Project ref. no.	IST-2000-26095
Project title	NITE: Natural Interactivity Tools Engineering

Deliverable status	Public
Contractual date of delivery	31. March 2002
Actual date of delivery	9. December 2002
Deliverable number	D2.3
Deliverable title	Best practice gesture, facial expression, and cross-modality coding schemes for inclusion in the workbench
Type	Report
Status & version	Final
Number of pages	43
WP contributing to the deliverable	WP2
WP / Task responsible	DFKI
Author(s)	Michael Kipp, Norbert Reithinger, Nils Ole Bernsen, Laila Dybkjær, Malene Wegener Knudsen, Maria Machuca, Montse Riera
EC Project Officer	Philippe Gelin
Keywords	best practice, coding scheme, meta-scheme, coding module, metadata, natural interactivity, multi-level, cross-level, cross-modality
Abstract (for dissemination)	This report presents coding scheme recommendations for natural interactivity research as well as coding module specifications of the recommended schemes for possible inclusion in the NITE workbench. The recommendations are based on clear definitions of the notions of coding scheme, meta-schema, coding module etc. We devised evaluation criteria for all these concepts in order to support our recommendations. For facial expression coding, semi-standards have already developed. Three coding schemes are presented and recommended here. For gesture coding, we propose a modular approach for building a gesture coding scheme to cope with the widely diverging research aims. Two recommendations are presented for the most basic modules. For cross-modality coding, we point to cross-modality aspects in the surveyed schemes and conclude that there are no coding schemes yet that are generic enough to qualify for recommendation. Finally, we define coding modules for all three facial expression coding schemes and one recommended gesture coding scheme.

NITE Deliverable D2.3

Best practice gesture, facial expression, and cross-modality coding schemes for inclusion in the workbench

December 2002

Authors

Michael Kipp[1], Norbert Reithinger[1], Nils Ole Bernsen[2], Laila Dybkjær[2], Malene Wegener Knudsen[2], Maria Machuca[3], Montse Riera[3]

1: DFKI, Saarbrücken, Germany. 2: NISLab, University of Southern Denmark, Denmark.

3: Universitat Autonoma de Barcelona, Spain.

Contents

1 Introduction

Standards for the annotation of corpora develop empirically over time. When annotators and consumers of corpora start adding new types of information to collected data files, and when these corpora are shared, usually a consensus emerges which type of information and which annotation style is the most useful. The area of gesture, facial expression, and cross-modality annotation does not have a very long tradition in mass data annotation, compared to, e.g. syntactic or dialogue act annotation, where large corpora are available. Therefore, recommendations for best practice or standards may continue to change for some time to come. In NITE, our goal is to provide the most comprehensive overview of earlier work and to base our decisions on the current state of the art.

This deliverable finalises a series of three NITE reports. In D2.1 ("Survey of existing gesture, facial expression, and cross-modality coding schemes"), we surveyed previous work, and in D2.2 ("The NITE markup framework") we laid the foundation of the markup framework to be used in the NITE natural interactivity coding tool prototypes. Finally, in this report we describe best practice in natural interactivity coding as we currently are able to define it using all available sources of information, especially the surveys and experiences of the ISLE project.

To define best practice, we first clarify some terminological issues in Section 2. We say, for example, what a coding scheme is as opposed to a meta-scheme. In Section 3 we define a number of general and specific criteria for coding schemes and meta-schemes that should be respected when selecting a certain scheme or defining a new one. Even though this is adventurous to do in the current state of the art, we finally present recommendations for meta-schemes as well as for gesture and facial expression coding schemes in Section 4. In the appendix, we provide coding modules for gesture and facial expression coding that can be used as a blueprint for users of the NITE systems.

2 Terminology

In this section, we introduce a number of relevant notions: meta-scheme, coding scheme, coding module, etc. We refer to the notions of coding module and metadata as defined in NITE Deliverable D2.2.

2.1 Meta-Schemes

To clarify the question of what a coding scheme is, we may start by saying what it is *not*. It is not a meta-scheme. So what is a meta-scheme? A meta-scheme is a framework of concepts that allows the definition of specific coding schemes. The meta-scheme is like a box of tools whereas the coding scheme is the product to be manufactured with the help of those tools. The wider the range of available tools and the more powerful their functionality, the more likely we will be in succeeding to implement whatever scheme we need. A coding scheme can also be considered an *instance* of the meta-scheme. Examples of meta-schemes are:

- Annotation Graphs (Bird, Liberman 2001)
- Anvil (Kipp 2001)
- BAS Partitur (Schiel et al. 1998)
- CHAT (CHILDES project)
- HIAT (Ehlich 1992)
- MATE (McKelvie et al. 2001)

One must distinguish between the syntax and the semantics of a meta-scheme. This is important because some researchers focus on syntax, some on semantics when defining a meta-scheme. BAS Partitur, for instance, called a file format by its creators, is a syntactic construct. Yet the file format always has a semantics that can be expressed formally, e.g. by using grammars or graphs. Annotation Graphs are formally defined using mathematical graphs. They express the semantics of the ATLAS annotation framework which can be called a meta-scheme. The key to the meta-scheme definition is the semantics. More often than not, the semantics is defined in a non-formal fashion (Partitur, CHAT, Anvil), which is not a serious flaw if compared to, e.g., programming languages which are usually learned in a non-formal way as well.

How do we describe what a meta-scheme is? Probably the best approach is by listing some common properties of existing meta-schemes. Although we will never have a meta-meta-scheme (hopefully not!), we will probably someday see the most general meta-scheme whereof all other meta-schemes are just special cases. Meta-schemes distinguish between the primary data source (video, audio, picture) and annotation (which can be considered secondary). The annotation consists of *units* which are added by the coder. Units are either simple strings (CHAT, Partitur) or structured objects with, e.g., a typed feature structure (Anvil). These units are often associated with a level, layer, tier, track or type (we will stick to the term *layer* in the following). Layers can have two purposes. Firstly, a layer can prescribe the internal structure of the units within this layer. In a typed feature structure approach (Anvil), this would mean that the features and their corresponding types are fixed within a layer. Secondly, a layer can prescribe the organisational structure of the units. For instance, the units could be arranged in temporal order without overlap (Anvil, CHAT, Partitur) or in a hierarchical fashion. Annotation Graphs only have one global arrangement along a general timeline. Hierarchical information is implicitly encoded.

2.2 Coding Module

The discussion in the previous section will help us defining the notions of coding scheme. Also, we will look into the components of good coding practice. Both concepts are subsumed by the notion of *coding module* which in NITE Deliverable D2.2 is described as “everything that is needed in order to perform a certain kind of markup of a particular natural interactivity corpus”.

2.2.1 Coding Scheme

What does the meta-scheme discussion in Section 2.1 above tell us about coding schemes? In a coding scheme, we have to decide on the number of layers, on how information is stored in a layer’s units (e.g. in feature-structures), and how units are to be arranged within their layer (temporally, hierarchically, otherwise). By declaring this information within the limits of our meta-scheme, we have fixed the *syntax* of our coding scheme. In Anvil, for instance, this is done in the specification file. In some cases the syntax can be called the *tagset*. The coding scheme *semantics* is defined in the coding manual. For example, we may define a „gesture“ layer where units include a feature called „handshape“ with a number of possible values like „fist“, „open flat“, „open curved“, etc. The coding manual should then explain

- **segmentation**: criteria where a single gesture starts/ends
- **classification**: criteria for identifying the handshape category („fist“, „open flat“ etc.)

If the coding scheme demands annotation of organisational structure (e.g. syntax trees), then the coding manual should also explain

- **organisation**: criteria to determinecorrect linkage/attachment (e.g. the PP attachment problem)

Coding scheme syntax and semantics are the main components of the *coding module*.

It is in the semantics of a coding scheme that we can distinguish between a *descriptive* and an *interpretative* scheme. A descriptive coding scheme has a semantics that can be formally defined. Handshapes, for instance, are descriptive as are anything that can be counted or otherwise measured by objective means: location, speed, shape, distance, angle etc. In contrast, interpretative schemes make use of human competence and implicit rules and correlations yet undiscovered and not exhaustively defined. For example, coding a gesture as a turn-taking signal is an interpretative act based on the coder's competence as a member of his/her language community. Of course, there are borderline cases where it is hard to tell whether a scheme is descriptive or interpretative, e.g. the coding of syntax trees.

There are two important adequacy criteria for coding schemes: *feasibility* and *consistency*. Descriptive schemes can easily be checked for consistency since they have a formal semantics. It may be, though, that a descriptive scheme is hard to apply for a human coder. If the task is to encode the elbow angle with one of fifteen possible values, this may be impossible for a human coder to reliably do. This is what we mean with feasibility. Feasibility for descriptive schemes can be checked with intra-coder reliability tests. For interpretative schemes, two error sources exist and are actually hard to tell apart. If a coder cannot discriminate between two categories it may be because the two are too hard to distinguish (feasibility) or because their definitions overlap, i.e. there is an inconsistency in the scheme. Inter-coder checks are needed to find such cases.

2.2.2 Coding Practice

Coding practice can be defined by answering the following questions:

- who is coding (human vs. computer, expertise)?
- how is the coder trained (pre-coding period)?
- how is the coder supported (coding period)?
- how is the coder controlled (coding and post-coding period)?

Moreover, we distinguish between two types of coding:

1. Fixed-scheme coding and
2. Evolving-scheme coding

In the first case, a fixed coding scheme, possibly a standard scheme where deviation from the standard would cause problems in usability of the annotation, is used and kept without changes. In the second case, coding starts with a coding scheme prototype that will be continually adapted to better cover the data. The scheme evolves in a cyclic process of coding and scheme revision.

Who is coding? First of all, it must be decided what kind of coder is desired. Coders with expert knowledge (high expertise) will keep training time short but may also be biased toward a certain theory. This is not necessarily a disadvantage but must be kept in mind when working with the resulting annotations. A bias is also introduced when using so-called bootstrapping techniques where coders use automatic classifiers to speed up the process. The computer suggests a number of prioritised categories for coding which is then

checked/confirmed by the human coder. This approach may lead to annotations that support the theory/statistics underlying the used classification algorithm.

How is the coder trained? Before coding, coders must first read the coding manual and look at sample annotations. As a second step, they either watch a trained coder doing annotations or they do annotations themselves and are supervised by a trained coder. Finally, the new coder does annotations on his/her own which are either corrected by a trained coder or for which there are already correctly annotated versions available that can be used for comparison. These steps should be quantified and documented (how many hours, how many sample files, a definite training suite of sample files).

How is the coder supported? The main support is the coding manual. For the manual one must ask whether it is available on-line and whether it is conceptualized as a reference book (quick access to information). On-line help can have many different faces and basically boils down to making the coding manual entries available at all those steps where categorization must be done – this is clearly the task of the coding tool designer. As for the design of the coding manual it depends too much on the actual scheme what design to choose. Some points to remember are:

- clearly distinguish between segmentation, classification and organization
- lead the coder from more general criteria to more specific criteria
- try to structure your classes in such a way that the user can find a classification by answering a number of questions or applying ‚objective' measures (counting)
- give many examples, use video-stills
- give cross-references if two categories A and B are similar and often confused (so that the coder automatically checks for B when looking at A)
- give examples for *borderline cases* where various interpretations (A or B) seem plausible

How is the coder controlled? A coder must be checked on a regular basis because systematic misinterpretations might sneak in. Also, the coding scheme might change in the process of coding and systematic checks are a good means to doublecheck that the coders stick to the changes. Checks can be informal by letting coders check arbitrarily selected annotations of another coder. Or checks can be based on agreement metrics like the kappa value. For this, a single data file must be be coded by two or more coders. One should decide on one of these control methods before coding starts and agree on standard time intervals between checks.

2.3 Coding File and File Format

A coding scheme defines the structure of actual annotations. Annotations are stored by writing them to a physical storage device like a harddisk or CD-ROM as a *coding file*. The file format is the physical representation of an annotation. It is a syntactically sound manifestation of the scheme's concepts. A scheme's file format can be devised using XML, ASCII or binary encoding. All types have different advantages/disadvantages. Binary encoding is cheapest in terms of memory but is more difficult to decode. It is not readable for humans and must come with a detailed and absolutely accurate specification to be decodable. Binary encoding is hardly used any more since memory is cheap. ASCII is human-readable but there are no standards for formal syntax specification, let alone semantic specification. So, checking syntactic correctness in two different systems is error-prone due to possibly differing interpretations of semi-formal specifications. The XML standard builds on ASCII

and provides a formalism for syntactic specification (DTD) and type specification (XML Schema). The formalism guarantees generality of syntax checks. Moreover, XML, being a standard, has led to the development of tools that allow visualisation, editing, checking, and transformation of XML files in a comfortable way. Furthermore, programmers can make use of existing parsers (e.g. Xerces) and internal representation mechanisms (document object model, DOM). It is quite clear that XML is the most suitable file encoding format to date.

XML (or ASCII or binary) is only the basic encoding. On top of this is the meta-scheme's file format. A concrete scheme's file format is an *instance* of the meta-scheme's format. For example, Anvil has a meta-scheme (Anvil's track-based annotation framework) and a corresponding file format. Concrete schemes are encoded in so-called specification files and define an instance of Anvil's file format for a particular scheme. Actual data files (Anvil annotation files) contain a reference to the corresponding specification file which encodes the concrete scheme, thus allowing syntax checking of particular coding schemes.

2.4 Metadata

Metadata is information about some data: Where does it come from? When and by whom was it created? For what purpose, using which methods etc.? Metadata is usually very small and structurally simple data (alphanumerical strings). The main purpose of metadata is to enable researchers to browse corpora, i.e. to search and find data using several dimensions. Moreover, metadata can be used to automatically sort, update, and merge databases since it provides a kind of semantics of the particular encoding.

As pointed out in NITE D2.2, metadata is directly attached to each of the three layers of data that we encounter in annotation: raw data, coding module and coding. For each of these layers, metadata should comply with a standardised set of metadata tags. Several standardization initiatives exist (IMDI[1], OLAC[2], DCMS[3]), IMDI being the most relevant for NITE as an initiative restricted to developing metadata sets for multimedia/multimodal corpora and lexica (Wittenburg et al. 2000).

IMDI suggests the use of *structured* metadata elements whereas the DCMS only allows a *flat structure*, i.e. a set of strings. For example, two recording participants and their respective age would be encoded in DCMS like this:

```
contributor=john
contributor=mary
age=20
age=22
```

[1] ISLE MetaData Initiative, http://www.mpi.nl/ISLE/

[2] Open Language Archives Community, http://www.language-archives.org

[3] Dublin Core Metadata Set, http://dublincore.org

In IMDI, a structured representation allows to see (and query) what age belongs to which person:

```
participant
        participant.name=john
        participant.age=20
participant
        participant.name=mary
        participant.age=22
```

So in order to be able to query metadata in a flexible and powerful way, we should rely on structured metadata representations.

2.5 Cross-Modality Coding

Cross-modality coding refers to annotations that serve the exploration of interrelationships between two or more modalities. In a strict sense, modalities are derived from the human senses: vision, hearing, touch, smell etc. In multimodal and other research though, in the modality of vision is often *further* differentiated by the *medium* used to communicate, i.e. the face, the hands, the whole body. For our purposes, we take a rather arbitrary (non-exhaustive) list of modalities often encountered in the research literature:

- speech
- gesture
- facial expression
- gaze
- body posture

Modalities are natural containers for coding information, i.e. they usually correspond to one or more layers. For instance, for speech, different types of linguistic information (part-of-speech, syntax, rhetorical structure etc.) are usually encoded on various layers.

A coding scheme is called *cross-modal* if any kind of relationship must be coded between layers of different modalities. Such relationship annotations can take on different forms:

- **Shared external reference**: Two elements on different levels (e.g., speech and gesture) co-refer to the same external object which is represented by a simple string ID. This object can be something concrete (person, location, object) or abstract/emotional (co-expressive speech/gesture).
- **Structural level relationships**: Very often, layers are structured in a hierarchical fashion, such that elements of level B are defined by a start and end element on level A. Level B is then called secondary to level A (e.g., CHAT, HIAT, Anvil, Partitur). Using this relationship, a gesture on level B can be related to words on level A.
- **Cross-level links**: Structural level relationships are very inflexible in the sense that, inherently, the temporal extension of secondary elements is determined by the elements they refer to (start time = start time of first element; end time = end time of last element). Cross-level links allow elements of level B to refer to arbitrary elements of level A (or any other level). Thus, if a gesture has a lexical affiliate on a speech layer (e.g., a word element), the gesture can be linked up with it without having to co-occur temporally with this element.

- **Abstract relations**: A variation of cross-level links are arbitrary relations that exist outside the temporally anchored annotation. Such nontemporal elements can be used as *connectors* to define relationships between different modalities, i.e. levels. Instead of linking a gesture to a word as mentioned above, one could create a nontemporal relation entity called "lexical affiliate" that takes two arguments: one gesture element and one word element.

Which type of relationship to choose for a specific coding scheme depends on the research task. A coding tool should support all four techniques.

3 Evaluation Criteria

3.1 Evaluation Criteria for a Meta-Scheme

A meta-scheme usually comes with a tool or, to put it the other way round, a tool implements a meta-scheme. Very often, tool and meta-scheme are not even explicitly distinguished (Anvil, HIAT, MATE). When evaluating a meta-scheme's applicability for a specific task, we can apply the following criteria:

- **Expressiveness**: The overall framework of the scheme (AGs, Tracks, DAGs) must be expressive enough to accommodate all the data properties we wish to encode.
- **Searchability/Tractability**: Relevant/interesting relationships should be easy to query for a user and efficiently computable.
- **Maintainability**: Implemented schemes will have their data written to formats that can easily be checked, converted, edited (XML, databases).
- **Metadata**: Meta-Schemes should support the inclusion of metadata according to a standard like IMDI, DCMS or OLAC. Since metadata is scheme-independent, it must be part of the meta-scheme.

These criteria/guidelines are very general. We will add more detail in the next section.

3.2 Features of Meta-Schemes

Most meta-schemes were developed to accommodate a specific coding scheme. The most common feature of these coding schemes was is the fact that the annotation should be performed on several layers that could be viewed and edited in temporal alignment. Apart from this, a number of other features have evolved, for example, hierarchical relations between tracks, structured annotation elements (as opposed to simple alphanumerical strings), hierarchical tagsets etc.

The following table presents a catalogue of such features. It also shows the specific coding schemes that require the existence of a particular feature to be implementable:

Feature	Schemes that need this feature
Multiple layers	SmartKom, Poggi/Magno Caldognetto, McNeill
Layer hierarchy	CHAT, Kita
Graphical symbols as values	HamNoSys, Birdwhistell
Structured objects	Poggi/Magno Caldognetto, SmartKom
Hierarchical values	CHAT
Non-temporal objects	LIMSI
Cross-level links	LIMSI, HIAT
Metadata	HIAT, CHAT
Free-form comments/descriptions	McNeill

3.3 Evaluation Criteria for a Coding Scheme

The following criteria for evaluating coding schemes can be found in the literature:

- **Documentation**: The scheme must be consistently and completely documented in a coding manual.
- **Simplicity/Orthogonality**: The representation should be as orthogonal across layers as possible, i.e. the same aspect should not appear encoded in two different layers, because this would not only increase work but also the number of possible errors.
- **Data-drivenness**: The scheme should enable you to code all the data, i.e. there should be no gaps, no pieces of data that cannot be coded because the scheme does not offer suitable constructs.
- **Tool supported**: The scheme should be an instance of a meta-scheme that is supported by tools (coding).
- **Reliability**: Coding should be reliably possible. There are two factors:
  - **feasibility**
  - **consistency**
- **Extensibility**: The scheme should be open to new entities since much research is exploratory (evolving-scheme coding) were categories are possibly made up on the fly.

3.4 Evaluation Criteria for a File Format

- **Encoding Format**: This should be XML for maximal efficiency in terms of data exchange, maintenance, query and software development. Examples of meta-schemes and schemes using XML are:
  - AIF (ATLAS Interchange Format)
  - Anvil
  - EAF (EUDICO/Elan)
  - TASX

  - The Observer (under development)
- **One File vs. Many Files**: annotations should be kept in one file if manipulations are mainly done via tools; annotations should be kept in many files if manipulations are mainly done manually (direct editing, simple scripts)
- **Stand-Off Representation**: see MATE (McKelvie et al. 2001)

4 Recommendations

Having discussed the differences of schemes and meta-schemes and outlined criteria for the evaluation of each, we will now proceed by applying the criteria to our meta-schemes and coding schemes and present recommendations.

4.1 Meta-Schemes

The following table shows available meta-schemes and their features. "Y" means that the feature is fully supported, whereas "~" means that only part of the feature is integrated or that it is left open in the specification whether a tool would support it. The meta-schemes marked with an asterisk (*) come with a fixed coding scheme but are generic enough to make the transition to a pure meta-scheme obvious.

	Multiple layers	Layer hierarchy	Structured obj.	Hier. values	Nontemp. obj.	Cross-level	Metadata	Comments
AGs	Y	~	~	~		~		
Anvil	Y	Y	Y		Y	Y	~	Y
BAS Partitur	Y	Y					Y	
CHAT (*)	Y	Y		Y			Y	Y
HIAT (*)	Y						~	Y
MATE	Y	Y	Y			Y	Y	

As far as the features in the table are concerned, there are three meta-schemes that appear recommendable:

- **Anvil**: It is the only meta-scheme offering non-temporal objects (Martin, Kipp 2002) and there is a working tool (freeware).
- **Annotation Graphs**: This seems to be the future American standard. It very open and flexible, which can also be considered a drawback since many features are not explicitly specified yet. The tools are open source.
- **MATE**.

All meta-schemes correspond to a tool: Anvil and MATE are names of tools themselves, Annotation Graph tools are currently under development within the ATLAS project.

4.2 Gesture Coding Schemes

Considering the many different research aims around we do not think that one single gesture coding scheme could be declared best practice for coding gestures in natural interactivity research. Rather, we see gesture coding as being decomposable into several modules dealing with different aspects which bear a slight resemblance to linguistic dimensions.

Figure 4.2.1 illustrates the modules which build upon each other in three layers. Both two bottom layers have a basic module (left) and a more advanced module (right) each. In the very bottom layer you find purely descriptive, non-semantic schemes. The basic module (left) deals with segmentation of the different gesture phases. The advanced module (right) covers the more detailed encoding of gesture form, i.e. hand/arm location, trajectory, hand shape etc. On the middle layer, using syntactic rules (see Kita et al. 1998), the basic gestural movements are assembled into gestures and categorised using either very basic categorisations (left module), i.e. emblems, deictics, adaptors etc. (cf. McNeill 1992) or an elaborated gesture lexicon (right module). On the top layer, the function of the gesture is being analysed and coded. This layer can consist of many dimensions like in Poggi/Magno Caldognetto's scheme, so that function is again coded on many levels/layers. Or it could be simply a link indicating co-reference with a speech object (LIMSI coding scheme) which is also an interpretative decision.

Having introduced these three layers of gesture coding makes it easier to assemble a "best practice" gesture coding scheme because we can propose recommendations for each module. Researchers can then "plug" their own scheme together using existing schemes. Note that the bottom-up direction of the illustration corresponds with the coding process itself. One usually approaches a gesture by first describing it objectively and then progressively doing more and more interpretative work. This order of going from objective measures to more subjective ones is known to increase reliability in coding.

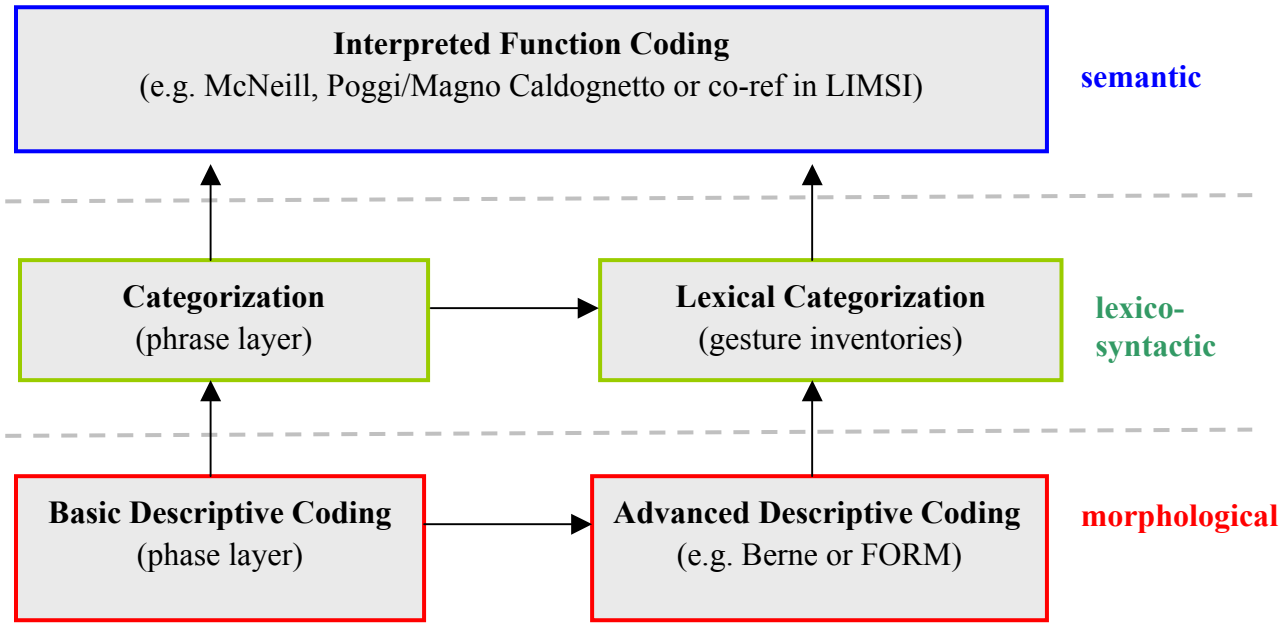

Figure 4.2.1. Layers of gesture annotation.

The following table shows which gesture coding scheme is suitable for which part of the coding in the three-layered view suggested in Figure 4.2.1. Kita et al.'s (1998) coding of phases (based on work by Adam Kendon and David McNeill) can be considered a standard. Also, for hand shape, HamNoSys seems to become a standard in Europe. American

researchers tend to use ASL codes for hand shapes. For Advanced Descriptive Coding there are several alternatives, ranging from very rough descriptions (McNeill), to "kinematically-based" ones (FORM or, for maximum detail, the Berne system).

Basic Descriptive Coding	MPI Movement Phase (Kita et al. 1998)	
Advanced Descriptive Coding	Handshape	HamNoSys
		ASL
	Berne (Frey et al. 1983)	
	McNeill (1992)	
	FORM (Martell 2002)	
Categorisation	McNeill (1992)	
Lexical Categorisation	n/a	
Interpreted Function Coding	Poggi, Magno Caldognetto (1996)	
	LIMSI (Martin et al. 2001)	
	McNeill (1992)	
	SmartKom	

To conclude, we recommend the MPI Movement Phase scheme for basic descriptive coding. For more advanced descriptive coding as well as for categorisation we recommend McNeill's coding scheme since it is widely used by gesture researchers, offers a moderate detail in terms of formal description, and is well-documented in (McNeill 1992). Depending on the research aim, one will have to decide on a more sophisticated kinematically-based encoding (e.g., when dealing with gesture recognition or computer animation). As for interpretative schemes, it does not make sense to recommend a scheme without knowledge of the pursued research.

4.3 File Format

As file encoding format we recommend XML. For the file format for storing annotations we recommend a standard like AIF or MATE.

4.4 Metadata

We recommend to use structured metadata as opposed to the flat structures used in the DCMS. The coding tool should allow the inclusion of *generic, structured metadata*. Metadata should comply with, or at least be compatible to, the existing and coming IMDI standards.

4.4.1 Raw Data Metadata

The following metadata is suggested in D2.2 for raw data, emphasising that the list is not intended to be exhaustive. The corresponding IMDI elements are juxtaposed (the star "*" is a placeholder for further IMDI subcategories).

D2.2 Label suggestion	IMDI Session Element
raw data referenced	Resources.Source.*
date of creation of raw data	Date
date of creation of metadata	

name(s) of creator(s) of raw data	Collector.{Name, Contact, Description}
name(s) of creator(s) of metadata	
location for creation of raw data	{Continent, Country, Region, Address}
purpose of creation of raw data, modalities involved, etc.	Description Content.{Modalities, Languages, Description}
interacting participants and roles	Participants.*
speaker characteristics (age, gender, native language, geographical provenance etc.)	Participants.Participant.*
size of raw data (duration, number of dialogues, file size)	Resources.MediaFile.Size
accessibility (commercial/free, contact information)	Resources.MediaFile.Access
file technicalities (file format, compression, etc.)	Resources.MediaFile.{Type, Format, Quality}
recording setup description (environment, equipment, setup etc.)	Resources.MediaFile.RecordingConditions
application description (subjects, task, scenario, instructions, domain knowledge etc.)	Content.*
references to other relevant raw data	References.*
references to literature	References.*
notes	Description

4.4.2 Coding Scheme Metadata

There seem to be no standards yet for coding scheme metadata. Coding scheme metadata may be seen as part of the coding module because the coding module can be viewed as common sense reasoning about what is useful information for other coders using the scheme. Therefore, we refer to NITE D2.2 where the following metadata for attachment to the coding module is recommended:

- author(s)
- version
- purpose of the coding module
- coding level(s)
- description of required data source types
- references to other coding modules
- coding procedure
- coding example
- coding semantics
- markup declaration
- coding files referenced

4.4.3 Coding Session Metadata

Since, again, there seem to be no standards for coding-session metadata, we refer to NITE D2.2 where a minimal list of metadata is recommended for attachment to coding files:

- reference to the coding file body
- first creation date and coder
- for all revisions: date and coder
- name(s) of coder(s) and their characteristics
- version number
- reference to the applied coding module
- references to associated coding files and raw data
- purpose of creation
- level(s) annotated
- history of coding procedure
- note
- references to literature

For more information, consult ISLE deliverable D10.2.

4.5 Facial Expression Coding Schemes

As found in ISLE Deliverable D9.1 (Knudsen et al. 2002a), the situation for facial expression coding schemes is quite clear. MPEG-4 is widely used and considered a standard. Ekman and Friesen's (1978) FACS is also in use but, having been developed for investigation of emotions without computer applications in mind, it has some drawbacks, e.g. that it is not suited for lip movement markup.

All other facial schemes reviewed in ISLE D9.1 are being considered rather isolated efforts without having a chance of competing with MPEG-4 or FACS as standards for the moment. There probably exist many more facial coding schemes belonging to the category of isolated efforts. However, if they are not well-described and often only used by a single person or two, they are difficult to find. The picture, provided by the ISLE survey, of a proliferation of home-grown coding schemes is supported by the 28 questionnaires in (Knudsen et al. 2002b), asking people at a multimodal interaction workshop, e.g., which coding scheme(s) they had used or planned to use for data markup. Some people did not answer the question or had not made a decision yet as to which coding scheme to use. However, in no less than 15 cases the answer indicated that a custom-made scheme would be, or was being, used.

MPEG-4 and FACS are clearly the two facial coding schemes we recommend for inclusion in the NITE workbench as constituting best practice or even standard in their field. The appendix describes each of the two coding schemes in terms of a coding module, cf. NITE D2.2. This means that not only the phenomena which can be marked up are described but also important meta-data information about the coding scheme is provided which serves to enable other users to use the scheme correctly and reliably.

A third scheme, ToonFace, is added in the appendix and also described as a coding module. ToonFace was also reviewed in ISLE D9.1 and, although it is far from being as widely used as MPEG-4 and FACS, we have two reasons for including it here. ToonFace is different from the two other schemes by only being suited for 2D models and it is the only other reviewed facial coding scheme in ISLE D9.1 which is being used across a number of sites and not only by one or two persons. ToonFace may also be considered for inclusion in the NITE software. However, one should be aware that ToonFace already comes with an editor tailored to its use.

It is interesting that the most frequently used facial coding schemes are all at what one might call the syntactical level, drawing a parallel to, e.g., part-of-speech tagging in the linguistic area. When it comes to semantics there is no standardisation.

4.6 Cross-Modality Coding Schemes

This section deals with aspects of cross-modality coding in the surveyed coding schemes, focusing on speech-gesture research. We conclude that no recommendations for a generic cross-modality coding scheme can be made yet.

4.6.1 Cross-Modality Coding in the Surveyed Coding Schemes

Taking a second look at the coding schemes surveyed in NITE D2.1, we identify the schemes that allow cross-modality coding. As outlined in Section 2.5, we consider the following modalities: speech, gesture, face (includes gaze) and posture. If a scheme allows coding of one such modality, this does not mean that it supports cross-modality coding. To implement cross-modality, a scheme must enable the encoding of relationships between annotated elements of different modalities by means of one of the techniques described in Section 2.5. The following table shows in which modalities coding takes place for the respective scheme, and whether relationships across modalities are encoded.

	modality				**cross-modality**		
	speech	**gesture**	**face**	**posture**	**speech-gesture**	**speech-face**	**speech-posture**
Berne (Frey et al. 1983)		X	X	X			
FORM (Martell 2002)		X					
HamNoSys		X					
MPI Movement Phase (Kita et al. 1998)		X					
Bull (1987)		X		X			
Schegloff (1984)	X	X			X		
CHAT	X	X			X		
DIME	X	X			X		
HIAT (Ehlich 1992)	X	X			X		
LIMSI Tycoon (Martin et al. 2001)	X	X			X		
McNeill (1992)	X	X			X		
Poggi, Magno Caldognetto (1996)	X	X	X	X	X	X	X
SmartKom (Steininger 2001)	X	X			X		

The relationship that is almost always examined is that between speech and some other modality. Apart from that, we are still a long way from cross-modality research (e.g. face and gesture, posture and gesture, face and gaze) that produces a general treatment of cross-modality coding. The next section will give some pointers in the literature concerning the well-covered area of speech-gesture cross-modality research.

4.6.2 Cross-Modality Research for Speech and Gesture

The issue of how gestures and speech relate in time is critical for understanding gestures and speech as part of the multimodal expression (McNeill, 1992). According to McNeill et al. (2001), the organisation of discourse is inseparable from gesture and prosody.

Three parts in the realisation of gesture should be taken into account to relate speech to gesture: the preparation of the gesture, the most energetic part of it and the relaxation of the gesture.

Prosodic parameters related with types of gesture are shown in the following summarising findings from Cruttenden (1986), Guaïtella (1991), Cavé et al. (1993), Bertrand et al. (1995), Cassell et al. (1994 a, b), Bruce (1999), Kettebekov et al. (2002), McNeill (1992), McNeill et al. (2001).

Prosodic elements related to gesture as mentioned in the reviewed literature	**Non verbal cues**
Intonational phrases	Gesture preparation Gesture relaxation
Pauses	Raising of the eyebrows Blinking Nodding of the head
Sentence stress	Stroke phase of the gesture
Syllable stress F0 prominences	Raising of the eyebrows Blinking Nodding of the head
Boundary tones	Gesture preparation Gesture relaxation
(Nuclear) sentence stress	Stroke phase of the gesture
Emphasis	Hand movements
F0 prominences	Head movements Eyebrow movements Hand movements
F0 range	Eyebrow movements
F0 movements and shape of the F0 contour	Head movements Eye movements Eyebrow movements Acceleration contours of the moving hand
Intensity changes	Eye movements Hand movements

5 References

Bertrand, R., Boyer, J., Cavé, C., Guaitella, I. and Santi, S. (1995) Voice and gesture relations in interaction situations: some prosodic and kinesic aspects of back-channel, in: Elenius, K.- Branderud, P. (Eds.) *Proceedings of the XIIIth International Congress of Phonetic Sciences*. Stockholm, Sweden, Vol.2., pp. 746-749.

Bird, S. & Liberman, M. (2001) A Formal Framework for Linguistic Annotation, in: *Speech Communication* **1-2**, pp. 23-60.

Bruce, G. (1999) Temporal control of fundamental frequency gestures, in: *Speech Communication and Language Development Symposium*, Stockholm University, June 1999.

Bull, Peter E. (1987) *Posture and Gesture*, Oxford: Pergamon Press.

Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S. and Stone, M. (1994b) Animated Conversation: Rule-Based Generation of Facial Expression, Gesture and Spoken Intonation for Multiple Conversational Agents, in: *Proceedings of SIGGRAPH '94.*

Cassell, J., Stone, M., Douville, B., Prevost, S., Achorn, B., Steedman, M., Badler, N., Pelachaud, C. (1994a) Modeling the Interaction between Speech and Gesture, in: Ram, A., Eiselt, K. (Eds.) *Proceedings of the Sixteenth Anuual Conference of the Cognitive Science Society.* Lawrence Erlbaum Associates, pp. 153-158.

Cavé, C., Guaitella, I. and Santi, S. (1993) Fréquence fondamentale et mouvements rapides des sourcils: une étude pilote, in: *Travaux de l'Institut de Phonétique d'Aix* 15: 25-42.

Cheyer, A., Julia, L. & Martin, J.C. (2001) A Unified Framework for Constructing Multimodal Experiments and Applications, in: *Cooperative Multimodal Communication*. Bunt, H., Beun, R.J. & Borghuis, T. (Eds.). Second International Conference, CMC'98, Tilburg, The Netherlands, January 1998. Springer. LNAI2155.

Cruttenden, A. (1986) *Intonation*. Cambridge: Cambridge University Press.

Ehlich, K. (1992) HIAT – a Transcription System for Discourse Data, in: *Talking Data: Transcription and Coding in Discourse Research*, J.A. Edwards & M.D. Lampert (eds.), Hillsdale: Erlbaum, pp. 123-148.

Ekman, P. and Friesen, W.V. (1978) *Facial Action Coding System*, Palo Alto, CA: Consulting Pychologists Press.

Frey, S., Hirsbrunner, H.P., Florin, A., Daw, W. and Crawford, R (1983) A Unified Approach to the Investigation of Nonverbal and Verbal Behavior in Communication Research, in: W. Doise and S. Moscovici *Current Issues in European Social Psychology*, Cambridge: Cambridge University Press, pp. 143-199.

Guaitella, I. (1991) Étude des relations entre geste et prosodie à travers leurs fonctions rythmique et symbolique, in: *Actes du XIIème Congrès International des Sciences Phonétiques*. 19-24 août 1991, Aix-en-Provence, France. Aix-en-Provence: Université de Provence, Service des Publications. Vol. 3. pp. 266-269.

Kettebekov, S., Yeasin, M. and Sharma, R. (2002) "Prosody Based Audio-Visual Czo-analysis for Coverbal Gesture Recognition," *Submitted to IEEE Transactions Multimedia: Multimodal Interfaces and Applications*. Also, Technical Report CSE-02-015.

Kipp, M. (2001) Anvil - A Generic Annotation Tool for Multimodal Dialogue, in : *Proceedings of Eurospeech 2001*, Aalborg, pp. 1367-1370.

Kita, S., van Gijn, I., & van der Hulst, H. (1998) Movement phases in signs and co-speech gestures, and their transcription by human coders, in: Wachsmuth, I. & Fröhlich, M.

(eds.) *Gesture and Sign Language in Human Computer Interaction*. Berlin: Springer, 23-35.

Knudsen, M. W., Martin, J.-C., Dybkjær, L., Ayuso, M. J. M, N., Bernsen, N. O., Carletta, J., Kita, S., Heid, U., Llisterri, J., Pelachaud, C., Poggi, I., Reithinger, N., van ElsWijk, G. and Wittenburg, P. (2002a) *Survey of Multimodal Annotation Schemes and Best Practice*. ISLE Deliverable D9.1.

Knudsen, M. W., Martin, J.-C., Dybkjær, L., Berman, S., Bernsen, N. O., Choukri, K., Heid, U., Mapelli, V., Pelachaud, C., Poggi, I., van ElsWijk, G. and Wittenburg, P. (2002b) *Survey of NIMM Data Resources, Current and Future User Profiles, Markets and User Needs for NIMM Resources*. ISLE Deliverable D8.1.

Martell, C. (2002) FORM: An Extensible, Kinematically-Based Gesture Annotation Scheme, in: *Proceedings of the Third International Conference on Language Resources and Evaluation*, Denver, pp. 353-356.

Martin, J.C., Grimard, S., Alexandri, K. (2001) On the annotation of the multimodal behavior and computation of cooperation between modalities, in: *Proceedings of the Workshop on " Representing, Annotating, and Evaluating Non-Verbal and Verbal Communicative Acts to Achieve Contextual Embodied Agents ",* May 29, Montreal, Fifth International Conference on Autonomous Agents. pp 1-7.

Martin, J.C., Kipp, M. (2002) Annotating and Measuring Multimodal Behaviour – Tycoon Metrics in the Anvil Tool, in: *Proceedings of the Third International Conference on Linguistic Resources and Evaluation*, to appear.

McKelvie, D., Isard, A., Mengel, A., Møller, M.B., Grosse, M. and Klein, M. (2001) The MATE Workbench: An annotation tool for XML coded speech corpora, in: *Speech Communication* **1-2**, pp. 97-112.

McNeill, D. (1992) *Hand and Mind. What Gestures Reveal About Thought*. Chicago: University of Chicago Press.

McNeill, D., Quek, F., McCullough, K., Duncan, S., Furuyama, N., Bryll, R., Feng Ma, X. and Ansari, R. (2001) Catchments, prosody and discourse, in: *Oralité et Gestualité, ORAGE 2001 (Speech and Gesture 2001)*, Aix-en-Provence.

Poggi, I. and Magno Caldognetto, E. (1996) A Score for the Analysis of Gesture in Multimodal Communication, in: L. Messing (ed.) *Proceedings of the Workshop on the Integration of Gesture in Language and Speech (WIGLS)*, Delaware.

Schegloff, Emanuel A. (1984) On some gestures‘ relation to talk, in: J. Maxwell Atkinson and John Heritage (eds.) *Structures of Social Action*, Cambridge: Cambridge University Press, pp. 266-296.

Schiel, F., Burger, S., Geumann, A. and Weilhammer, K. (1998) The Partitur Format at BAS, in: *Proceedings of the First International Conference on Language Resources and Evaluation*.

Steininger, Silke (2001) Labeling Gestures in SmartKom – Concept of the Coding System, LMU, *SmartKom Report* **2**.

Wittenburg, P., Broeder, D. and Sloman, B. (2000) EAGLES/ISLE: A Proposal for a Meta Description Standard for Language Resources, in: *Proceedings of the Second International Conference on Language Resources and Evaluation*.

6 Appendix: Facial and gesture coding modules for possible inclusion in the NITE software.

In the following four sub-sections we describe the three coding schemes mentioned in Section 4.5, i.e. MPEG-4, FACS and ToonFace, and one gesture coding scheme (McNeill 1992) recommended in Section 4.2. The descriptions are in terms of the coding module entries mentioned in Section 4.4.2 and described in detail in NITE D2.2. We strongly recommend using the coding module entries for a thorough description of a coding scheme to make it usable to others than its creator(s). The coding module description will be supported by the NITE software. The descriptions below serve to exemplify the use of the NITE coding module and, at the same time, present in detail four coding schemes which may be incorporated in the NITE software to the extent time allows. Note that ToonFace already comes with an editor which is tailored to support markup using the ToonFace coding scheme.

6.1 MPEG-4

1. Name:

MPEG-4 SNHC (Moving Pictures Expert Group, Synthetic/Natural Hybrid Coding)

2. Author(s):

The Moving Picture Experts Group (MPEG) is a working group of ISO/IEC in charge of the development of international standards for compression, decompression, processing, and coded representation of moving pictures, audio and their combination.

Contact person:

Leonardo Chiariglione

Telecom Italia Lab

Via G. Reiss Romoli, 274

I-10148 Torino (Italy)

tel: +39 011 228 6120 / 6116 / 5111

fax: +39 011 228 6299 / 6190 /5520

Email: leonardo.chiariglione@tilab.com

Homepage: http://leonardo.tilab.com

The Moving Pictures Expert Group's Homepage: http://mpeg.telecomitalialab.com/

3. Version:

Not applicable.

4. Notes:

More information can be found here:

A description of MPEG-4: http://ligwww.epfl.ch/mpeg4

A description of parameter-based facial animation:

http://www.research.att.com/~osterman/AnimatedHead/index.html

A description of MPEG-4 compliant facial animation and Hybrid Video Coding: http://www-dsp.com.dist.unige.it/~pok/RESEARCH/index.htm

Tutorial issue on the MPEG-4 standard: Elsevier:
http://leonardo.telecomitalialab.com/icjfiles/mpeg-4_si/

Papers:

P. Doenges, F. Lavagetto, J. Ostermann, I.S. Pandzic and E. Petajan: MPEG-4: Audio/Video and Synthetic Graphics/Audio for Mixed Media. Image Communications Journal, vol. 5(4), May 1997.

J. Ostermann: Animation of synthetic faces in MPEG-4. Computer Animation'98, Philadelphia, USA, pp. 49-51, June 1998.

E. Petajan: Facial Animation Coding, Unofficial Derivative of MPEG-4. Work-in-Progress, Human Animation Working Group, VRML Consortium, 1997.

5. Purpose of the coding module:

MPEG-4 is an object-based multimedia compression standard, which allows for encoding of different audio-visual objects (AVO). The MPEG-4 SNHC group proposes an architecture for the efficient representation and coding of synthetically and naturally generated audio-visual information. MPEG-4 foresees that talking heads will serve an important role in future customer service applications. For example, a customised agent model can be defined for games or web-based customer service applications. To this effect, MPEG-4 enables integration of face animation with multimedia communications and presentations and allows face animation over low bit rate communication channels, for point-to-point as well as multi-point connections with low-delay. MPEG-4 also has derived a standard for facial animation coding.

6. Coding level(s):

MPEG-4 can be used for coding of facial expressions.

7. Description of data source type(s) required for use of the coding module:

The application of the coding scheme produces a text file while the application of this file produces an animated face which requires video rendering.

8. Explanation of references to other coding modules:

None, really. But note the following pre-condition. The coding scheme is meant to define a set of parameters to define and control facial models. A representation of a generic face with a neutral expression is required as point of departure. The shape, texture and expressions of the face are controlled by Facial Definition Parameters (FDPs) and/or Facial Animation Parameters (FAPs). Application of animation parameters will produce animation of the face. Definition parameters serve to change the appearance of the face from something generic to a particular face with its own shape and (optionally) texture.

9. Coding procedure:

It is probably sufficient to let one expert coder make the encoding and then let another coder check the result by watching the animated face.

10. Coding example showing the markup in use:

The picture of the 3D face in Figure 6.1.1 has been generated from a set of FAP values. FAP values are explained under entry 11. Figure 6.1.7 shows an example of FAP values. Note that Figures 6.1.1 and 6.1.7 are not related.

Figure 6.1.1. Facial expression generated from a set of FAP values.

11. Clear description of each phenomenon, example(s) of each phenomenon:
MPEG-4 defines a generic face model in its neutral state matching properties of a human face in its relaxed state. The head in its neutral state is defined as follows:

- gaze is in direction of Z axis
- all face muscles are relaxed
- eyelids are tangent to the iris
- the pupil is one third of the diameter of the iris
- lips are in contact
- the line of the lips is horizontal and at the same height as the lip corners
- the mouth is closed and the upper teeth touch the lower ones
- the tongue is flat, horizontal with the tip of tongue touching the boundary between upper and lower teeth

Feature Points:

For the animator to interpret the FAP values using its face model, the animator has to have predefined model specific animation rules to produce the facial action corresponding to each FAP. MPEG-4 specifies 84 feature points on the neutral face (see Figure 6.1.2). The main purpose of these feature points is to provide spatial references for defining FAPs. Some feature points such as the ones along the hairline are not affected by FAPs. However, they are required for defining the shape of a proprietary face model using feature points. Feature points are arranged in groups like cheeks, eyes, and mouth. The location of these feature points has to be known for any MPEG-4 compliant face model. The feature points on the model should be located according to Figure 6.1.2.

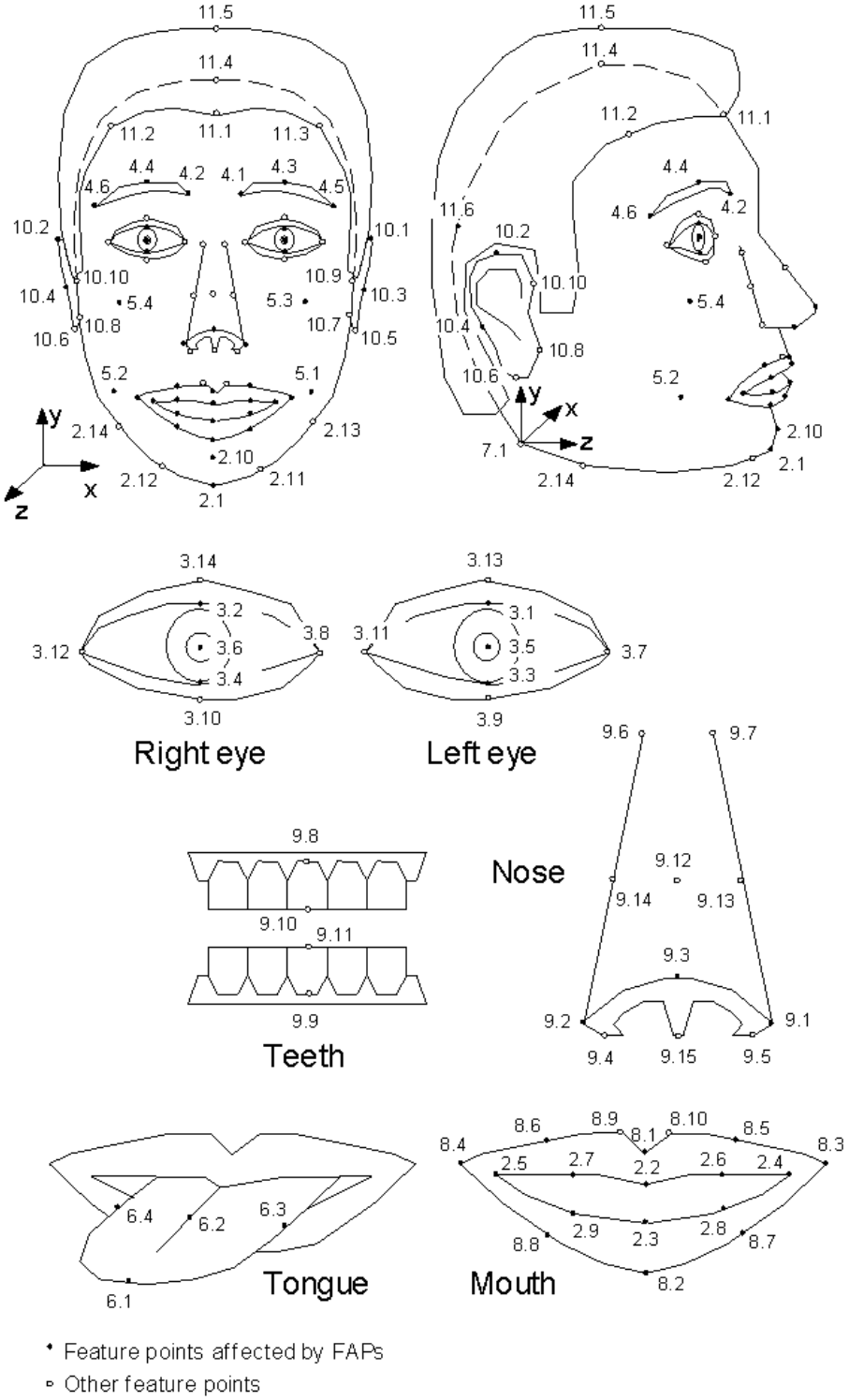

Figure 6.1.2. Feature points may be used to define the shape of a proprietary face model. The facial animation parameters are defined by motion of some of these feature points.

Deforming a neutral face model according to some specified FAP values at each time instant generates a facial animation sequence. The FAP value for a particular FAP indicates the magnitude of the corresponding action, e.g., a big versus a small smile or deformation of a mouth corner. For an MPEG-4 terminal to interpret the FAP values using its face model, it has to have predefined model-specific animation rules to produce the facial action corresponding to each FAP. The terminal can either use its own animation rules or download a face model and the associated face animation tables (FAT) to generate customised animated

behaviour. Since the FAPs are required to animate faces of different sizes and proportions, the FAP values are defined in face animation parameter units (FAPUs). The FAPUs are computed from spatial distances between major facial features on the model in its neutral state.

Face Animation Parameter Units:

A FAPU and the feature points used to derive the FAPU are defined with respect to the face in its neutral state, cf. Figure 6.1.3.

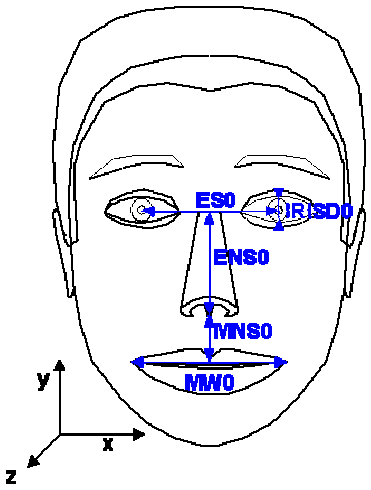

Figure 6.1.3. A face model in its neutral state and the feature points used to define FAP units (FAPU). Fractions of distances between the marked key features are used to define FAPUs (Face Animation Parameter Units).

In order to define face animation parameters for arbitrary face models, MPEG-4 defines FAPUs that serve to scale facial animation parameters for any face model. FAPUs are defined as fractions of distances between key facial features (see Figure 6.1.3.). These features, such as eye separation, are defined on a face model that is in the neutral state. The FAPU allows interpretation of the FAPs on any facial model in a consistent way, producing reasonable results in terms of expression and speech pronunciation. The measurement units are shown in Figure 6.1.4.

IRISD0	Iris diameter (by definition it is equal to the distance between upper and lower eyelid) in neutral face	IRISD = IRISD0 / 1024
ES0	Eye separation	ES = ES0 / 1024
ENS0	Eye - nose separation	ENS = ENS0 / 1024
MNS0	Mouth - nose separation	MNS = MNS0 / 1024
MW0	Mouth width	MW=MW0 / 1024
AU	Angle unit	10E-5 rad

Figure 6.1.4. Facial animation parameter units and their definitions.

Face Animation Parameters:

The FAPs are based on the study of minimal perceptible actions and are closely related to muscle actions. FAPs represent a complete set of basic facial actions including head motion, tongue, eye, and mouth control. They allow representation of natural facial expressions. There are 68 parameters that are categorised into 10 groups related to parts of the face, cf. Figures 6.1.5 and 6.1.6. For each FAP, the standard defines the appropriate FAPU, FAP group, direction of positive motion and whether the motion of the feature point is unidirectional (see,

e.g., FAP 3, open jaw) or bi-directional (see, e.g., FAP 48, head pitch). FAPs can also be used to define facial action units. Exaggerated amplitudes permit the definition of actions that are normally not possible for humans, but are desirable for cartoon-like characters.

Group	Number of FAPs
1. Visemes and expressions	2
2. Jaw, chin, inner lowerlips, cornerlips, midlips	16
3. Eyeballs, pupils, eyelids	12
4. Eyebrow	8
5. Cheeks	4
6. Tongue	5
7. Head rotation	3
8. Outer lip positions	10
9. Nose	4
10. Ears	4

Figure 6.1.5. FAP groups and number of FAPs per group.

The FAP set contains two high-level parameters, visemes and expressions (FAP group 1). A viseme (FAP 1) is a visual correlate to a phoneme. Only 14 static visemes that are clearly distinguished are included in the standard set. In MPEG-4, transitions from one viseme to the next are defined by blending only two visemes with a weighting factor.

The expression parameter FAP 2 defines 6 primary facial expressions (anger, joy, fear, sadness, disgust and surprise). In contrast to visemes, facial expressions are animated by a value defining the excitation of the expression. Two facial expressions can be animated simultaneously with amplitude in the range of [0-63] defined for each expression. The facial expression parameter values are defined by textual descriptions. The expression parameter allows for an efficient means of animating faces. They are high-level animation parameters. A face model designer creates them for each face model.

The remaining FAPs (66) are clustered in different groups (such as outer lip, cheeks, eyebrow). With the exception of some FAPs which control the head rotations, the eyeball rotations etc, each low-level FAP indicates the translation of the corresponding feature point, with respect to its position in the neutral face, along one of the coordinate axes.

#	**FAP name**	**FAP description**	Units	Uni- or bidirectiona	Position motion	Group	FDP subgroup number
1	viseme	Set of values determining the mixture of two visemes for this frame (e.g. pbm, fv, th)	na	na	na	1	na
2	expression	A set of values determining the mixture of two facial expression	na	na	na	1	na
3	open_jaw	Vertical jaw displacement (does not affect mouth opening)	MNS	U	down	2	1
4	lower_t_midlip	Vertical top middle inner lip displacement	MNS	B	down	2	2
5	raise_b_midlip	Vertical bottom middle inner lip displacement	MNS	B	up	2	3
6	stretch_l_cornerlip	Horizontal displacement of left inner lip corner	MW	B	left	2	4
7	stretch_r_cornerlip	Horizontal displacement of right inner lip corner	MW	B	right	2	5

#	FAP name	FAP description	Units	Uni- or bidirectiona	Position motion	Group	FDP subgroup number
8	lower_t_lip_lm	Vertical displacement of midpoint between left corner and middle of top inner lip	MNS	B	down	2	6
9	lower_t_lip_rm	Vertical displacement of midpoint between right corner and middle of top inner lip	MNS	B	down	2	7
10	raise_b_lip_lm	Vertical displacement of midpoint between left corner and middle of bottom inner lip	MNS	B	up	2	8
11	raise_b_lip_rm	Vertical displacement of midpoint between right corner and middle of bottom inner lip	MNS	B	up	2	9
12	raise_l_cornerlip	Vertical displacement of left inner lip corner	MNS	B	up	2	4
13	raise_r_cornerlip	Vertical displacement of right inner lip corner	MNS	B	up	2	5
14	thrust_jaw	Depth displacement of jaw	MNS	U	forward	2	1
15	shift_jaw	Side to side displacement of jaw	MW	B	right	2	1
16	push_b_lip	Depth displacement of bottom middle lip	MNS	B	forward	2	3
17	push_t_lip	Depth displacement of top middle lip	MNS	B	forward	2	2
18	depress_chin	Upward and compressing movement of the chin (like in sadness)	MNS	B	up	2	10
19	close_t_l_eyelid	Vertical displacement of top left eyelid	IRISD	B	down	3	1
20	close_t_r_eyelid	Vertical displacement of top right eyelid	IRISD	B	down	3	2
21	close_b_l_eyelid	Vertical displacement of bottom left eyelid	IRISD	B	up	3	3
22	close_b_r_eyelid	Vertical displacement of bottom right eyelid	IRISD	B	up	3	4
23	yaw_l_eyeball	Horizontal orientation of left eyeball	AU	B	left	3	5
24	yaw_r_eyeball	Horizontal orientation of right eyeball	AU	B	left	3	6
25	pitch_l_eyeball	Vertical orientation of left eyeball	AU	B	down	3	5
26	pitch_r_eyeball	Vertical orientation of right eyeball	AU	B	down	3	6
27	thrust_l_eyeball	Depth displacement of left eyeball	ES	B	forward	3	5
28	thrust_r_eyeball	Depth displacement of right eyeball	ES	B	forward	3	6
29	dilate_l_pupil	Dilation of left pupil	IRISD	B	growing	3	5
30	dilate_r_pupil	Dilation of right pupil	IRISD	B	growing	3	6
31	raise_l_i_eyebrow	Vertical displacement of left inner eyebrow	ENS	B	up	4	1
32	raise_r_i_eyebrow	Vertical displacement of right inner eyebrow	ENS	B	up	4	2
33	raise_l_m_eyebrow	Vertical displacement of left middle eyebrow	ENS	B	up	4	3
34	raise_r_m_eyebrow	Vertical displacement of right middle eyebrow	ENS	B	up	4	4
35	raise_l_o_eyebrow	Vertical displacement of left outer eyebrow	ENS	B	up	4	5
36	raise_r_o_eyebrow	Vertical displacement of right outer eyebrow	ENS	B	up	4	6
37	squeeze_l_eyebrow	Horizontal displacement of left eyebrow	ES	B	right	4	1
38	squeeze_r_eyebrow	Horizontal displacement of right eyebrow	ES	B	left	4	2
39	puff_l_cheek	Horizontal displacement of left cheek	ES	B	left	5	1
40	puff_r_cheek	Horizontal displacement of right cheek	ES	B	right	5	2
41	lift_l_cheek	Vertical displacement of left cheek	ENS	U	up	5	3
42	lift_r_cheek	Vertical displacement of right cheek	ENS	U	up	5	4
43	shift_tongue_tip	Horizontal displacement of tongue tip	MW	B	right	6	1
44	raise_tongue_tip	Vertical displacement of tongue tip	MNS	B	up	6	1
45	thrust_tongue_tip	Depth displacement of tongue tip	MW	B	forward	6	1
46	raise_tongue	Vertical displacement of tongue	MNS	B	up	6	2
47	tongue_roll	Rolling of the tongue into U shape	AU	U	concave upward	6	3, 4
48	head_pitch	Head pitch angle from top of spine	AU	B	down	7	1
49	head_yaw	Head yaw angle from top of spine	AU	B	left	7	1
50	head_roll	Head roll angle from top of spine	AU	B	right	7	1

#	FAP name	FAP description	Units	Uni- or bidirectiona	Position motion	Group	FDP subgroup number
51	lower_t_midlip _o	Vertical top middle outer lip displacement	MNS	B	down	8	1
52	raise_b_midlip_o	Vertical bottom middle outer lip displacement	MNS	B	up	8	2
53	stretch_l_cornerlip_o	Horizontal displacement of left outer lip corner	MW	B	left	8	3
54	stretch_r_cornerlip_o	Horizontal displacement of right outer lip corner	MW	B	right	8	4
55	lower_t_lip_lm _o	Vertical displacement of midpoint between left corner and middle of top outer lip	MNS	B	down	8	5
56	lower_t_lip_rm _o	Vertical displacement of midpoint between right corner and middle of top outer lip	MNS	B	down	8	6
57	raise_b_lip_lm_o	Vertical displacement of midpoint between left corner and middle of bottom outer lip	MNS	B	up	8	7
58	raise_b_lip_rm_o	Vertical displacement of midpoint between right corner and middle of bottom outer lip	MNS	B	up	8	8
59	raise_l_cornerlip_o	Vertical displacement of left outer lip corner	MNS	B	up	8	3
60	raise_r_cornerlip _o	Vertical displacement of right outer lip corner	MNS	B	up	8	4
61	stretch_l_nose	Horizontal displacement of left side of nose	ENS	B	left	9	1
62	stretch_r_nose	Horizontal displacement of right side of nose	ENS	B	right	9	2
63	raise_nose	Vertical displacement of nose tip	ENS	B	up	9	3
64	bend_nose	Horizontal displacement of nose tip	ENS	B	right	9	3
65	raise_l_ear	Vertical displacement of left ear	ENS	B	up	10	1
66	raise_r_ear	Vertical displacement of right ear	ENS	B	up	10	2
67	pull_l_ear	Horizontal displacement of left ear	ENS	B	left	10	3
68	pull_r_ear	Horizontal displacement of right ear	ENS	B	right	10	4

Figure 6.1.6. The list and descriptions of the 68 facial animation parameters (FAPs).

Face Animation Stream:

Coding produces a sequence of frames (FAP Frames) which are meant to be sent as a stream to an animator. One FAP frame carries information for rendering one image on the client side. Thus, to achieve smooth face animation, usually 20~30 FAP frames per second are transmitted. A FAP frame is a sequence of up to 68 integers, each specifying a movement offset for a corresponding FAP.

To conserve bandwidth it is possible to mask out FAP's the values of which are not affected in the transmitted frame. FAP bitmask may be specified either once in the stream header for the entire stream, or it may be included in each frame one 0 or 1 for each of the 68 FAPs indicating whether it is used. Figure 6.1.7 shows an example of a frame with a preceding FAP bitmask. The line of FAP values starts with the frame number followed by the FAP values for the FAPs in use. A zero value for any FAP indicates a neutral face.

```
0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0….
12 -361 311 31 150 15 -145 -175 0
```

Figure 6.1.7. Frame with a FAP bitmask.

Another way to minimise used bandwidth is to omit values for FAPs, that may be calculated from other FAPs. For example: left ↔ right parts of a face, or inner ↔ outer lip contours.

Generation of FAP files may be achieved manually or automatically by employing image processing algorithms which are capable of identifying and tracking facial features.

12. Description of markup language/markup declaration:

FAPs represent a complete set of basic facial actions including head motion, tongue, eye, and mouth control. MPEG-4 includes 68 FAPs which are listed in Figure 6.1.6. These are the concepts used during annotation.

13. Coding files referenced:

Not applicable.

6.2 FACS

1. Name:

Facial Action Coding System – FACS

2. Author(s):

FACS was developed by Paul Ekman and Wallace Friesen at the Langley Porter Neuropsychiatric Institute in San Francisco in 1975.

Paul Ekman
Department of Psychiatry
University of California
San Francisco
Email: ekmansf@itsa.ucsf.edu

3. Version:

Not applicable.

4. Notes:

FACS has no website of its own, but more information on the coding scheme can be found here:

P. Ekman and W. Friesen: Facial Action Coding System. Consulting Psychologists Press, Inc. 1978.

P. Ekman and W. Friesen: Unmasking the Face: A guide to recognize emotions from facial clues. Prentice-Hall, Inc. 1975.

P. Ekman, and E. Rosenberg (Editors): What the face reveals: Basic and Applied Studies of Spontaneous Expression using the Facial Action Coding System (FACS). Oxford University Press, Oxford. 1997.

Ekman, P., Huang, T.S., Sejnowski, T.J. and Hager, J.C.: Report To NSF of the Planning Workshop on Facial Expression Understanding. 1992. Can be found at: http://mambo.ucsc.edu/psl/nsf.txt

In order to recognise the subtle changes of facial expressions, several researchers propose to recognise minimal facial signals and combine the signals to recognise the complete facial expressions. That is, rather than trying to recognise the entire facial expression they are working on recognising singular facial actions. The facial expression is then deduced by combining several facial actions. This recognition method is based on The Facial Action Coding System, FACS. FACS is a system to measure facial signals using minimal action units (AUs). This recognition method follows the same logic as FACS as it looks at singular facial actions.

Cohn developed a facial recognition system composed of three modules. These modules are used to extract information on facial actions. One module extracts information on particular facial features (e.g., brows, mouth), another gets data from larger facial regions (such as chin and cheek) and the final module looks at the appearance of the wrinkles and furrows. The combination of the information provided from the three modules gives good and precise results. More precisely, the feature-point tracking module tracks feature points within a small feature window. The authors also employ a neural network to recognise the action units after the facial features are correctly extracted and suitably represented. Eleven basic lower face action units and combinations (Neutral, AU9, AU 10, AU 12, AU 15, AU 17, AU 20, AU 25, AU 26, AU 27, and AU23+24) and seven basic upper face action units (Neutral, AU1, AU2, AU4, AU5, AU6, AU7) are identified by a single neural network for lower face and upper face separately. The recognition rate results are comparable to the rate obtained by highly

trained FACS coders. The results of recognising single AUs (except for the combination of AU23 + AU24) is more than 95% agreement.

Bartlett has examined several techniques to measure facial actions from the upper face: PCA, optical flow and feature measurement. The authors also combined all three methods (hybrid system) in a single neural network. The four techniques were compared on the same dataset of face image sequences. The dataset was built by asking an actor to perform specific facial actions (corresponding to Action Units as defined in FACS). Each sequence started with the neutral expression. All the faces in the datasets were transformed (scaled, rotated...) so that the facial features of all the faces were aligned. Moreover, the images were cropped in order to contain only the upper part of the face. The best recognition result is obtained for the hybrid system: 92.2%. The results were compared with recognition rates from both naive and expert coders. To both types of coders were given a set of pair of images consisting in the neutral image along with the test image. The neutral image served as a reference basis for the recognition task. Test images contained different faces performing several AUs at low, medium, and high intensity. A training session was given to the naive subjects. Naive coders arrived at 73.7% agreement while expert coders showed 91.8% agreement in their recognition results. The recognition rate obtained with the hybrid method is similar to the one from the expert coders. This is an encouraging result. But as the authors pointed out, the system should be tested on spontaneous expressions. These expressions are often blended expressions, increasing enormously the complexity of the recognition task.

Bartlett, M.S., Hager, J.C., Ekman, P., and Sejnowski, T.J.: Measuring facial expressions by computer image analysis. Journal of Psychophysiology, vol. 36, pp. 253-263, 1999.

Bartlett, M.S., Viola, P. A., Sejnowski, T. J., Golomb, B.A., Larsen, J., Hager, J. C., and Ekman, P.: Classifying facial action. Advances in Neural Information Processing Systems 8, MIT Press, Cambridge, MA. p. 823-829, 1996.

Donato, G.L., Bartlett, M.S., Hager, J.C., Ekman, P., and Sejnowski, T.J.: Classifying Facial Actions. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(10) p. 974-989. 1999.

Kanade, T., Cohn, J.F., and Tian, Y.: Comprehensive Database for Facial Expression Analysis.

5. Purpose of the coding module:

FACS was developed by Paul Ekman and Wallace Friesen in 1975 in order to encode facial expression. Paul Ekman, Wallace Friesen, and S.S. Tomkins had already developed another system called the Facial Affect Scoring Technique (FAST) that is not used any more. FACS is a further development of that technique.

The FACS system is meant to allow expert coders to manually measure facial expressions by breaking them down into component movements of individual facial muscles (Action Units).

6. Coding level(s):

The coding scheme covers the annotation level of facial expression.

7. Description of data source type(s) required for use of the coding module:

Any visual data, image or video.

8. Explanation of references to other coding modules:

None.

9. Coding procedure:

Coders spend approximately 100 hours learning FACS. To use the coding scheme, the coder repeatedly views records of behaviour in slowed and stopped motion to determine which AU or combination of AUs best account for the observed changes.

At least one expert coder should code a corpus. If high reliability of results is important, two expert coders should be used.

10. Coding example showing the markup in use:

Figures 6.2.1 provides a drawn illustration of examples of AUs.

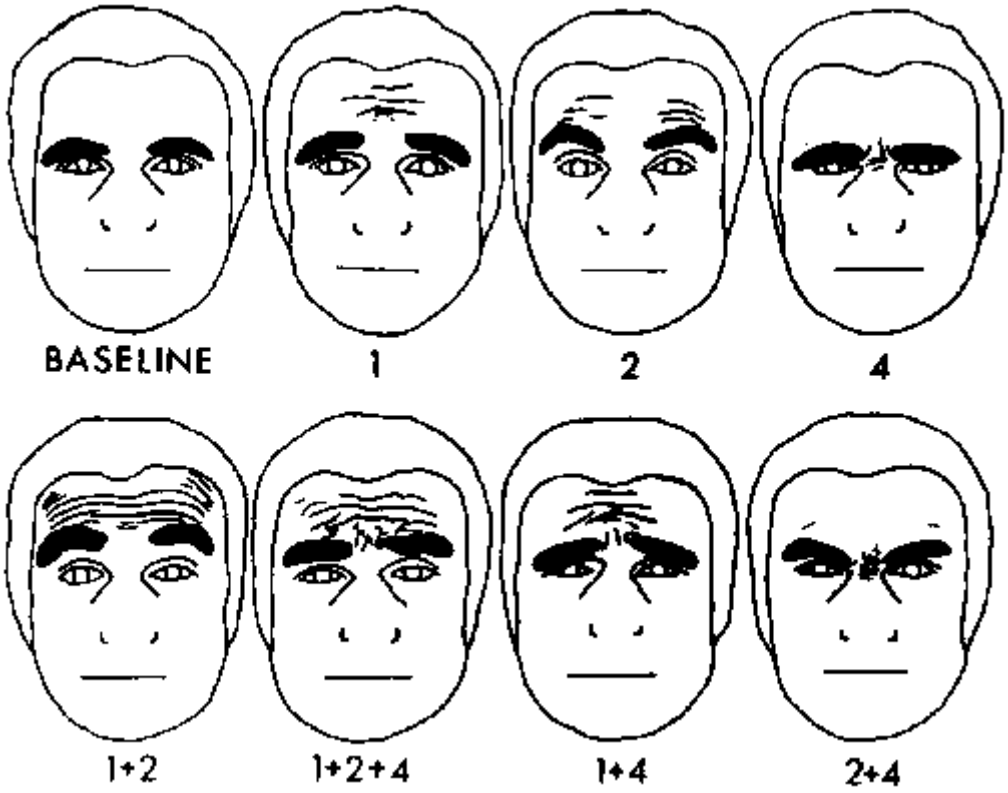

Figure 6.2.1. AU1 corresponds to the raise of the inner brow while AU2 corresponds to the raise of the outer brow. AU4 corresponds to a lowering of the brow. Their combination is the action of both AUs simultaneously. For example, AU1+2 raises the inner and the outer part of the brow.

To perform AU1+2, one should simply raise one's eyebrows. It is an easy action to perform which is not the case of every AU. Indeed, some AUs can be difficult to execute separately and/or consciously; but once they appear in combination or as part of a conversational signal, emotion or any type of spontaneous facial expression, they are easily produced.

Doing the expression of AU1+2, one can notice the following facial changes:

- apparition of wrinkles on the forehead. For some people no wrinkle appears but the skin of the forehead bulges nevertheless. The wrinkles are much more apparent than in neutral position (that is without expression). They are deeper.
- the entire eyebrow is raised.
- the eye cover (it is the part between the eye and the brow) becomes more apparent.
- in case people have heavy eye cover, this action raises it, making apparent their eyelid that usually disappear under their eye cover.

In the description of each AU, minimum requirements for scoring the AU are defined. For AU1+2, the minimum requirements are not simply the sum of both minimum requirements for AU1 and AU2. They correspond to the facial changes explained above but in low intensity. That is, the entire eyebrow is raised slightly with slight apparition of wrinkles on the forehead and/or slight apparition of the eye cover.

Figure 6.2.2 shows a combination of AUs on a human face.

Figure 6.2.2. AUs 10+15. AU-10 raises the upper lip and curves the nasolabial fold (action of levator labii superioris). AU-15 pulls the corners of the lips downwards obliquely (action of triangularis).

For more examples see:

From P. Ekman, W. Friesen: Unmasking the face: A guide to recognizing emotions from facial clues. Prentice-Hall, INC. Englewood Cliffs, New-Jersey, 1975.

11. Clear description of each phenomenon, example(s) of each phenomenon:

FACS involves four operations:

- determining which AUs are responsible for the observed movements.
- scoring the intensity of the actions on a three-point scale: low (X), medium (Y), and high (Z).
- deciding whether an action is asymmetrical or unilateral.
- determining the position of the head and the position of the eyes during a facial movement.

Most of the AUs combine additively. In other cases, rules of dominance, substitution or alternation take over. The dominance rule dictates when an AU disappears for the benefit of another AU. The substitution rule allows for the elimination of certain AUs when others produce the same effects. Finally, the alternation rule takes over when AUs cannot combine. A facial expression is the result of different AUs. Describing a facial expression consists in recognising which AU is responsible for which facial action.

Paul Ekman explains: “A FACS coder "dissects" an observed expression, decomposing it into the specific AUs, which produced the movement. The coder repeatedly views records of behaviour in slowed and stopped motion to determine which AU or combination of AUs best account for the observed changes. The scores for a facial expression consist of the list of AUs, which produced it. The precise duration of each action is also determined, and the intensity of each muscular action and any bilateral asymmetry is rated. In the most elaborate use of FACS, the coder determines the onset (first evidence) of each AU, when the action reaches an apex (asymptote), the end of the apex period when it begins to decline, and when it disappears from the face completely (offset). These time measurements are usually much more costly to obtain than the decision about which AU(s) produced the movement, and in most research only onset and offset have been measured.” The quotation is taken from:

Paul Ekman, Thomas S. Huang, Terrence J. Sejnowski and Joseph C. Hager: Report To NSF of the Planning Workshop on Facial Expression Understanding. 1992. Can be found at: http://mambo.ucsc.edu/psl/nsf.txt

Any visible change on the face (muscle deformation, apparition of wrinkles / bulges, folds), and secondary movements can be annotated by the FACS scheme.

AUs	Name	AU	Name
AU1	Inner Brow Raiser	AU31	Jaw Clencher
AU2	Outer Brow Raiser	AU32	Lip Bite
AU4	Brow Lowerer	AU33	Cheek Blow
AU5	Upper Lid Raiser	AU34	Cheek Puff
AU6	Cheek Raiser, Lid Compressor	AU35	Cheek Suck
AU7	Lid Tightener	AU36	Tongue Bulge
AU8	Lips Toward Each Other	AU37	Lip Wipe
AU9	Nose Wrinkler	AU38	Nostril Dilator
AU10	Upper Lip Raiser	AU39	Nostril Compressor
AU11	Nasolabial Furrow Deepener	AU41	Lip Droop
AU12	Lip Corner Puller	AU42	Slit
AU13	Sharp Lip Puller	AU43	Eyes Closed
AU14	Dimpler	AU44	Squint
AU15	Lip Corner Depressor	AU45	Blink
AU16	Lower Lip Depressor	AU46	Wink
AU17	Chin Raiser	AU51	Head Turn Left
AU18	Lip Pucker	AU52	Head Turn Right
AU19	Tongue Show	AU53	Head Up
AU20	Lip Stretcher	AU54	Head Down
AU21	Neck Tightener	AU55	Head Tilt Left
AU22	Lip Funneler	AU56	Head Tilt Right
AU23	Lip Tightener	AU57	Head Forward
AU24	Lip Presser	AU58	Head Back
AU25	Lips Part	AU61	Eyes Turn Left
AU26	Jaw Drop	AU62	Eyes Turn Right
AU27	Mouth Stretch	AU63	Eyes Up
AU28	Lip Suck	AU64	Eyes Down
AU29	Jaw Thrust	AU65	Walleye
AU30	Jaw Sideways	AU66	Cross-eye

Figure 6.2.3. List of action units (AUs).

12. A markup declaration of the tags for the phenomena which can be marked up using the coding module:

An AU is a minimal visible action. It corresponds to the action of a muscle or a group of related muscles. Each AU describes the direct effect of muscle contraction as well as secondary effects due to movement propagation and the presence of wrinkles and bulges. Available AUs are listed in Figure 6.2.3.

13. Coding files referenced:

Not applicable.

6.3 Toonface

1. Name:

Toonface.

2. Author(s):

ToonFace was created by Kristinn R. Thórisson at the M.I.T.Media Laboratory:
http://www.media.mit.edu/

An extension of ToonFace, CharToon, has been developed at CWI under the supervision of P.J.W ten Hagen as part of the European project Facial Analysis and Synthesis of Expressions (FASE), 1997- 2000. See H. Noot and M. Ruttkay: CharToon 2.0 manual. INS-R0004, ISSN 1386-3681. 2000. Can be downloaded from: http://www.cwi.nl/ftp/CWIreports/INS/INS-R0004.ps.Z.

3. Version:

Not applicable.

4. Notes:

A description of the ToonFace animation framework:
http://xenia.media.mit.edu/~kris/gandalf.html

References to additional information on the coding scheme:

Thórisson, K. R.: ToonFace: A System for Creating and Animating Cartoon Faces. Learning & Common Sense Section Technical Report 1-96. 1996. Can be downloaded from: http://xenia.media.mit.edu/~kris/ftp/toonface.pdf

For more details on FASE, see:

The Facial Analysis and Synthesis of Expression homepage: http://www.cwi.nl/projects/FASE/

The Facial Analysis and Synthesis of Expression Chartoon homepage:
http://www.cwi.nl/projects/FASE/CharToon/

5. Purpose of the coding module:

ToonFace is a coding scheme to code 2D facial expression with limited detail.

6. Coding level(s):

ToonFace can be used to code facial expressions.

7. Description of data source type(s) required for use of the coding module:

The application of the coding scheme produces a text file while the application of this file produces an animated face which requires video rendering.

8. Explanation of references to other coding modules:

None, really. But note the following pre-condition. The coding scheme is meant to define a set of parameters to define and control a 2D facial model. A representation of a face with a neutral expression is therefore required as point of departure.

9. Coding procedure:

Probably is it sufficient to let one expert coder make the encoding and then let another coder check to result by watching the animated face.

10. Coding example showing the markup in use:

ToonFace consist of two parts, an editor and an animation engine or animator. The editor supports markup via a graphical interface. Faces are constructed in the editor by drawing a number of polygons. Figure 6.3.1 shows examples of faces created using Toonface

Figure 6.3.1. Examples of faces created with Toonface.

11. Clear description of each phenomenon, example(s) of each phenomenon:
A face is divided into seven main features: Two eye brows, two eyes, two pupils and a mouth. The eyebrows have three control points each, the eyes and mouth four, and the pupils one each, see also Figure 6.3.2. The ToonFace editor allows manipulation of a face by adjusting control points, cf. entry 12.

12. A markup declaration of the tags for the phenomena which can be marked up using the coding module:
Control points that can be animated are given the codes shown in Figure 6.3.2. These points were selected to maximise the expressiveness/complexity tradeoff. In the case of points that can move in two dimensions, each dimension is denoted as either "h" for horizontal or "v" for vertical.

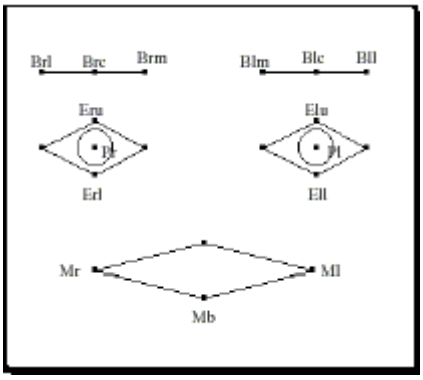

Figure 6.3.2. Codes used for the animated control points.

The following is a complete list of all one-dimensional motors that can be manipulated in a face [control point number in brackets]:
Brl = brow/right/lateral [3]; Brc = brow/right/central [2]; Brm = brow/right/medial [1]
Bll = brow/left/lateral [6]; Blc = brow/left/central [5]; Blm = brow/left/medial [4]
Eru = eye/right/upper [7]; Erl = eye/right/lower [9]
Elu = eye/left/upper [8]; Ell = eye/left/lower [10]
Plh = pupil/right/horizontal [15]; Plv = pupil/left/vertical [15]
Prh = pupil/right/horiz [16-h]; Prv = pupil/right/vertical [16-v]
Mlh = mouth/left/horizontal [14-h]; Mlv = mouth/left/vertical [14-v]
Mrh = mouth/right/horizontal [13-h]; Mrv = mouth/right/vertical [13-v]
Mb = mouth/bottom [12]
Hh = head/horizontal [17-h]; Hv = head/vertical [17-v]
Horizontal motion is coded as 0, vertical as 1. Each of the motors can move a control point between a minimum and a maximum position (for a given dimension). Thus, max and min values mark the limits of movement for each motor. For the eyes and head, these are given in degrees, (0,0) being straight out of the screen; upper left quadrant being (pos, pos), lower left quadrant being (pos, neg)

13. Coding files referenced: Not applicable.

6.4 McNeill's Gesture Coding Scheme

1. Name:

Since there is no official name, we simply refer to the scheme's creator David McNeill.

2. Author(s):

The coding scheme was developed by David McNeill and colleagues and described in McNeill's book *Hand and Mind* (1992).

3. Version:

Not applicable.

4. Notes:

Although this scheme has been used by many researchers, the specific instances may diverge strongly from each other. Some researchers may only use a subset of the scheme, others may have added new features. Still, the original scheme is a good starting point for finding a gesture coding scheme that can be later tailored to fit a specific research aim.

5. Purpose of the coding module:

The scheme is used to transcribe hand and arm gestures for psycholinguistic research. Note that most raw data was of average people trying to retell a cartoon story they had seen before (Sylvester and Tweety). Therefore, the scheme is tailored to cover the many iconic and deictic gestures that usually accompany such a narration. More generic/abstract conversational gestures (e.g. metaphorics) were less in the focus of research.

6. Coding level(s):

The scheme can be used to code manual gestures. The original scheme also suggests as another level the transcription of speech. For brevity, speech transcription is not elaborated here.

7. Description of data source type(s) required for use of the coding module:

The application of the coding scheme produces a text file.

8. Explanation of references to other coding modules:

The temporal location of gestures can be either encoded (1) with reference to time or (2) with reference to speech (words or syllables). In the original form, at a time where coders did not use computers, coders used references to the typed speech transcription to temporally localise gestures. Reference to time is more precise and thus preferable.

9. Coding procedure:

Some time needed to synchronise the coders' interpretations since not all concepts are clearly defined. McNeill (1992:375) emphasises "the great extent to which our procedure relies on discussion". Gesture transcription takes place in the following rough steps:

1. Identify the movements that are gestures.
2. Identify the preparation, stroke and retraction phases of the gestures.
3. Locate the gesture phase boundaries in the relevant part of the speech transcription (better: mark boundaries on a timeline).
4. Locate gesture movements in space. McNeill (1992:378) gives a definition of regions based on concentric circles around the speaker's chest (Figure 6.4.1).

Steps 1 and 2, identification of gestures and their movement phases, can be performed with the procedure described by Kita et al. (1998) which will not be further elaborated here. Step 3 is not necessary when coding gesture boundaries relative to a timeline (absolute time). For step 4, look at Figure 6.4.1. Gesture space will be used at various places in the detailed coding procedure as described under item 12.

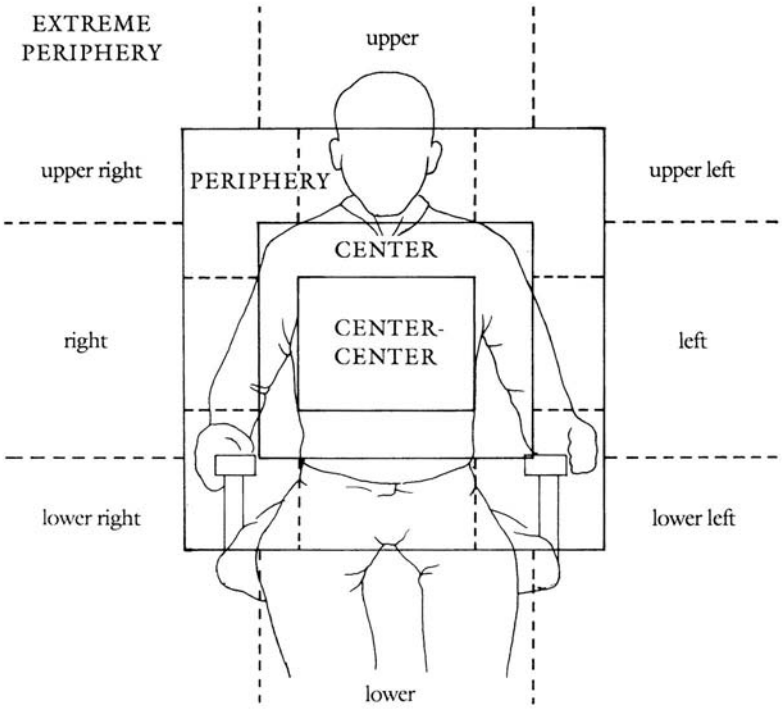

Figure 6.4.1. Segmentation of gesture space, taken from (McNeill 1992:378).

10. Coding example showing the markup in use:

Figure 6.4.2 shows an iconic gesture that has been sample coded by McNeill (1992:382). The gesture was performed while recounting a cartoon with protagonists Sylvester (S) and Tweety Bird (TB).

Figure 6.4.2. Sample iconic gesture, taken from (McNeill 1992:382).

The gesture is coded as follows:

Speech	[so he's **looking** at Tweety Bird]
Gesture type	iconic
Which hand	2SH
Shape	O-hand
Palm/finger orientation	PTC, FAB
Hand place	at eyes

Motion shape and place	no motion
Hand meaning (H)	Sylvester's hands holding binoculars
Motion meaning (M)	no motion
Body meaning (B)	Sylvester
Space meaning (S)	space between Sylvester and Tweety Bird
Gloss	S is looking through binoculars at TB
Viewpoint	C-VPT (Sylvester)
Beat filter	4 (Q1 = yes, Q2 = 2, Q3 = yes, Q4 = nonapplicable)
Confidence	5

11. Clear description of each phenomenon, example(s) of each phenomenon:

See next section.

12. A markup declaration of the tags for the phenomena which can be marked up using the coding module:

The following is a restructured account of the procedure described by McNeill (1992:377-382). It describes how to transcribe a gesture once its temporal boundaries have been identified. On the top level, a gesture is encoded in terms of (1) type, (2) form and (3) meaning:

1. GESTURE TYPE
   Classify the gesture into one of the following (use the beat filter below):
   a. **Representational**: The gesture represents attributes, actions, relationships of objects/characters; subclassify into iconic or metaphoric.
   b. **Deictic**: Finger points or other indications of either concrete or imaginary objects/people.
   c. **Beats**: Formless hands that convey no information but move in rhythmic relationship to speech.

   Also attach a confidence value, i.e. a number reflecting the coder's confidence that the typing is correct, ranging from 1 (unsure) to 5 (certain).
2. FORM
   In case of beats, only code form if it varies from the usual B-shape + up/down strokes.
   a. Hand
      i. handedness
         1. **RH**: right hand
         2. **LH**: left hand
         3. **2SH**: both hands, same shape
         4. **2DH**: both hands, different shape
      ii. hand shape: use ASL[4] (American Sign Language) handform descriptors as depicted in Figure 6.4.3.
      iii. palm/finger orientation
         1. **P/FTU**: palm/finger toward up
         2. **P/FTD**: palm/finger toward down

[4] A table of used ASL descriptors is depicted in Figure 6.4.3, taken from (McNeill 1992:87/88). It is also possible to use another decriptive system for hand shapes like HamNoSys.

3. **P/FAB**: palm/finger away from body
4. **P/FTB**: palm/finger toward body
5. **P/FAC**: palm/finger away from center

iv. place in gesture space

b. Motion

i. shape of motion (trajectory)

1. **TB**: toward body
2. **AB**: away from body
3. **PF**: parallel to front of body
4. **PS**: parallel to side of body

ii. place in gesture space

iii. direction of motion

1. **Uni-1**: unidirectional, one movement
2. **Uni-2**: two movements, second one returning hand to starting place
3. **BiDir**: bidirectional
4. **2SM**: both hands move in same way (mirror)
5. **2DM**: each hand moves own way

3. MEANING
(not coded for beats)

a. Hand

i. what does the hand represent (object, character, feature)?

ii. what viewpoint is the hand representing (**C-VPT** = character viewpoint, **O-VPT** = observer viewpoint, **D-VPT** = dual viewpoint)?

b. Motion

i. what does the motion represent (action, features)?

ii. what viewpoint is the motion representing (character, observer or dual viewpoint)?

The so-called *beat filter* allows one to better identify the gesture type where beats are formless gestures. The following procedure, developed by Bill Eilfort (McNeill 1992:380-82), works on a purely formal level without reference to meaning or function. It is a scoring system where you add 1 for each yes answer to the following questions:

1. Does the gesture have more or less than two movement phases?
2. How many times does wrist/finger movement OR tensed stasis appear (ignore retraction phase, add number to score)?
3. If the first movement is in noncentral space: is any other movement performed in central space?
4. If there are exactly two movement phases: is the first phase in the same place as the second phase?

A score of zero means a beat on formal grounds, higher scores reflect increasing iconicity.

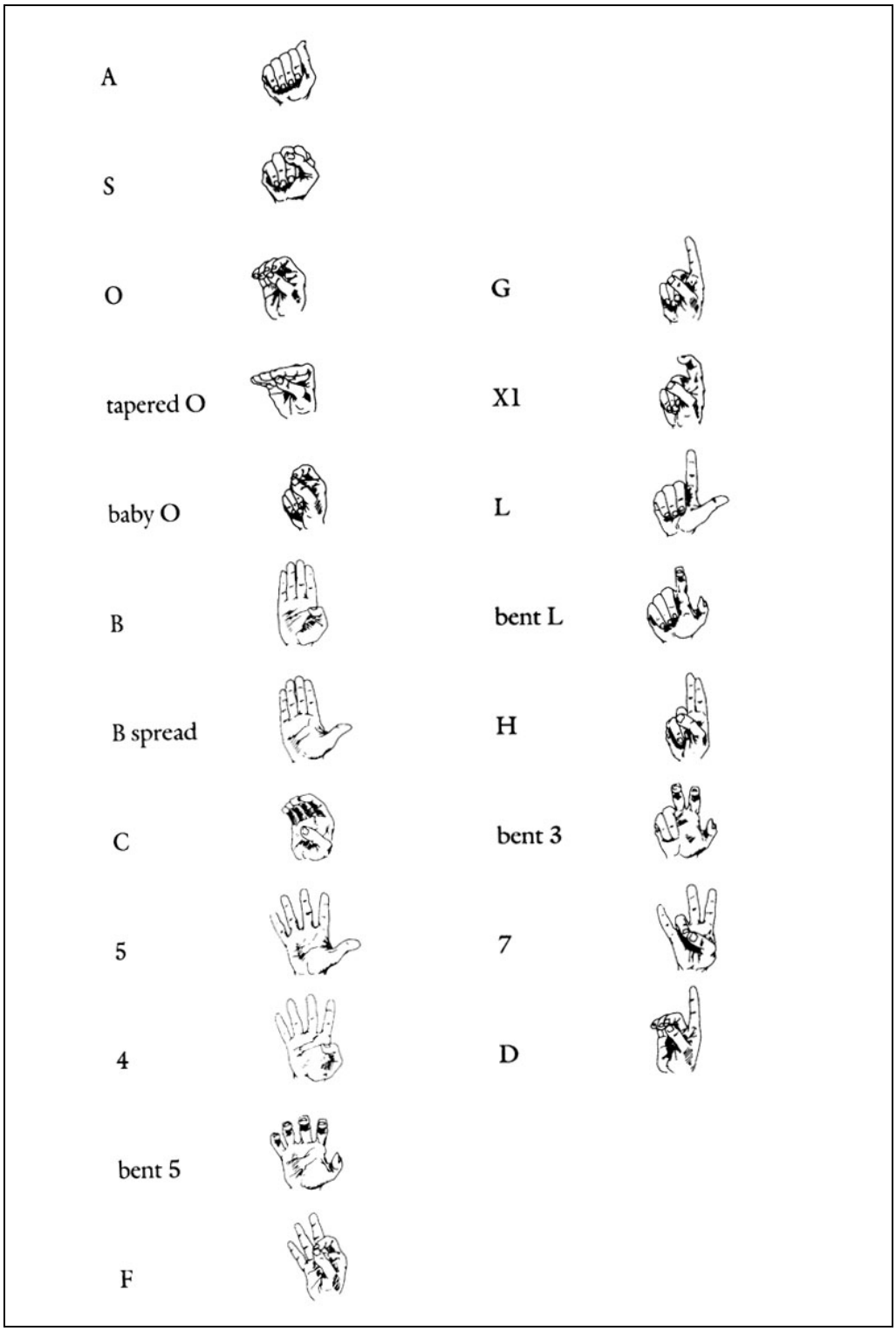

Figure 6.4.3. ASL hand shape codes, taken from (McNeill 1992:87/88).

13. Coding files referenced:
Not applicable.