '`again`' should also be handled like commands.

Any time that a subject used for example he word '`left`' the camera turned on the left exactly five degrees, sometime much more or much less than necessary. Positioning exactly the camera without specifying the degrees seems difficult as to imagine how many degrees are necessary. The GUI based in mouse click gives a scale that it is more accurate and clicking somewhere in the picture for centering the camera seems simpler than using the speech version.

A collected interaction in the test 1 follows:

```
- turn on left
- more
- more to the left
- more
- more
- more to left
- stop
- turn little to the right
- right
```

'`Stop`' is not supported neither by the camera or the prototype. '`Little`' was not supported so the user found it more difficult to complete the task.

The most successful test was about recording and viewing the presets. Users found it useful, and easy to use. They state that it should be possible to name each preset by speech and then call them instead of, as now, using numbers.

During the tests the accuracy was very good and the engine proved to be good enough for commands.

6 Conclusions

The exactly positioning of the camera seems difficult using speech, as well as following moving objects. Our simple commands don't allow accurate positioning for the pan-tilt-zoom values.

Our prototype seemed more useful when using presets positions.

6.1 Future directions

Furter development of the prototype should investigate other approaches like for example using scales on the side of the video. We think the accuracy will increase but it will never be as simple as using the mouse. Probably speech control is most suited to situations where the users are unable to use the mouse or the keyboard (or they are used for other purposes).

Future version should allow renaming the position and maybe interacting with the user in a dialogue about the position presets available.

Other camera client functions that can be easy mapped to commands like '`Take a shot`', '`Show Full Screen`' and more could be added.

Networks Cameras can also be accessed using a PDA, that field should also be investigated at least with preset positions in mind.

7 Acknowledgements

We would like to thank Pierre Nugues and Mathias Haage at the Department of Computer Science at Lund University for valuable insights and comments during our work.

References

1. History of Speech Recognition
   `http://nexus.carleton.ca/~kekoura/history.html`
2. VoiceXML 2.0 Last Call
   `http://www.w3.org/TR/voicexml20/`
3. SALT Forum
   `http://saltforum.org/`
4. MS Speech SDK
   `http://microsoft.com/speech/`
5. What is a Network Camera?
   `http://www.axis.com/products/video/video_clip/axiscam_448x336_mpeg1_high.mpg`
6. AXIS 2130 PTZ Camera
   `http://www.axis.com/products/cam_2130/index.htm`

Appendix A – Axis PTZ API

The complete Axis Camera API, HTTP - Interface Specification v 1.11 can be found at: `http://www.axis.com/techsup/cam_servers/dev/cam_http_api.htm`. Only the Pan/Tilt/Zoom cameras support the PTZ commands.

To control the Pan, Tilt, and Zoom behavior of a PTZ unit, the following PTZ control URL is used.
Method: GET/POST
Syntax: `http://<servername>/axis-cgi/com/ptz.cgi?<parameter>=<value>[&<parameter>=<value>...]`
with `parameters` and `values` from the Interface Specification.

Appendix B – PTZ Grammar

```
<GRAMMAR LANGID="409">
  <DEFINE>
    <ID NAME="VID_Left" VAL="10"/>
    <ID NAME="VID_Right" VAL="20"/>
    <ID NAME="VID_Up" VAL="30"/>
    <ID NAME="VID_Down" VAL="40"/>
    <ID NAME="VID_Home" VAL="50"/>

    <ID NAME="VID_One" VAL="60"/>
    <ID NAME="VID_Two" VAL="70"/>
    <ID NAME="VID_Three" VAL="80"/>

    <ID NAME="VID_Place" VAL="90"/>
    <ID NAME="VID_Navigation" VAL="100"/>

    <ID NAME="VID_Position" VAL="110"/>
    <ID NAME="VID_Record" VAL="120"/>
    <ID NAME="VID_Number" VAL="130"/>

    <ID NAME="VID_Zoom" VAL="140"/>
    <ID NAME="VID_Direction" VAL="150"/>
    <ID NAME="VID_In" VAL="160"/>
    <ID NAME="VID_Out" VAL="170"/>
  </DEFINE>

  <RULE ID="VID_Navigation" TOPLEVEL="ACTIVE">
    <O>Please</O>
    <P>
      <L>
        <P>Show me</P>
        <P>Go to</P>
      </L>
    </P>
    <O>the</O>
    <RULEREF REFID="VID_Place" />
  </RULE>

  <RULE ID="VID_Place" >
    <L PROPID="VID_Place">
      <P VAL="VID_Up">up</P>
      <P VAL="VID_Right">right</P>
      <P VAL="VID_Left">left</P>
      <P VAL="VID_Down">down</P>
      <P VAL="VID_Home">home</P>
    </L>
  </RULE>

  <RULE ID="VID_Record" TOPLEVEL="ACTIVE">
    <O>Please</O>
    <P>
      <L>
        <P>Record</P>
      </L>
    </p>
    <RULEREF REFID="VID_Number" />
  </RULE>

  <RULE ID="VID_Position" TOPLEVEL="ACTIVE">
    <O>Please</O>
    <P>
      <L>
        <P>Show me</P>
      </L>
```

```
    </p>
    <RULEREF REFID="VID_Number" />
  </RULE>

  <RULE ID="VID_Number" >
    <L PROPID="VID_Number">
      <P VAL="VID_One">one</P>
      <P VAL="VID_Two">two</P>
      <P VAL="VID_Three">three</P>
    </L>
  </RULE>

  <RULE ID="VID_Zoom" TOPLEVEL="ACTIVE">
    <O>Please</O>
    <P>
      <L>
        <P>Zoom</P>
      </L>
    </p>
    <RULEREF REFID="VID_Direction" />
  </RULE>

  <RULE ID="VID_Direction" >
    <L PROPID="VID_Direction">
      <P VAL="VID_In">in</P>
      <P VAL="VID_Out">out</P>
    </L>
  </RULE>
</GRAMMAR>
```