PRE-PRINT VERSION

to appear in: J. Durand, U. Gut, G. Kristofferson (eds.) Handbook of Corpus Phonology, Oxford University Press.

ANVIL
The Video Annotation Research Tool

Michael Kipp, DFKI, Saarbrücken, Germany

michael.kipp@dfki.de

Abstract: Empirical research often involves three activities: the systematic annotation of audiovisual media (coding), the management of the resulting data in a corpus and various forms of statistical analysis. This chapter presents ANVIL, a highly generic and theory-independent research tool that supports all three activities in conjunction with audio, video and 3D motion capture data. The tool allows to specify a formal coding scheme as a blueprint for project-specific coding. The process of coding is conducted on parallel time-aligned tracks. Instead of simple tags, ANVIL offers typed attributes for describing the basic annotation elements which can be used to hide away complexity and reduce visual clutter. Track types (interval vs. point) and logical track relationships (singleton, span, subdivision) further increase the clarity and consistency of the coding. Spatial mark-up on the video frame allows to annotate regions over time. For corpus management, ANVIL allows to group annotation files into projects for browsing, export and analysis across a corpus of data. For analysis, the tool supports simple descriptive analyses like transition diagrams or label frequency histograms and more complex operations like automatic inter-coder agreement computation (Cohen's kappa).

1 Introduction

Empricial researchers from many diverse areas use video recordings are their primary media for the systematic investigation of, e.g., human communication, animal behavior or human-computer interaction (cf. Rohlfing et al. 2006, Kipp et al. 2009 for overviews). No matter whether the investigation is more of qualitative or more of quantitative nature, a differentiated visualization and a symbolic transcription are prerequisite in these efforts. The topic of adding mark-up or annotations to media extends to a multitude of media types (pictures, video, audio, text, 3D data, biosignals etc.) required in many different fields (linguistics, information extraction, computer animation etc.).

This chapter presents ANVIL[1] (Kipp 2001, 2008, 2010), an annotation tool that is widely used in various research projects, ranging from gesture research to human-computer interaction, from computer animation to oceanography. It is usually used in conjunction with video and allows to systematically and efficiently encode information about events in the video and to view and analyze these encodings. This chapter is meant as an in-depth introduction to ANVIL's underlying concepts which is especially important when comparing it to alternative tools, several of which are described in other chapters of this volume. It also tries to highlight some of the more advanced features (like track types, spatial coding, manual generation) that can significantly increase efficiency and robustness of the coding process.

[1] http://www.anvil-software.de

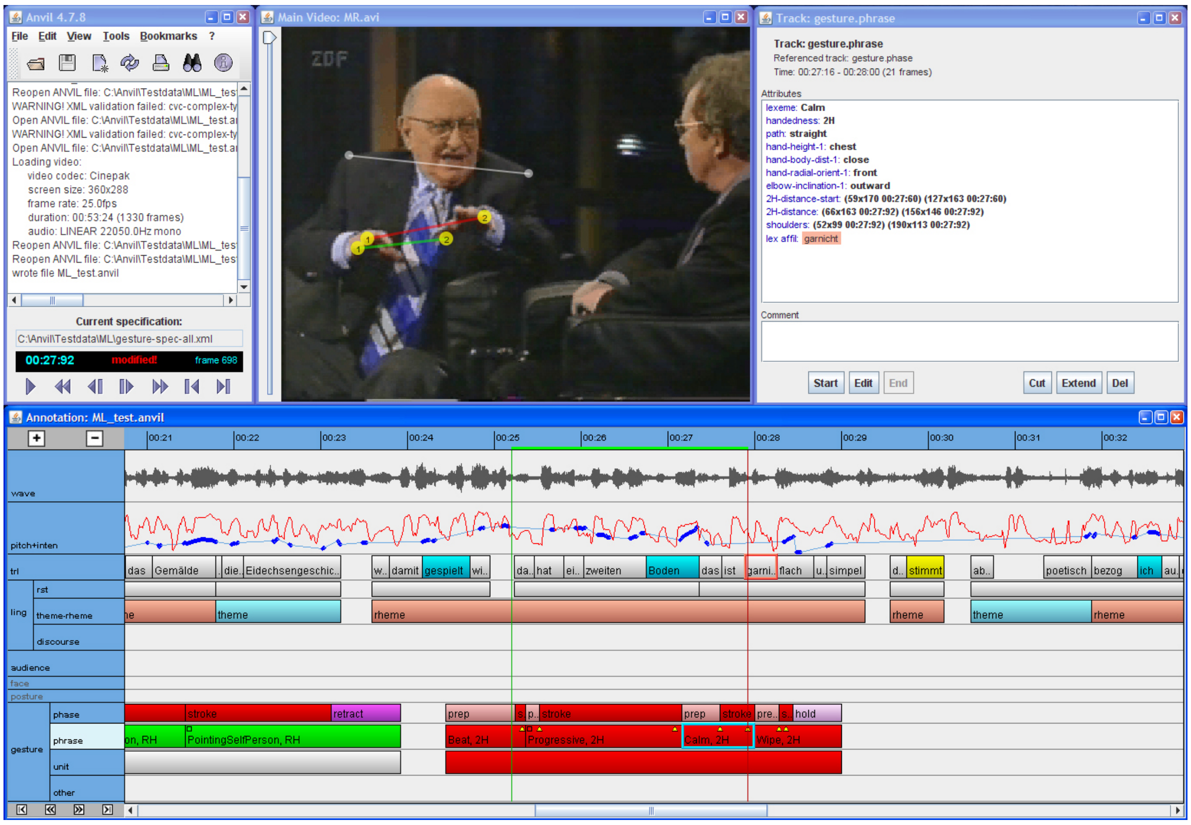
Recently, there have been significant advances in capture technology that make highly accurate data available to the broader research community. Examples are the tracking of face, gaze, hands, body and the recording of articulated full-body motion using motion capture. These data are much more accurate and complete than simple videos that are traditionally used in the field and therefore, will have a lasting impact on multimodality research. However, the richness of the signals and the complexity of the recording process make access to these technologies difficult, especially for non-experts. This is one reason why in ANVIL, a first step in this direction was taken by adding a motion capture viewer. It allows to visualize recordings of human motion with a 3D skeleton that can be freely rotated, thus viewed from all angles.

An annotation tool (like any other tool) has to make a trade-off between the two extremes of flexibility and usability. Extreme flexibility means that the tool could be used in many research contexts by offering the most general concepts, the most powerful search and analysis techniques, and the widest range of features. On the other hand, extreme usability means that the tool is so simple to use that no handbook is required and coding proceeds without effort. Naturally, each tool strikes a balance between the two extremes, a solution that is both completely flexible and highly intuitive is almost impossible to achieve. One consequence is the co-existence of multiple tools, each tool with a slightly different emphasis, with different strengths and weaknesses, as becomes clear nicely in this chapter of the volume. While this diversity may look like a disadvantage at first glance, it can be turned into great benefit if the problem of interoperability can be solved to a satisfying degree, i.e. that data encoded in tool A, can be visualized in tool B, analyzed in tool C, managed in tool D etc. One approach is to suggest a single interchange file format that every tool can import and export. This has been pursued to some degree by an international initiative where ANVIL also was part of (Schmidt et al. 2009). Part of the insight from this effort, however, was the fact that tool differences can become almost impossible to overcome. In consequence, ANVIL now supports the import of ELAN data files and introduced a new track type, called *subdivision*, which mirrors an ELAN functionality, to make these two tools more compatible.

Technically, ANVIL is implemented in Java, including the Java Media Framework (JMF) and Java3D. It therefore runs on multiple platforms (Windows, Mac, Linux) but is limited in the range of video codecs it supports. ANVIL is free for research and educational purposes.

2 Fundamental Annotation Concepts

The general goal of a video annotation tool is to support the process of adding annotations for a particular video (or set of synchronized videos that show the same session from different angles). These annotations usually refer to events observed in the video. Examples for such events in the context of human communication are verbal utterances, gestures or changes in posture. ANVIL belongs to a class of video annotation tools which can be called *tier- or track-based*. Other track-based annotation tools are, for instance, ELAN (Wittenburg et al. 2006), Exmaralda (Schmidt 2004) and PRAAT (Boersma, Weenink 2005). The basic idea is to visualize events as boxes/rectangles on parallel tracks along a horizontal timeline (Figure 1). Different types of events, for instance words and gestures, can be placed on different, parallel tracks so that the temporal relationships between the two modalities can easily be seen. This method of time-aligned tracks is well-known in other domains like movie editing (mixing various film/music tracks), project management (Gantt charts) or sheet music (one instrument per „track").



```
Reopen ANVIL file: C:\Anvil\Testdata\ML\ML_tes
WARNING! XML validation failed: cvc-complex-ty
Open ANVIL file: C:\Anvil\Testdata\ML\ML_test.a
WARNING! XML validation failed: cvc-complex-ty
Open ANVIL file: C:\Anvil\Testdata\ML\ML_test.a
Loading video:
    video codec: Cinepak
    screen size: 360x288
    frame rate: 25.0fps
    duration: 00:53:24 (1330 frames)
    audio: LINEAR 22050.0Hz mono
Reopen ANVIL file: C:\Anvil\Testdata\ML\ML_tes
Reopen ANVIL file: C:\Anvil\Testdata\ML\ML_tes
wrote file ML_test.anvil
```

Current specification: C:\Anvil\Testdata\ML\gesture-spec-all.xml

00:27:92 modified! frame 698

Track: gesture.phrase
Referenced track: gesture.phase
Time: 00:27:16 - 00:28:00 (21 frames)

Attributes
lexeme: Calm
handedness: 2H
path: straight
hand-height-1: chest
hand-body-dist-1: close
hand-radial-orient-1: front
elbow-inclination-1: outward
2H-distance-start: (59x170 00:27:60) (127x163 00:27:60)
2H-distance: (66x163 00:27:92) (156x146 00:27:92)
shoulders: (52x99 00:27:92) (190x113 00:27:92)
lex affil: garnicht

Comment

Figure 1. ANVIL's graphical user interface.

Figure 1 shows the ANVIL user interface. The lower window represents the *annotation board* where horizontal tracks contain the basic *annotation elements*, depicted as boxes that visually represent an event's duration. The human coder continually adds these boxes according to the observed events (e.g. gestures) and fills them with information (e.g. gesture type, handedness, intensity). Note that events do not necessarily have to have a temporal duration (time interval); they could be punctual (time point). For instance, one may want to encode only the peak in the dynamics of a gestural motion, or one may decide to only encode the starting point of events. This is one reason to distinguish different *types* of tracks. The second reason is that a track may *depend* on another track in the sense that elements $x_1 \dots x_n$ of track T are constituents of element y in track T'. In Section 2.3 we discuss ANVIL's various track types.

When encoding information there are various important aspects:

- **Consistency**: make sure that all data sets have the same structure, e.g. the same track names.
- **Validity**: make sure all annotations belong to the „grammar" of valid entries
- **Efficiency**: Adding new annotations should be fast
- **Reliability**: Increase chances that two different annotators would make the same annotation decision at the same point in the video

The issues of consistency made us adhere to a strict separation of coding scheme and coding data. The coding scheme, also called *specification* in Anvil, is the place where the user defines track names and possible content.

2.1 Coding Scheme

The coding scheme is the "blueprint" of any annotations one performs, and this scheme is fundamentally different from the actual coding data that the human coder accumulates in the process of annotation. The scheme defines all the tracks, all the attributes and attribute types that will be filled with information in the process of coding. The coding scheme is decisive as it defines what categories to use. Technically, it is only a syntactic description that the computer can use to tailor the user interface. In contrast, the *semantics* of the scheme is what the human coder has to know in order to assign the correct categories to an event. The coding scheme semantics is usually communicated to the coder in the form of a *coding book* (see manual generation in Section 5.2). However, even on the syntactic level, some choices can positively influence the consistency of later coding, e.g. keeping the number of categories to a necessary minimum and selecting the right types (boolean, number, string) for the attributes.

In short, the coding scheme in ANVIL contains:

- Names of tracks, type of the track and grouping of tracks
- For each track: attributes with type
- For each group: contained tracks and groups

An important design decision with regard to the coding scheme is how to store it. It can be either stored as part of the annotation data, keeping scheme and data in a single place, or it can be stored in a separate file, cleanly separating scheme from data. In ANVIL, a clean separation was chosen. It means that data files (.anvil extension) only contain a link to the coding scheme file (e.g. "myscheme.xml").

The main reason for factoring out the coding scheme in a separate file is the often incremental nature of coding scheme development. This means that a project often starts out with a coding scheme S_1. Coding proceeds by creating data files (.anvil files) $d_1 ... d_k$. If during coding it is discovered that several categories are missing, these can be added to S_1, which becomes the new coding scheme S_2. If the coding scheme is stored in the data files, then all files $d_1 ... d_{k-1}$ have to be corrected. Even if this process is automated, there is significant potential of introducing inconsistencies between the k copies of the coding schemes that reside in $d_1 ... d_k$. Therefore, in ANVIL the data files $d_1 ... d_k$ only point to the coding scheme S_1. If the coding scheme is changed, the data files do not have to be changed at all.

2.2 Annotation Elements as Objects

Annotations can quickly become complex. For instance, to describe a gesture, one may need to encode a number of different properties like handedness, hand shape, trajectory, addressed person, meaning etc. All these data refer to the same entity, a single gesture, so that there is no need to change the temporal interval of the corresponding annotation element. Rather, it makes sense to package all this data into a single element and store the data in various *attributes* of the same element. This is a well-known strategy in programming languages where such entities are know as *objects*. In ANVIL, this technique is consequently applied by introducing typed attributes. "Typed" means that the coder specifies what kind of information is to be stored in the attribute:

- **String**: an alphanumeric string of characters
- **Number(x, y)**: a natural number in the range of {x ... y}
- **Boolean**: a truth value of either TRUE or FALSE
- **ValueSet**: a value from a user-defined set of values {v_1 ... v_N}; this is sometimes called a controlled vocabulary

Attribute types enable ANVIL to automatically offer the more suitable user interface for each attribute (e.g. a number slider for **Number(x,y)**, a check box for **Boolean** or a drop-down menu for **ValueSet**). Note that the set of attributes must be defined and remain constant for a whole track. The track thus defines an area of homogenous data. Since, when looking at existing coding schemes, tracks usually refer to a single "kind" of information (gestures, words, head nods etc.) this seems to be a natural way of conceptualizing tracks.

2.3 Track Types

In ANVIL, most annotations are time-based. Usually, annotation elements have a start and an end time that correspond to time points in the video, making these elements *time intervals*. However, sometimes it makes more sense to annotate elements that only have a single *time point*. Apart from the interval vs. point distinction, there are several types of relationships between pairs of tracks. For instance, elements in track *T* may simply add information to existing elements in another track *T'*, without changing the time information of those elements. An example is a part-of-speech track *T* that encodes the part-of-speech information for every word in the transliteration track *T'*. In such a case, the coder should not be required to enter start/end time for all the existing elements of track *T'* again. Therefore, track *T* can be declared of type *singleton with reference track T'*. Track types establish logical relationships between track elements, which allows their automatic temporal alignment. This greatly increases the robustness of coding and resulting databases. Track types distinguish two dimensions: the first dimension distinguishes whether elements describe a time *interval* or a single time *point*. The second dimension distinguishes whether the track has a logical dependency on another track (the latter is then called the *reference track*).

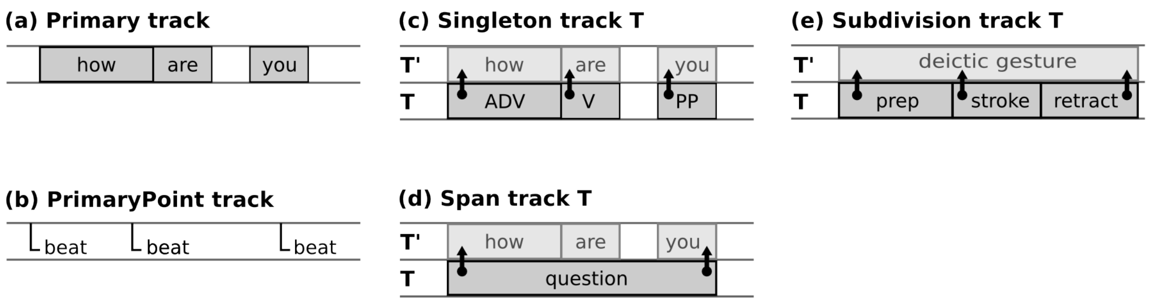


Figure 2. ANVIL's five track types.

2.3.1 Primary track

In a primary track, every element represents a *time interval* with an absolute start time and end time (Fig. 2a). There is no dependency on another track, therefore *primary*. During coding, the coder sets the start time using the green line, whereas the red line marks the end time (Fig. 1). Typical examples of primary tracks are: words in a speech transcription or head movements (nods/shakes).

2.3.2 PrimaryPoint track

PrimaryPoint tracks are similar to a primary track, but the contained elements only represent a single absolute time point (Fig. 2b). This type is used if it is more convenient or intuitive to mark only single time points instead of intervals, e.g. the peak (apex) of a gesture or a beat in a musical tune. Every element represents a single time point and there is no dependency on other tracks.

2.3.3 Singleton track

Singleton tracks contain elements that correspond to *one and only one* other element in the reference track (Fig. 2c). For instance, if primary track T' contains words, then track T could be a singleton track representing the words' part-of-speech (T' is called the *reference track* of T). Note that whether a singleton element represents a time interval or a time point depends and the reference track.

2.3.4 Span track

Span tracks contain elements that *span* a number of contiguous elements in the reference track (Fig. 2d). The coder defines a starting reference element and an ending reference element, and the span element then simply takes of the start time of the first and the end time of the last element. ANVIL will automatically adjust to changes in the reference elements. Note that the reference track can itself be a span or singleton track. This way, it is possible to build a hierarchy of dependent tracks. Note that a span track can only have a reference track that is defined above it. Note that the elements of a span track always contain time intervals, even if the reference track is of type PrimaryPoint.

2.3.5 Subdivision track

In a subdivision track T one can "split" elements of the reference track T' into smaller units (Fig. 2e). For instance, if one encodes whole gestures in track T' and one wants to split a gesture into more primitive *movement phases* (Kita et al. 1998) in track T, this works as follows: Track T is declared a subdivision track with reference track T'. For each gesture in T' one can split it into multiple compartments to accommodate the phases. In a way, this is the inverse track relationship of *span* (see Section 2.3.4 above). A subtle difference is that a span relationship allows gaps between elements whereas a subdivision relationship does not.

3 Advanced Annotation Concepts

3.1 Nontemporal elements and annotation sets

When coding video one naturally thinks in terms of time events. However, at times it is necessary to encode nontemporal entities like objects in a room or a person. Nontemporal entities can be either concrete (object in the room) but not changing for the whole video, or they can be abstract (a discourse topic). For ANVIL, such elements are simply annotation elements without timestamps (Martin, Kipp 2002).

The term "track" implies some temporal succession of subelements that is not given any more when removing timestamps. We therefore call an *annotation set* an entity that contains nontemporal elements. Technically, sets and tracks are subclasses of the same concept called *container*. Also, the elements of a track and the nontemporal elements of a set are subclasses of the same abstract element class. Because of these equivalences it is straightforward to use cross-level links (see next Section 3.2) to refer between elements of tracks and sets.

3.2 Cross-level Coding

The whole point of multi-track annotation lies in finding relationships between different tracks, which often represent different modalities like gaze, gesture or speech. While one line of analysis is to look at recurring temporal relations (co-occurrence, overlap, sequence) or patterns (e.g. t-patterns), in some domains one needs to encode the relationship between elements of different tracks explicitly (e.g. if a word and a gesture express the same meaning but do not exactly temporally co-occur). In ANVIL, the coder can insert so-called cross-level links into elements of a track. Thus, s/he can express that e.g. a particular word „belongs“ to a gesture (Kipp et al. 2007a). It is realized through a special type of attribute called "MultiLink" that can contain an arbitrary sequence of logical links to any other element on the annotation board. Note that the more general solution is to have a collection of abstract relation objects that take two or more arguments. This would have made it simple to encode symmetric relationships. However, the first solution is easier to understand and to use in practice. For symmetric or reciprocal relationships, ANVIL can automatically insert backlinks in the target element. The coder can also restrict the links' containing track by specifying it in the type, e.g. "MultiLink(words)" restricts the links to come from the "words" track.

3.3 Spatial Coding

Most annotation tools allow the encoding of time-anchored elements, either as intervals (start/end time) or time points. Since there is no information referring to *space*, the annotation element refers to the whole video frame or rather the sequence of frames in that time interval. However, the coder may want to be more specific about the spatial region that the annotation refers to, e.g., a building or a face in the upper left corner of the frame. If this feature is moving, due to own movement or camera movement, the coder wants the annotation to correctly follow the feature for each video frame that is contained in the annotation element's interval. This can be done by specifying key locations and interpolating in-between like in computer animation. In ANVIL, spatial coding can be done by defining a special attribute, which then stores screen locations (Kipp 2008). The graphical user interface allows to interactively define these locations by clicking on the video screen. For the moment, only points can be defined. For the future, rectangles, circles and other shapes will be added. The two new attribute types are "Points" and "TimestampedPoints". The "Points" attribute contains a sequence of 2D screen coordinates that refer to the video screen pixels in x- and y-dimensions. The "TimestampedPoints" attribute additionally has a timestamp attached to each location.

4 Interface Matters

For annotation tools the user interface is of crucial importance, mainly for the two reasons of *efficiency* and *exploration*. Efficiency is important because for quantitative analysis huge amounts of data have to be annotated. Therefore small differences in interface efficiency can have a significant impact on the amount of time spent on annotation. To maximize efficiency, one has to minimize the coding effort (reduce number of user actions, e.g. by offering short-cuts), minimize the chance of errors (reliability), while making sure that the tool usage can be easily learned (learnability). On the other hand, for exploration it is important that certain relationships between encoded elements can easily be discerned in a qualitative analysis. For the latter aspect ANVIL has always relied on a realtime timeline view, i.e. at any time during coding all elements are displayed in temporal relationship with each other.

4.1 Visualization and control

Multimodal annotation can become a complex activity with much more information than a single glance can process. Therefore, the annotation board becomes a valuable visual real estate that must not be wasted with unimportant information. ANVIL gives coder a number of techniques to carefully manage this visual real estate: elements as objects, attribute display rules, grouping/hiding tracks, color-coding attributes, and elements as objects.

First of all, even the most basic element in coding can be quite information-rich elements with several attribute-value pairs. Coder can thus annotate complex information, which is nicely packaged away in the elements. Using display rules, the coder can control which parts of the annotation to make visible on the annotation board (Fig. 1). Moreover, the coder can decide to visualize a single attribute using *color-coding* where one color is defined for each of the possible values of an attribute. Moreover, tracks can be organized in groups. A group is a folder-like concept that can in turn contain groups. Thus, one can define a group for "speaker_A" with subgroups "speech" and "gesture" which then contain actual tracks. To reference such nested tracks ANVIL uses dot-separated path notation, e.g. "speaker_A.gesture.phase". In the user interface, groups can be hidden which enables the coder to focus on the remaining tracks. Coders can also hide and show any combination of tracks using the "hide track" function.

An important issue is how fast an annotation can be performed. For this, ANVIL offers both general and user-defined keyboard short cuts. The general short cuts refer to positioning the start line (F1) or to create a new, empty element (F2). The user-defined short-cuts refer to the setting of an attribute (of type ValueSet) with a particular value. The short cuts are defined in the coding scheme.

From our experience, using short-cuts can speed up coding by a factor of up to 10. To give an example: a coder can first run trough a video using only the two keys **F1** (set start line) and **F2** (create element) to create empty annotation elements for a track T, to annotate say head movements. In a second run, the coder could have defined keyboard short-cuts for the attribute "motion type" consisting of nods (key **H**), shake (key **S**) and tilt (key **T**) - the coder then uses these keys to fill the empty elements. This is much faster than opening the input dialog window each time one needs to enter information.

4.2 Top-down vs. bottom-up coding

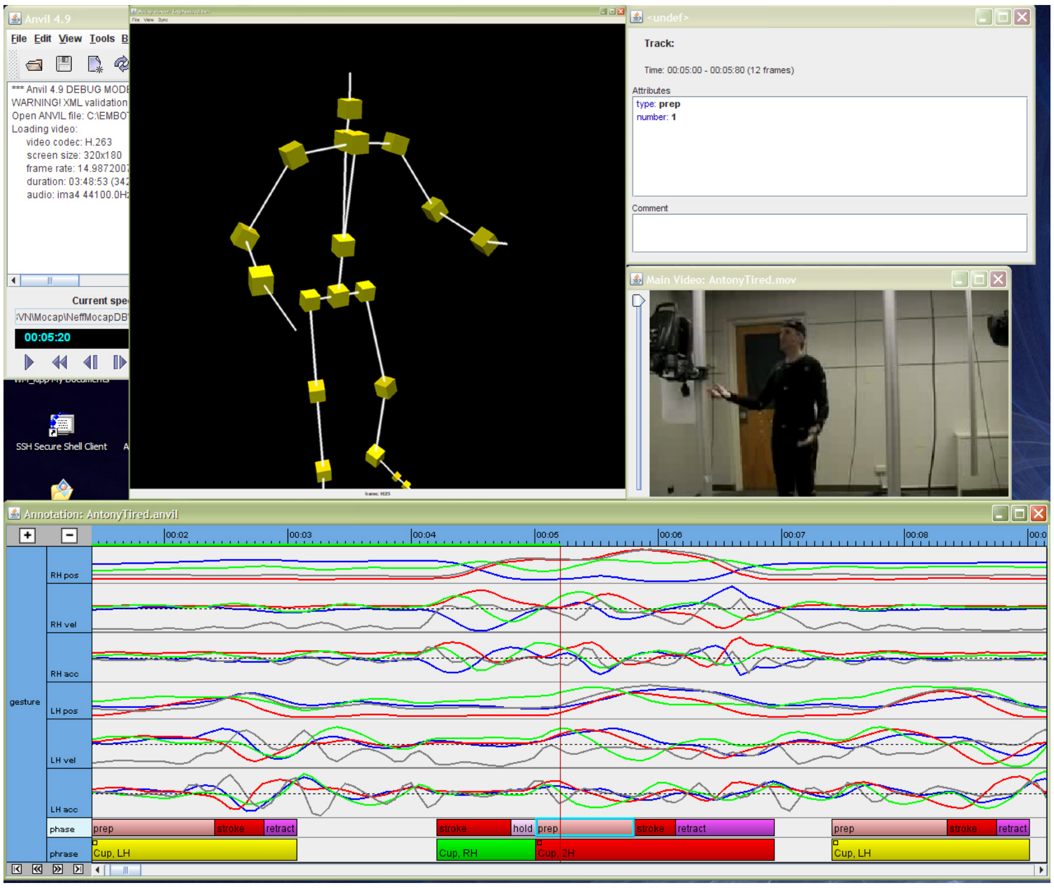
ANVIL supports various annotation procedures: top-down or bottom-up coding. Top-down coding means that a coder first annotates more high-level concepts (i.e. longer time intervals), which are then broken up into smaller parts. In bottom-up coding the coder first encodes the atomic units of, e.g., a behavior and then proceeds with higher-level entities that connects these atomic units. For instance, when coding gestures, a coder can start by coding the gesture constituents, so-called gesture phases (preparation, stroke, hold, retraction) and then encode the actual gesture. This can be called bottom-up coding. Alternatively, she can start by encoding the entire gesture and then, break up the gesture into gesture phase, what can be called a top-down approach.

Technically, a top-down approach would use the *subdivision* relationship: track A, a primary track, would encode the gestures, whereas track B, a subdivision track with reference track A, would encode phases. In contrast, a bottom-up approach would utilize the *span* relationship: track A, a primary track, would encode gesture phases, while track B, a span track with reference track A, would encode gestures by connecting constituting phases of track A.

4.3 Media Viewers

ANVIL's primary media consists of video. Often, video recordings of e.g. human behavior involve the use of multiple cameras at different viewpoints. Therefore, ANVIL offers to open an arbitrary number of videos for synchronized playback. A temporal offset can be specified if videos do not start at the same time point.

Additionally, ANVIL features an integrated 3D motion capture (mocap) data viewer using a 3D stick figure to visualize skeletal motion. As motion capture is becoming more affordable, such technology is becoming more likely to be employed in human behavior analysis (Fikkert et al. 2008). The 3D viewer is opened alongside video viewers and can be synchronized with video playback as well. ANVIL can read the two most widely used motion capture data formats: BVH and ASF/AMC. Both formats store the skeleton topology and naming in one part and the actual joint angle data per time frame in a second part. For analysis, ANVIL allows to display motion curves of e.g. the wrist joint's absolute position in space, their velocity and acceleration on special tracks (Fig. 3).


```
*** Anvil 4.9 DEBUG MODE
WARNING! XML validation
Open ANVIL file: C:\EMBOT
Loading video:
    video codec: H.263
    screen size: 320x180
    frame rate: 14.9872007
    duration: 03:48:53 (342
    audio: ima4 44100.0Hz
```

Track:

Time: 00:05:00 - 00:05:80 (12 frames)

Attributes

type: **prep**
number: **1**

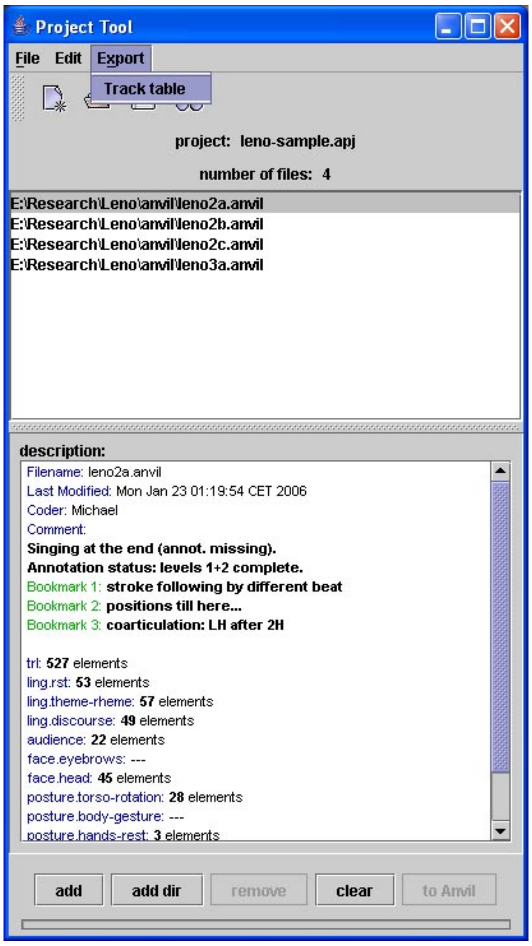
Comment

Figure 3. 3D motion capture data can be visualized using a 3D stick figure animated in realtime.

5 Corpus Management

5.1 Projects and the Project Tool

The *project tool* allows declaring a number of annotation files (.anvil files) as a *project*. The only condition is that all files point to one and only one coding scheme file. Once a project is created, one can search and export over the whole set of files. One can also conveniently browse the annotation files, since the project tool shows meta-data and the number of annotated elements for each track at a single glance. In the Figure X one can see the list of annotation files in the top half, whereas the lower half shows information about the selected file.



project: leno-sample.apj

number of files: 4

```
E:\Research\Leno\anvil\leno2a.anvil
E:\Research\Leno\anvil\leno2b.anvil
E:\Research\Leno\anvil\leno2c.anvil
E:\Research\Leno\anvil\leno3a.anvil
```

description:

```
Filename: leno2a.anvil
Last Modified: Mon Jan 23 01:19:54 CET 2006
Coder: Michael
Comment:
Singing at the end (annot. missing).
Annotation status: levels 1+2 complete.
Bookmark 1: stroke following by different beat
Bookmark 2: positions till here...
Bookmark 3: coarticulation: LH after 2H

trl: 527 elements
ling.rst: 53 elements
ling.theme-rheme: 57 elements
ling.discourse: 49 elements
audience: 22 elements
face.eyebrows: ---
face.head: 45 elements
posture.torso-rotation: 28 elements
posture.body-gesture: ---
posture.hands-rest: 3 elements
```

Figure 4. ANVIL project tool.

5.2 Coding Manual Generation

For the creation of a consistently annotated corpus is it essential that the coding scheme be defined as clearly as possible. Putting such definitions in words, i.e. writing a *coding manual*, can be painstaking work. Anvil alleviates this task by exploiting the fact that the structure of the coding manual is exactly the same as the one already defined in the coding scheme specification file: Here, the whole scheme, the groups, the tracks, the attributes are specified in XML in a hierarchical fashion. By inserting definitions and descriptions into the specification file (bracketed by <doc> tags) one can let ANVIL automatically generate a suite of hierarchically organized HTML pages[2] where descriptions of scheme, groups, tracks, attributes and attribute values can be accessed by clicking on the respective names. The advantage of HTML being that one can open the electronic manual on-screen while in the process of annotation using a web browser, one can also use regular HTML markup for formatting options of for integrating images.

ANVIL also uses the <doc> tags to generate online help windows for each attribute that can be opened by clicking on a small question mark that resides in the input dialog during annotation. Thus, the coder has multiple options to access the coding book.

6 Analysis

6.1 Descriptive Stats

If T_{video} is the duration of the video, ANVIL displays a short summary of every track with the following values:

- N = number of annotation elements
- $T_{elements}$ = total duration of all elements
- number of elements per sec/min = T_{video} / N
- number of elements per covered time = $T_{elements}$ / N

Another important information is how often different categories actually occur in the data. In ANVIL, categorical data is usually coded using the ValueSet type where a set of labels can be defined. The "histogram" function shows the distribution of labels for one attribute (either for the current media or for the whole corpus).

6.2 Transition Diagrams

Transition diagrams visualize how categories transition into one another when looking at the whole corpus. The diagram consists of states and transitions (Fig. 5). Each transition has a probability attached to it (here, given in percent). All transitions coming from the same state must add up to 100%. A transition with 21% between state A and state B means that in 21% of all times that the system was in state A, the immediately following state happened to be B. Transition diagrams visualize the temporal neighborhood of discrete events in a quantitative fashion. For example in Fig. 5, a stream of gestures is analyzed in terms of how often the speaker changes from left hand (LH) to right hand (RH) to two hands (2H), in all possible combinations.

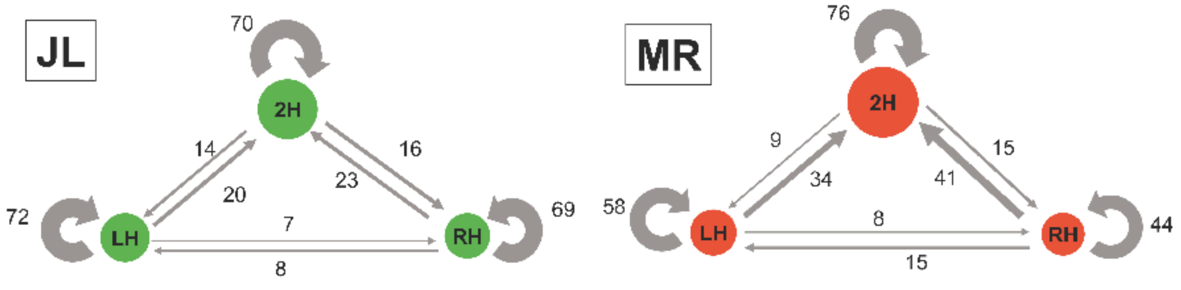


Figure 5. Handedness transition diagrams for speakers JL and MR.

[2] For Java programmers: The pages have the look-and-feel of Javadoc pages.

Transition diagrams give an immediate visualization of the conditional probabilities of being in one state given that another state preceded it. The example of gesture handedness in Fig. 5 was used by Kipp et al. (2007b) and Neff et al. (2008) to detect and model idiosyncrasies in gesture behavior for two distinct speakers.

6.3 Inter-Coder Agreement

Manual annotation often involves a high degree of human interpretation. Therefore, it is essential to measure how "valid" the annotations are by conducting inter-coder or intra-coder agreement studies. This means that multiple coders annotate the same media (inter-coder) or the same coder annotates the same media after some time has passed (intra-coder). In both cases, the degree of correspondence between two annotation files has to be measured. ANVIL can compute Cohen's kappa as one such measure. This statistic is appropriate for testing whether agreement exceeds chance levels for binary and nominal ratings (Cohen 1960). The input consists of two annotation files (or two sets of files) to compare. The user must decide which track and which attribute to analyze. In the computation of kappa, the elements of two tracks are compared where each element has one out of *n* behavior categories $C_1, \ldots, C_n$. The confusion matrix records the occurrences of paired elements in terms of categories. The diagonal in the matrix is the number of occurrences of agreement between the two coders for each behavior. This matrix is also informative for understanding possible sources of disagreement. However, the challenge is to decide which elements to compare in cases where the segmentation is different. In ANVIL, this problem is solved by considering time slices instead of elements. For videos with a frame rate of 25 frames per second, ANVIL cuts the annotation file into slices of .04 sec and compares categories on each time slice, adding one additional category VOID for the case that no annotation resides on the slice. These counts are put into a confusion matrix used to compute kappa.

For every performed agreement analysis, ANVIL displays the confusion matrix and the computed kappa. The resulting kappa value can be used to get an impression of how consistent the annotation is. Fleiss (1981) considers a kappa between 0.40 and 0.60 as fair, between 0.60 and 0.75 as good and over 0.75 as excellent. A satisfactory kappa must still be taken with care. It is still preferable to computing, e.g., the simple percentage of agreement because the latter does not factor out chance agreement.

7 Interoperability

ANVIL supports various import/export formats to allow a workflow involving multiple tools.

7.1 Import from ELAN

Comparing ANVIL with competing tools currently available, the ELAN software (Wittenburg et al. 2006) is the most similar in functionality and also in terms of user base. For instance, a large number of gesture and sign language researchers use either ELAN or ANVIL. These tools share a number of features but also each have their unique strengths (e.g. ELAN's powerful search functionality or ANVIL's agreement analysis). While the vision of having a common interchange format that is understood by all tools is still an ideal worthwhile pursuing (Schmidt et al. 2009), in the short-term a direct tool-to-tool translation is more feasible and allows treating the subtle similarities and differences between two particular tools with more detail. Therefore, an ELAN *import* feature was integrated in ANVIL, to be complemented by an ELAN export in the future.

What is called a track in ANVIL is called a *tier* in ELAN. Every ELAN tier has a *linguistic type* that corresponds to a track type in ANVIL. This type also defines a certain relationship between a tier *T* and a reference tier *T'* (in ELAN, *T'* is called the *parent tier* of *T*). When importing an ELAN data file, every ELAN tier is mapped to an ANVIL track, converting the tier's linguistic type to a corresponding ANVIL track type. The following list gives a short description of the ELAN tier type and how it is converted to ANVIL:

- **NONE**: This ELAN tier has no dependency on another tier. It is mapped to an ANVIL *primary track*.
- **Included-In**: In ELAN, this means that very element in tier *T* must be located within the temporal limits (therefore "included") of an existing element in parent tier *T'*. Since ANVIL does not have this type of relationship, tier T is converted to an ANVIL *primary track*. The information of temporal inclusion between T and T' is lost.
- **Time-Subdivision**: In ELAN, this means that every element in parent tier *T'* is decomposed/partitioned by several elements in tier *T*. In other words, the elements of tier *T* are constituents of elements in *T'*. This is mapped to ANVIL's equivalent *subdivision track*.
- **Symbolic-Subdivision**: This type corresponds to *Time-Subdivision* with the difference that the division of elements in *T'* are not rooted in time but simply by having a certain number *N* of subdivisions. The visualization of the subdivisions is usually based on equidistant time subdivision. Since ANVIL does not offer this type, the tier is mapped to ANVIL's *subdivision track*; note that artificial timestamps have to be created in ANVIL.
- **Symbolic-Association**: In ELAN, this means that every element in *T* corresponds to exactly one element in parent tier *T'*. In ANVIL, this is mapped to the equivalent *singleton track*.

7.2 Export

7.2.1 Text tables

Although ANVIL offers some analysis facilities, for most flexibility and power, annotation data must be exported to a format that can be read by standard statistics tools like SPSS and Statistica. While of these tools can read ASCII tables, it is essential how the data in a table are organized. Therefore, ANVIL offers different structural formats for ASCII table export:

- Element-by-attribute matrix
- Framenumber-by-attribute matrix

In the element-by-attribute matrix, each row represents an annotation element (e.g. the encoding of a single gesture) while the columns contain the element's start/end time and all attributes (e.g. gesture type, handedness, shape etc.). This table may contain several tracks, resulting in a sparse table where, in each row, only those columns are filled which represent the current track's attributes. In this representation, temporal relations are difficult to reconstruct in analysis. Even if elements are ordered chronologically, neighboring rows need not overlap or even reside in temporal proximity. Therefore, this kind of output is problematic if one wants to analyze temporal relationships in standard statistics tools which cannot infer which elements co-occur, overlap etc. In the second format, the framenumber-by-attribute matrix, each row represents a single video frame like a cross-section at each time point. Columns are similar like in the first format, i.e. they represent attributes. In this table, everything represented in a row occurs at the same time, facilitating statistical analysis of co-occurrence.

Since SPSS can only handle values represented as numbers, ANVIL allows exporting a table using numbers as codes, including a cross-reference file with the meaning of each number.

7.2.2 Exporting to the WEKA machine learning and knowledge discovery tool

WEKA[3] (Waikato Environment for Knowledge Analysis) is a free machine learning and knowledge discovery workbench developed at the University of Waikato. It features a number of standard machine learning algorithms, together with powerful filtering and analysis facilities. WEKA's input file format is called ARFF (Attribute-Relation File Format). ANVIL can export a single track/set from a single data file or a whole project to ARFF format.

The translation from a track/set to ARFF data file is straightforward. The following concepts are mapped:

- An ANVIL *track* or *set* becomes an ARFF *relation*
- Each ANVIL *attribute* becomes an ARFF *attribute*. Note that only certain ANVIL attribute types are mapped (Number, ValueSet, Boolean) and all others are ignored (String, Link etc.)
- Each ANVIL *annotation element* becomes an *instance* in ARFF

8 Conclusions

This chapter presented ANVIL, a highly generic annotation tool, that covers the three activities of coding, corpus management and analysis, in conjunction with audio, video and motion capture data. Coding is based on the specification of a formal coding scheme as a blueprint for project-specific coding. ANVIL uses parallel time-aligned tracks for visualization and offers typed attributes for describing the basic annotation elements. Five different track types (Primary, PrimaryPoint, Singleton, Span and Subdivision) allow for both flexible and robust coding. ANVIL offers spatial mark-up on the video frame and for corpus management, the project tool allows to group annotation files into projects for browsing, export and analysis across a corpus of data. For analysis, the tool supports simple descriptive analyses like transition diagrams or label frequency histograms and more complex operations like automatic inter-coder agreement computation.

There are a number of tools that share the same goals as ANVIL, many of which are described in other chapters of this volume. However, ANVIL has a number of unique characteristics. Most importantly, ANVIL is the only tool that treats annotations as objects with typed attributes, making annotations much more compact. It is also the only tool that keeps the coding scheme strictly separated from the annotated data, which has proven to be a major advantage in the iterative development of coding schemes. Another unique feature are symbolic links, an essential tool when investigating cross-modal relationships. ANVIL is also the only tool that allows the encoding of spatial information on the video frame, important for preparing information extraction training

[3] http://www.cs.waikato.ac.nz/~ml

material, and the only tool to offer a fully 3D motion capture viewer. ANVIL shares with the ELAN tool the use of track relationships to make coding more robust and with the EXMARaLDA tool a dedicated corpus management tool. For the future, it is be desirable to increase interoperability between tools so that end users can exploit the individual strengths of various tools on the same data.

Acknowledgements

Part of this research has been carried out within the framework of the Cluster of Excellence Multimodal Computing and Interaction (MMCI), sponsored by the German Research Foundation (DFG).

References

Boersma, P. and Weenink, D. (2005) *Praat: doing phonetics by computer* (version 4.3.14) [computer program]. Retrieved from http://www.praat.org/.

Cohen, J. A. (1960) A coefficient of agreement for nominal scales. In: *Educational and Psychological Measurement*, 20, pp. 37-46.

Fikkert, W., van der Kooij, H. Ruttkay, Z. and van Welbergen, H. (2008) Measuring behavior using motion capture symposium. In: *Proceedings of Measuring Behavior*.

Fleiss, J. L., Levin, B., & Paik, M. C. (2004) *Statistical methods for rates and proportions*. New York: Wiley.

Kipp, M. (2010) Multimedia Annotation, Querying and Analysis in ANVIL. In: M. Maybury (ed.) *Multimedia Information Extraction*, Chapter 21. MIT Press, *to appear*.

Kipp, M., Martin, J.-C., Paggio, P., und Heylen, D. (eds.). (2009) Multimodal Corpora: From Models of Natural Interaction to Systems and Applications, Lecture Notes on Aritificial Intelligence, LNAI 5509, Springer.

Kipp, M. (2008) Spatiotemporal coding in ANVIL. In: *Proceedings of the 6th international conference on Language Resources and Evaluation (LREC).*

Kipp, M. (2001) Anvil – a Generic Annotation Tool for Multimodal Dialogue. In: *Proceedings of Eurospeech*, pp. 1367–1370.

Kipp, M., Neff, M. and Albrecht, I. (2007a) An Annotation Scheme for Conversational Gestures: How to economically capture timing and form. In: *Journal on Language Resources and Evaluation - Special Issue on Multimodal Corpora*, 41(3-4), pp. 325–339.

Kipp, M., Neff, M., Kipp, K. H. and Albrecht, I. (2007b) Toward Natural Gesture Synthesis: Evaluating gesture units in a data-driven approach. In: *Proc. of the 7th International Conference on Intelligent Virtual Agents (IVA-07)*, pp. 15–28. Springer.

Kita, S., van Gijn, I. and van der Hulst, H. (1998) Movement phases in signs and co-speech gestures, and their transcription by human coders. In: Wachsmuth, I. and Fröhlich, M. (eds.) *Gesture and Sign Language in Human-Computer Interaction*, Springer, pp. 23–35.

Martin, J.-C. and Kipp, M. (2002) Annotating and Measuring Multimodal Behaviour – Tycoon Metrics in the Anvil Tool. In: *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pp. 31–35.

Rohlfing, K., Loehr, D., Duncan, S., Brown, A., Franklin, A., Kimbara, I., Milde, J.-T., Parrill, F., Rose, T., Schmidt, T., Sloetjes, H., Thies, A. and Wellinghoff, S. (2006) Comparison of multimodal annotation tools — workshop report. In: *Gesprächsforschung*, 7, pp. 99–123.

Schmidt, T. (2004) Transcribing and annotating spoken language with Exmaralda. In: *Proceedings of the LREC-Workshop on XML based richly annotated corpora*.

Schmidt, T., Ehmer, O., Hoyt, J., Kipp, M., Loehr, D., Rose, T., Sloetjes, H., Duncan, S., and Magnusson, M. (2009) An exchange format for multimodal annotations. In: *Multimodal Corpora: From Models of Natural Interaction to Systems and Applications,* Lecture Notes on Aritificial Intelligence, LNAI 5509, Springer.

Wittenburg, P., Brugman, H., Russel, A., Klassmann, A. and Sloetjes, H. (2006) ELAN: A professional framework for multimodality research. In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC).*