interspeech 2008 incorporating SST 08

A Composite Framework for Affective Sensing

Gordon McIntyre, Roland Goecke

Research School of Information Sciences and Engineering,
Australian National University, Canberra, Australia
gordon.mcintyre@anu.edu.au, roland.goecke@ieee.org

Abstract

A system capable of interpreting affect from a speaking face must recognise and fuse signals from multiple cues. Building such a system requires the integration of software components to perform tasks such as image registration, video segmentation, speech recognition and classification. Such software components tend to be idiosyncratic, purpose-built, and driven by scripts and textual configuration files. Integrating components to achieve the necessary degree of flexibility to perform full multimodal affective recognition is challenging. We discuss the key requirements and describe a system to perform multimodal affect sensing which integrates such software components and meets these requirements.

Index Terms:emotion recognition, affective sensing

1. Introduction

The term “composite framework” is used to denote a structure made up of discrete components while at the same time connoting a sense of heterogeneity and challenge in the integration of the individual entities. Indeed, one of the biggest hurdles in building a system to perform multimodal sensing is the integration of multiple idiosyncratic software components, purpose-built and typically driven by scripts and configuration files.

If one contemplates the operation of a full-lifecycle system that can be trained from audio and video samples and then used to perform multimodal affect recognition, the requirements are extensive and diverse. For example, to detect emotion in the voice the system must be capable of training, say, Hidden Markov Models (HMM) from prosody in the speech signals. Another requirement might be that a Support Vector Machine (SVM) is to be trained to recognise still image facial expressions, e.g. fear, anger, happiness, sadness, disgust, surprise, neutral. More complex, is the requirement to grab a sequence of video frames and, from the sequence, recognise temporal expressions. In order to perform the latter, it might be necessary to use a deformable model, e.g. an Active Appearance Model (AAM) [2] to fit to each image and provide parameters that can then, in turn, be trained using some classifier - possibly another HMM. Other features might also be considered for input to the system, e.g. eye gaze and blink rate. Ultimately, some strategy is required to assess the overall meaning of the signals, whether that involves fusion using a combined HMM or some other technique.

From the concise consideration of the requirements it can be seen that a broad range of expertise and software is needed. It is not practical to develop the software from first principles. Software capable of the recognition of voice and facial expressions implement techniques from different areas of specialisation. Automatic speech recognition (ASR) techniques have evolved over several decades while computer vision has become practical in the last ten years with the evolution of statistical techniques and computer processing power.

Here we discuss the functional requirements of a system capable of sensing multiple variable inputs from voice, facial expression and movement, making some assessment of the signals of each, and then fusing them to provide some degree of affect recognition. Due to space constraints, the reasons for choosing one software product over another is not within the scope of this paper. We present a brief overview of some of the critical components and the ”composite framework” used to harness them.

Section 2 discusses the functional system requirements followed by a summary of the operational requirements in section 3. Section 4 describes some of the key features of the system that we believe makes it significant. Section 5 highlights the progress to date and, finally, future directions are discussed in section 6.

2. Functional requirements of a system to perform affective sensing

There are several levels of sophistication that a system capable of sensing affect could provide:

1. recognition of affect from audio only
2. recognition of affect from image only
3. recognition of affect from video without audio
4. recognition of affect from video with audio

Indeed, the ultimate is to be able to recognise emotion from both audio and video inputs from a speaking face. Consider a speaking face in a real world situation. Voice expression is not necessarily continuous. There may be long pauses or periods of sustained speech. Vocal speech and facial expression may not necessarily be contemporaneous. The verisimilitude of the voiced expression might be confirmed or contradicted by the facial expressions. The face might be expressionless, hidden, not available or occluded to some extent for certain periods of time. This implies that the system needs to be able to:

1. detect the voice and facial expressions independently;
2. operate on only one modality in some cases; and
3. weigh one signal against the other

Lastly, the system must be flexible so that alternative software products and techniques can be substituted without a large amount of effort involved. For example, to compare classification performance, it might be desirable to substitute an Artificial Neural Network (ANN) package with an SVM package or simply evaluate different types of SVMs.

The following sections present a minimalist requirement statement of some of the key functional areas.

Accepted after peer review of full paper

2.1. Audio processing

The most common approach to recognising affect in speech is to use existing ASR software and to try to detect prosody from the energy levels and variance in the the signals. There are several ASR packages freely available, some with better support than others. Whatever the choice, this capability is mandatory.

2.2. Image processing

A major issue in recognising facial expressions from video is boundary detection, i.e. the boundary between the onset and offset of each expression. One approach to this is to try to recognise either the onset or the apex of a facial expression. For example, in a simple situation a subject might begin with a neutral expression and then progress to an expression of surprise. [10] have attempted expressions from peak to peak. Whatever the temporal position in the expression, this implies training a classifier of still images and then grabbing one or more video frames from a video sequence and attempting to match the expression.

Before we can recognise a face in an image it is necessary to have a software capability to first detect the face. Once we have found the face, it is necessary to fit some type of model, e.g. an AAM to the face. This will yield parameters that can be used by the classifier.

2.3. Video processing

As mentioned previously, there needs to be some capability to grab frames from the video at certain intervals. This function will be performed both manually and automatically. The frames need to be grabbed for training and recognition phases.

2.4. Classification

It is essential that the system be capable of incorporating different types of classifiers for individual inputs and possibly an ensemble of classifiers for fusing the individual classifications and weighting them.

2.5. Miscellaneous

In order to compare different techniques it must be possible to re-run training and testing phases, i.e. persisting all the available inputs, interim and final results. It is also desirable to be able to compare studies or projects and store the results separately for later comparison.

The system must be capable of running in batch or online mode. Likely this will be batch mode for training and online mode for the recognition phase.

3. Operational requirements of a system to perform affective sensing

3.1. Implementation platforms

Ideally the system should have broad platform support. In practical terms, this translates to variants unix and linux, Mac OS and Windows.

3.2. Audio and Video formats

One of the first hurdles that one encounters, especially with video processing, is the number of different video container formats and their lack of availability on one or more operating environments. Where possible the system should be capable of supporting multiple audio and video container formats. At a minimum these should include WAV, AVI, MP4, MPEG2, and MOV. If support is not available then there should be some simple way of converting between formats where this is possible.

3.3. Image processing

The system needs the capability to train a classifier on a corpus of still emotional expressions. The corpus could be images of jpeg, png or some other format. Alternatively, there may be no image corpus, rather, a video collection that will require significant images to be grabbed into a suitable format, thus creating a de facto corpus. The images will then be subjected to some recognition process.

3.4. Video processing

A video can be hours in duration or it could simply be, as in the Cohn-Kanade database [4], collections of short, sample expressions. Both in training and testing, the system needs to be able to grab frames from a video segment. The frames will then be subjected to a treatment similar to the images processing mentioned previously, and the resulting parameters input to, say, a HMM.

3.5. Classification

Several subsystems require some form of classification component. It is essential that the system be able to perform some data reduction of any input vectors that are large in dimensionality, e.g. principal component analysis (PCA) or linear discriminant analysis (LDA).

3.6. System performance

The system must make use of multi-threading and multi-processing capabilities of the operating system. Performance is critical, as is efficient memory usage. It is preferable that the system be able to execute in online mode.

3.7. User interface

The core system must be simple to use so that the effort required to integrate components and to re-run exercises is minimised.

4. System description

4.1. Software selection for the core system

In order to meet the cross-platform operating environment requirement, C++ or Java was considered suitable for development of the core system. However, given the critical performance requirements and the fact that most high-performing video processing libraries are C or C++ based, C++ has been selected. Trolltech's Qt product [9] was selected for development of the core system and user interface. Qt has another very attractive feature - its MetaObject Pattern - supports reflection which is a feature found in many object oriented (OO) languages (natively in Java) but not natively in C++. Qt's metaobject processor can generate code to support reflection.

The benefits of reflection are realised when it comes to saving and restoring exercises. The state and values of objects and their properties are "reflected" fairly simply into, in our case, extensible markup language (XML). Making use of this metaobject, we can do "round-trip" xml - serialising our objects, persisting them, and then deserialising them. In practical terms, this is the means to saving and restoring projects.

4.2. Software selection for major functions

We have decided on use of the Vision-something-Libraries (VXL) for image processing [12]. The libraries are written in C++ and are very efficient. OpenCV is used for simple video display and to grab images from videos [7]. SVMLIB [6] will be used for facial expression recognition, Sphinx [11] will be evaluated for use in the recognition of prosodic features in speech.

4.3. Class structure

The system has been built for using design patterns as described by [5]. Qt lends itself to building C++ in accordance with design patterns [3]. Figure 1 depicts the conceptual class structure. We also make use of serialiser and composite patterns to effect the round-trip processing, mentioned above.

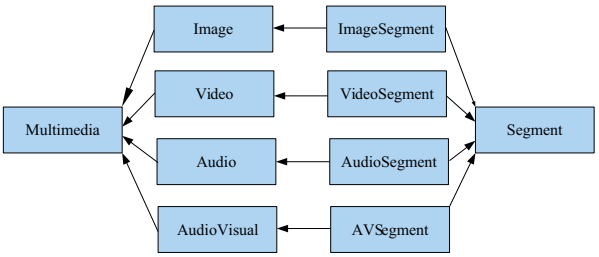

Figure 1: *Class hierarchy.*

Factories are used to produce the various objects that go to make up an project. A "project", which is our top level concept, is simply a collection of segments (discussed in the next section) and is created by a software factory. Facades are used to abstract the details of external classes such as those used to perform AAM processing. A form factory is used to create simple dialog boxes for user input, reducing the amount of effort that would have otherwise have been required if the dialogs had been hand-crafted.

4.4. Segments

Central to the design of the system is the concept of segments. This borrows to some extent from, but is simpler than, MPEG-7 [1] and its concept of segment types. This is no coincidence. MPEG-7, previously known as "Multimedia Content Description Interface", is a standard for describing the multimedia content data that supports some degree of interpretation of the information meaning, which can be passed onto, or accessed by, a device or a computer code. However, our implementation deviates slightly in that segment and multimedia data members and operations are combined.

The various types of segments are created by a segment factory. As was seen in figure 1, these include Image Segments, Image Sequence Segments, Image Collections, Audio Collections and Video Collections. Using factories to provide a layer of abstraction not only conceals the implementation complexity from the calling functions but simplifies the creation of new types of segments. Figure 2 depicts the segment factory class diagram.

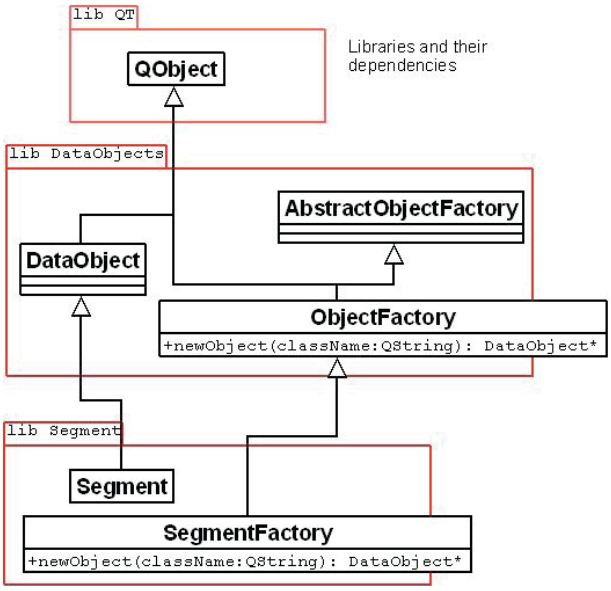

Figure 2: *Class diagram of the Segment Factory.*

4.5. User interface

4.5.1. Dialog creation

User dialogs are, in keeping with the design pattern approach, created by factories. Figure 3 demonstrates the simplicity in creating new dialogs through the use of a form factory [3].

```
bool ImageSegment::askQuestions() {
    FormFactory ff;
    *this << ff.newQuestion("name", "Image name", "ISEG000001");
    *this << ff.newQuestion("parent", "Parent Image Sequence Segment number", "");
    // new segment y or n
    m_fv = new FormDialog(this);
    //qDebug() << "Setting parent";
    m_fv->setWindowTitle("Image sequence details.");
    m_fv->exec();
```

Figure 3: *Dialog creation.*

4.5.2. Processing scenario

Rather than simply list each function, this is better described by a practical walk through a typical processing scenario. Most major functions are accessible by right-clicking to present a context menu as shown in figure 4.

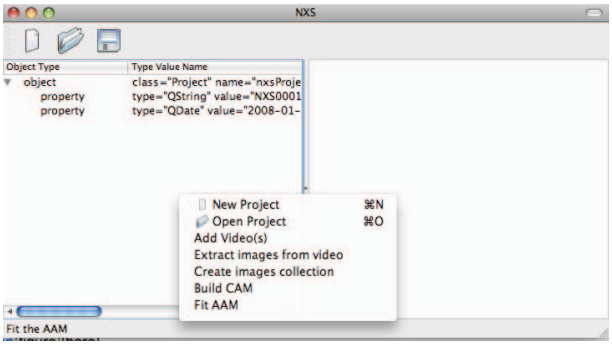

Figure 4: *The system menu.*

The starting point is the creation of a "Project" as seen in figure 5.

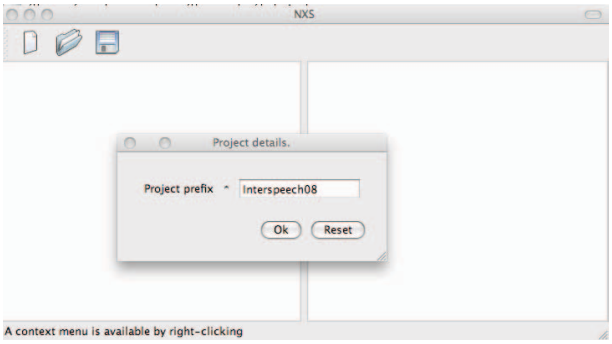

Figure 5: *Project creation.*

Figure 6 shows the project tree structure after a project has been created, an image segment, image collection segment, video segment, and model segment have been added to the project. The tree structure is effectively the xml that reflects the objects' states and data member values. From here the xml can be saved and reopened later.

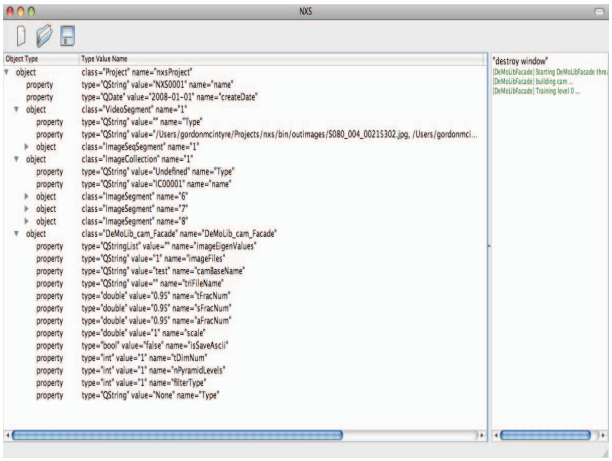

Figure 6: *User Interface.*

5. Progress to date

We have created a framework that, without much effort, allows us to substitute different software components. It also allows us to easily set up new exercises, or "projects" as we call them, saving and restoring them. In summary, the benefit of the framework is that it vastly reduces the amount of time spent in:

1. adding or substituting software components or functions
2. setting up and running recognition experiments
3. re-running experiments with different parameters and software components

To date the system is capable of defining projects, and to those projects, collections of images, sequences of images, audio and video samples. One can train an AAM from the image collections and fit the AAM to previously unseen images. The shape and texture parameters from fitting the AAM can be used to train a classifier. Sequences of images can be grabbed from videos. Within the next three months we will add functionality to train HMMs from the image sequences.

6. Future work

More software components will be added. Eye tracking and blink rate are not yet implemented. However, with the flexibility of the framework adding these with a wrapper class will be a fairly simple task. It is possible that the authors will make the software available to promote and improve research into the field of affective computing [8]

7. Acknowledgements

The authors would like to thank Jason Saragih and appreciate the use of the DeMoLib AAM processing software library.

8. References

[1] O. Avaro and P. Salembier. Mpeg-7 systems: overview. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):760–764, Jun 2001.

[2] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active Appearance Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[3] A. Ezust and P. Ezust. *An Introduction to Design Patterns in C++ with Qt 4 (Bruce Perens Open Source)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[4] Facial Expression Database, http://vasc.ri.cmu.edu/idb/html/face/facial_expression/index.html, last accessed 8 May 2008.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.

[6] LIBSVM – A Library for Support Vector Machines, http://www.csie.ntu.edu.tw/ cjlin/libsvm/, last accessed 8 May 2008 .

[7] Open Source Computer Vision Library, http://www.intel.com/technology/computing/opencv/, last access 8 May 2008.

[8] R. Picard. *Affective Computing*. MIT Press, Cambridge (MA), USA, 1997.

[9] Qt Cross-Platform Application Framework (2008), http://trolltech.com/products/qt, last accessed 6 May 2008.

[10] S. Lucey, I. Matthews, C. Hu, Z. Ambadar, F. de la Torre and J. Cohn. AAM Derived Face Representations for Robust Facial Action Recognition. In *FGR '06: Proceedings of the 7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 155–162, Washington, DC, USA, 2006. IEEE Computer Society.

[11] The CMU Sphinx Group Open Source Speech Recognition Engines, http://cmusphinx.sourceforge.net/html/cmusphinx.php, last accessed 8 May 2008 .

[12] The VxL Homepage, http://vxl.sourceforge.net/, last accessed 8 May 2008.