Visualization Methods for the VITRA Workbench

Gerd Herzog
SFB 314, Project VITRA
Universität des Saarlandes
D-66041 Saarbrücken, Germany
herzog@acm.org

Abstract

The project *Vitra* (VIsual TRAnslator) deals with the relationship between natural language and vision. Experimental studies are being carried out in the way of designing an interface between image-understanding and natural language systems, with the aim of developing systems for the natural language description of image sequences.

Together with the vision group at the IITB, Karlsruhe, first results in connecting a vision component and a natural language access system have been obtained. These previous attempts have been restricted to a bird's eye view and thus 2-dimensional representation of the scenes under discussion. Within the new *Vitra* system the methods developed so far will be extended in order to cope with 3-dimensional visual information.

In this contribution[1] the capabilities of the *Vitra* workbench concerning 3-dimensional geometric representations will be presented. The paper will concentrate on the visualization of the geometric data as well as on the representation and manipulation of the underlying image sequences.

This paper has been published as: Memo 53, Universität des Saarlandes, SFB 314 (VITRA), December 1992.

[1]The work described here was partly supported by the Sonderforschungsbereich 314 der Deutschen Forschungsgemeinschaft, "Künstliche Intelligenz und wissensbasierte Systeme" project N2: VITRA.

1 Introduction

The project *Vitra* (VIsual TRAnslator) deals with the relationship between natural language and vision. Experimental studies are being carried out in the way of designing an interface between image-understanding and natural language systems, with the aim of developing systems for the natural language access to visual data.

Collaborating with the vision group at the IITB, Karlsruhe, different domains of discourse and communicative situations are examined. Scenarios under investigation include:

- Answering questions about observations in traffic scenes
- Describing routes based on a 3-dimensional model of the Saarbrücker University Campus
- Generating running reports for short sections of soccer games
- Communicating with an autonomous mobile robot (*planned*)

The task of the vision group at the IITB is to recognize and to track moving objects within real world image sequences. This information about mobile objects and there locations over time together with the knowledge about the stationary background constitutes the so-called *geometrical scene description* (GSD). In Neumann [1984] this intermediate geometrical representation, enriched with additional world knowledge about the objects, has been proposed as an idealized interface between a vision component and a natural language system.

In our joint work results have already been obtained in the investigation of traffic scenes (Schirra et al. [1987]) and short sequences from soccer matches (Herzog et al. [1989]). Apart from the trajectory data supplied by the *Actions*[2] system synthetic scenes have been studied in *Vitra* as well (c.f. Herzog [1986]).

In the more recent *Xtrack* system (Koller [1992]) a model-based procedure has been employed for the detection, tracking, and classification of vehicles in a traffic scene. This system is able to provide a 3-dimensional reconstruction of the recognized mobile objects. The vision group has even been concerned with the model-based recognition of non-rigid mobile objects. Using a cylindric representation and a model of human walking the approach described in Rohr and Nagel [1990] aims at recognizing a pedestrian and his exact state of motion.

Based on these advances the *Vitra* system, which constitutes a workbench for the development of an integrated vision and natural language processing system, can be extended in order to cope with 3-dimensional geometric representations. This paper will focus on the representation and visualization of 3-dimensional geometric information within the new *Vitra* workbench. In addition, the processing and visualization of the underlying image sequences will be discussed.

[2]The akronym stands for "Automatic Cueing and Trajectory estimation in Imagery of Objects in Natural Scenes"

2 Geometric Modeling

The aim of the modeling system in *Vitra* is to provide simple geometric representations for solid objects. The objects are described in terms of there surface boundaries, i.e. they are specified as collections of faces. The geometric representation is currently restricted to the following types of faces: planar polygon, disc, ring, cylinder and sphere. The geometrical model can have a hierarchical structure since it is possible to group objects together into a new object. The system also supports the definition of types of objects, which might be instantiated several times.

In *Vitra* we are concerned with the automatic interpretation of time-varying scenes. Thus the system has to cope with mobile objects and their trajectories. Object movements correspond to the application of geometric transformations. In our previous system, where mobile objects were represented as centroids, only translations were possible. In addition, the new *Vitra* workbench also allows for rotations around the three coordinate axes. The general case of an arbitrary rotation does not occur in our applications and restricting the possible geometric transformations to a limited set of rotations yields a performance improvement. In order to specify relative motion, moveable parts of articulated objects are represented in a local coordinate system. The origin of this coordinate system will be located on the corresponding joint. The overall movement of an articulated object is then decomposed into the recursive application of the simple geometric transformations mentioned above.

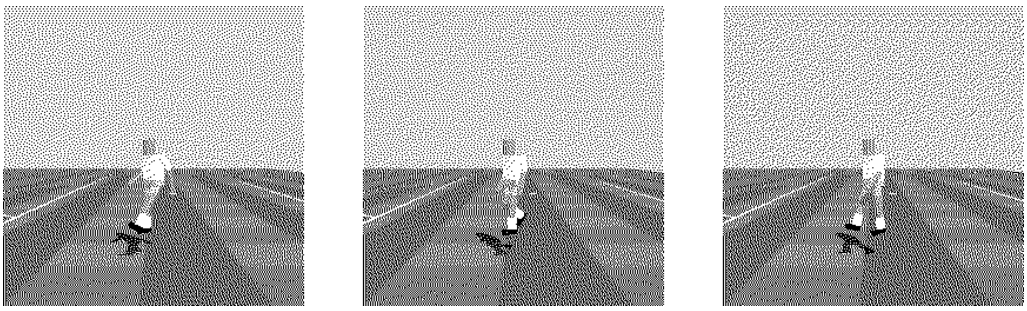

Figure 1: Geometric model of a human body

Fig. 1 shows an example of an articulated body. The approach of Rohr (cf. Rohr [1989]) has been adopted in *Vitra* in order to represent the players in the soccer domain[3]. Synthetic trajectories for this model of the human body can be obtained by applying a kinematic model of human walking. This modeling of human walking is based on medical data and has been utilized for the incremental recognition of pedestrians in image sequences (cf. Rohr [1989]). In fig. 1 different movement states of the walking cycle are shown.

[3]The representation of the torso has been simplified to a rectangular solid, because ellipsoids are not yet availlable in our geometric representation.

The geometric models for vehicles as described in Koller [1992] will also be incorporated into our system.

3 Visualizing the 3-dimensional Model

Within the basic windows of the *Vitra* system a parallel projection (from a bird's eye view) of the 3-dimensional geometric model is visualized. Using *CLIM* (Common Lisp Interface Manager), operations like zooming, scaling, and rotating were easy to implement since the graphic functions support affine transformations in the 2D plane directly (cf. fig. 2).

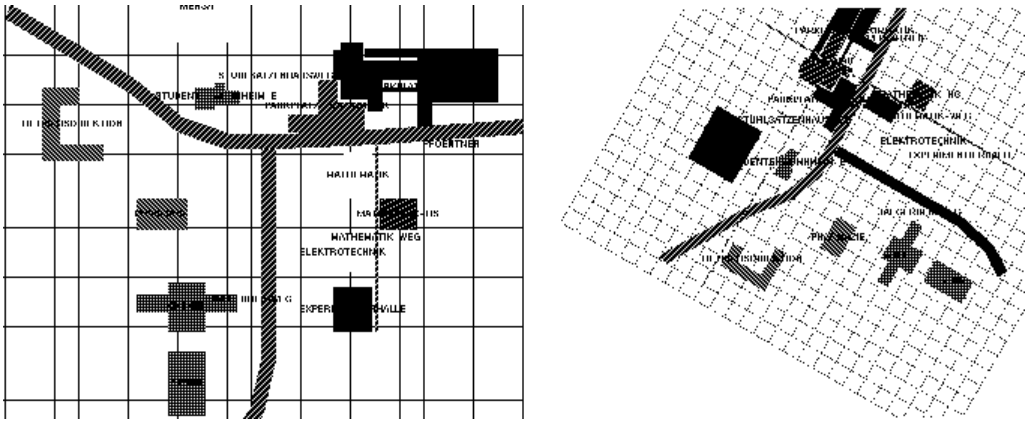

Figure 2: Some 2D representations of the 3D model

In order to generate perspective views, the internal 3-dimensional geometric representation is translated into a format suitable for a raytracer[4]. Fig. 4 and fig. 3 show images from two different domains, which have been generated using the `rayshade` raytracing program.

Current work concentrates on integrating a hidden line/hidden surface algorithm into the implemeted system in order to provide a means for the interactive generation of perspective views (without shading).

[4] see Foley et al. [1990]

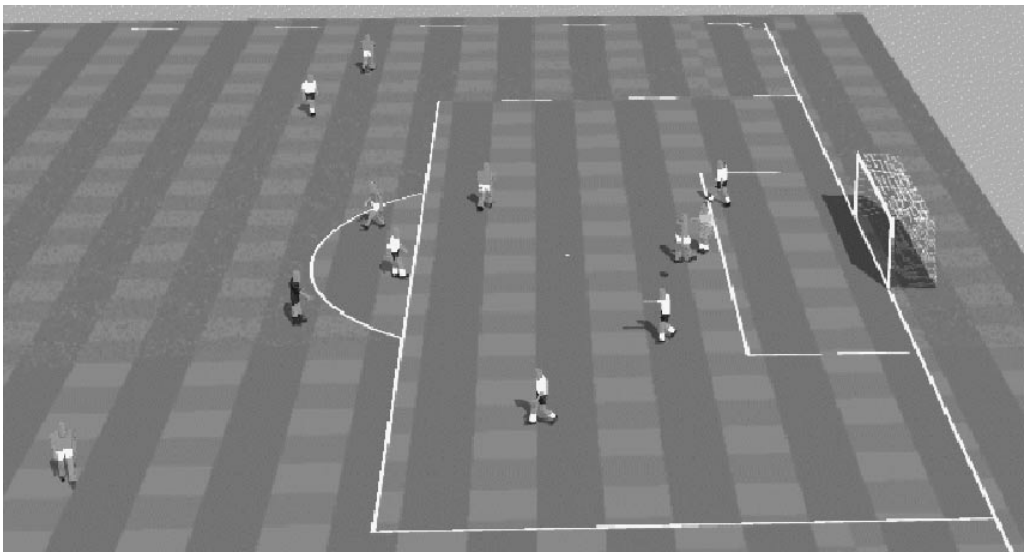

Figure 3: Synthesized image from the Soccer domain

4 Images and Bitmaps

This section describes the different formats and tools that are used for images and bitmaps in the *Vitra* workbench.

4.1 Formats

Within the workbench three basic picture types can be distinguished:

1. Synthetic *color images* constitute different views of the 3-dimensional geometric representation of a scene. With usually 24 bits per pixel this picture type may provide the most detailled visual information.

2. Since color images are not yet considered by the vision group the digitized video frames represent *grayscale images* with 8 bits per pixel. Such grayscale images might also be generated from color images when ordinary printing is required.

3. *Bitmaps* are used to display pictures on a 1-bit monochrome computer screen. In a bitmap each pixel can either be turned on or off. In *Vitra* bitmaps will also be utilized for screen hardcopies and graphical on-line documentation.

The following particular file formats are used with color images:

PPM: Portable pixmap — a simple uncompressed full color image format. This is the basic format for color images in *Vitra*.

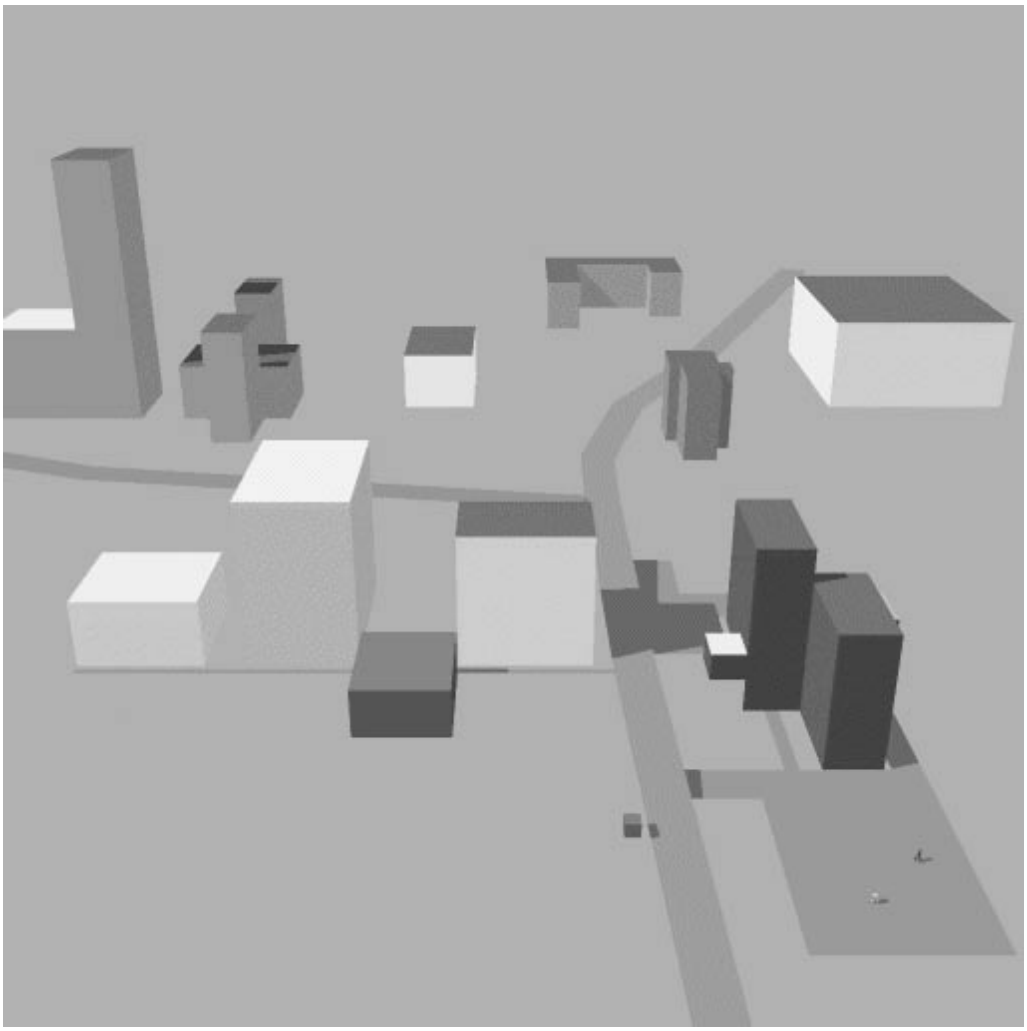

Figure 4: Partial 3D model of the University campus

MTV: Mark VanDeWettering raytracer format — uncompressed full color image as it is generated by the `rayshade` raytracer, too.

GIF: Graphics interchange format — common compressed color image format with colormap and maximal 256 colors. The **GIF** format is mainly used to produce color hardcopies.

PIC: `PCPAINT/Pictor` format — compressed color image with maximal 256 colors and with an optional colormap. This format is used in animation.

For grayscale images the following file formats have to be considered:

PGM: Portable graymap — a simple uncompressed grayscale image format. This is the basic format for grayscale images in *Vitra*.

VIP: IITB format — used within the *Actions* system.

PM: GRASP Lab (University of Pennsylvania) format — used within the *Xtrack* system.

PS: PostScript — Picture written in the PostScript page description language using the `image` operator. This format is used for printing grayscale images. Automatic half-toning is performed by the PostScript interpreter.

Bitmaps will be represented in one of the following formats:

PBM: Portable bitmap — a simple uncompressed bitmap format This is the basic format for bitmaps in *Vitra*.

PS: PostScript — used for printing bitmaps and screen hardcopies.

AGFA: Agfa P400 printer format — used for printing.

BMP: Picture Editor format — bitmap file format which allows either for run-length encoding, Huffman encoding, or Lempel-Ziv[5] compression.

4.2 Tools

To load and display images and bitmap graphics within the *Vitra* system the author has written a programm called `bitmap tool`. The program is able to read and write several image formats and was also designed to transform grayscale images into bitmaps.

As one way of displaying grayscale images on a bi-level device we use 2*2 or 3*3 pixel areas in order to simulate the local picture intensity (c.f. Herzog [1986]). Smaller bitmaps and good results can be obtained using dithering with blue noise, namely the

[5] c.f. Welch [1984]

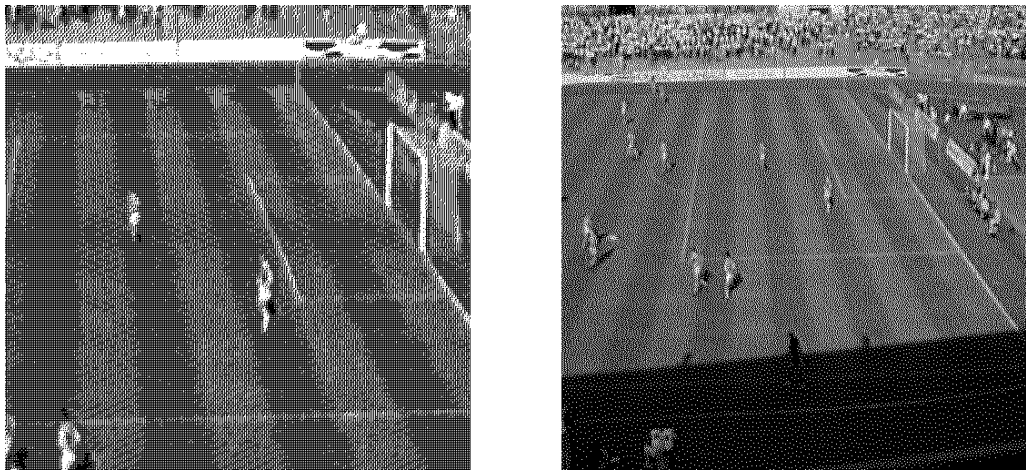

Figure 5: Half-toning with pixel patterns and with Floyd/Steinberg filter

Floyd/Steinberg error diffusion algorithm (c.f. figure 5). The half-toning techniques mentioned here are described in more detail in Burger and Gillies [1989] and Ulichney [1987].

Since much of the functionality of the `bitmap tool` is also provided with the utilities mentioned below the future versions of the program can be simplified a lot.

`Pbmplus` is a set of C programs for converting various image formats from and to the basic `pbmplus` formats **PPM**, **PGM**, and **PBM**. In addition, `pbmplus` includes several tools for manipulating images.

The `xv` program is an interactive tool to display and manipulate images and bitmaps in the X window system. Since the internal representation of an image and its visualization on a particular screen are clearly distinguished any type of picture can be shown and manipulated on any type of X display, whether it is a color screen or a 1-bit monochrome display. Figure 6 shows an example.

4.3 Summary

Within the *Vitra* workbench three different tools are used to transform and visualize images and bitmaps in various formats. The following table summarizes which utilities and formats may be used together:

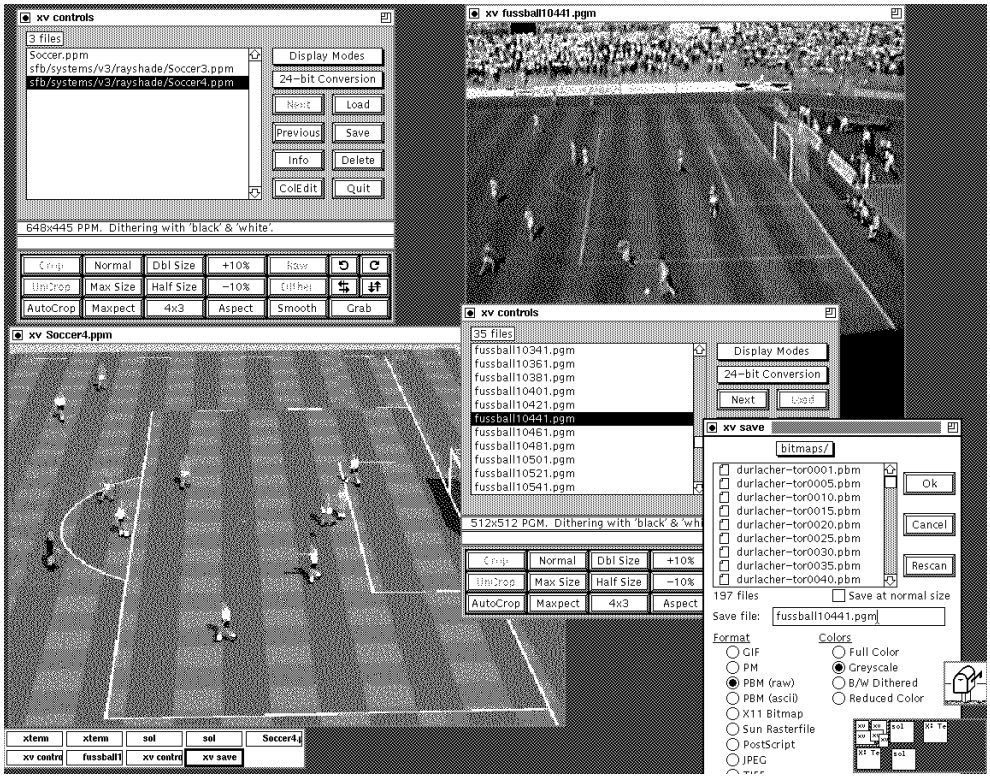

Figure 6: Interacting with xv

Tools and Formats	`Bitmap Tool`	`pbmplus`	`xv`
PPM	read	read/write	read/write
MTV	—	read/write	—
GIF	—	read/write	read/write
PIC	—	read/write	—
PGM	read/write	read/write	read/write
VIP	read/write	—	—
PM	—	—	read/write
PS 8 bit	write	write	write
PBM	read/write	read/write	read/write
PS 1 bit	read/write	write	write
AGFA	read/write	—	—
BMP	read/write	—	—

The filters for the **PIC** format are added to `pbmplus` by the `xviewgl` program (c.f. section 5). Only PostScript bitmaps written by the `bitmap tool`, `pbmplus`, or Symbolics hardcopies will be recognized.

Besides the tools used directly for the *Vitra* workbench many more utilities exist under the X window system. These include graphic editors, a PostScript previewer, and other tools to visualize and modify images and bitmaps.

5 Animation

One major theme of research within *Vitra* is the automatic interpretation of dynamic imagery. Thus the need for visualizing time-varying visual information as well as the underlying image sequences arises.

As described in Herzog [1986], the movements of the recognized objects are displayed by projecting an iconic representation for each object into the 2D representation of the stationary background of the dynamic scene (cf. fig. 7). Since only the moving icons are redrawn, the animation is fairly fast. Synchronisation is achieved by transforming the current timemark into an array index for accessing the corresponding object coordinates. The same synchronization technique is utilized when animating bitmaps, i.e. half-toned images, which are generated from the underlying video sequences (cf. fig. 8) or from synthesized images.

In the *Vitra* workbench, the animation operation can also be performed in slow motion or single step mode. Bitmaps and iconic representation might be displayed at the same time (cf. fig. 9).

Displaying bitmaps is usually sufficient in the *Vitra* system. Sometimes however, the need to visualize a sequence of color images arises. Since it is fairly impossible to implement this feature within the different lisp environments[6] the `xviewgl` program is used to run animations under the X window system.

[6] CLIM does not support color images very well (c.f. CLIM [1991]).

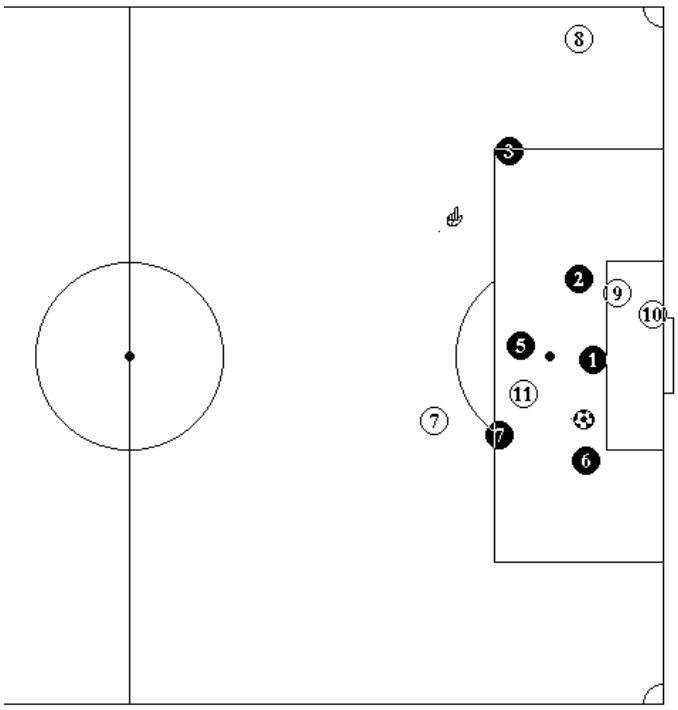

Figure 7: Iconic representation

Figure 8: Three frames from a traffic scene

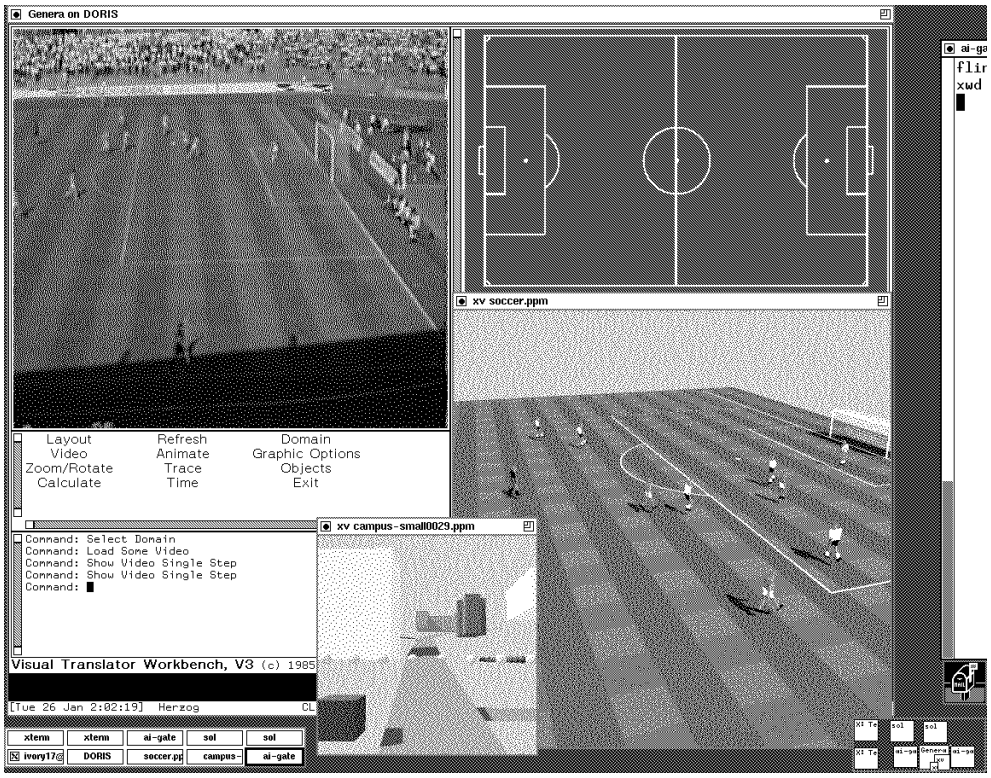

Figure 9: Basic windows of the *Vitra* workbench

```
VIDEO t
PLOAD frame01.gif, 1
PLOAD frame02.gif, 2
PLOAD frame03.gif, 3
PLOAD frame04.gif, 4
PLOAD frame05.gif, 5
PLOAD frame06.gif, 6
TEXT 100,400,"VITRA GRASP Demo",100
PFADE 0,1,1,8
PFADE 0,2,1,8
PFADE 0,3,1,8
PFADE 0,4,1,8
PFADE 0,5,1,8
PFADE 0,6,1,8
WAITKEY 1000
PFADE 0,5,1,20
PFADE 0,4,1,20
PFADE 0,3,1,20
PFADE 0,2,1,20
PFADE 0,1,1,20
WAITKEY 1000
```

Figure 10: A simple GRASP animation script

This tool was inspired by the `GRASP` (GRAphical System for Presentation) animation package, which is popular on IBM PCs. `Xviewgl` can be used to "*view*" packed and unpacked *GRASP libraries*. Such a library consists of a set of color images and an animation script written in the GRASP language. Figure 10 shows a sample animation script. The most interesting GRASP commands are:

`PLOAD` *name, buffer* Load a picture, i.e. an image with its color map, from the file *name* into a buffer. Since there are 16 picture buffers availlable *buffer* has to be a number between 1 and 16.

`PFADE` *fade, buffer, speed, delay* Fade the picture from the specified picture buffer to the screen. The *delay*, in 1/10 seconds, specifies how long to pause after the fading and the *speed* parameter determines the duration of the fading itself. The *fade* must be a number between 0 and 25.

`PFREE` $buffer_1$ ... $buffer_n$ Free the specified picture buffers.

`CLEARSCR` Clear the `xviewgl` window.

`CLOAD, CFADE, CFREE` Like `PLOAD`, `PFADE` and `PFREE`, but for clippings. A clipping is an image without an own color map. There are 128 clipping buffers.

`TEXT` *x, y, string, delay* Draws a string onto the screen at position *x*, *y*. The *delay* specifies how long to wait afterwards.

`VIDEO` *type* Set (virtual) screen type.

`WAITKEY, IFKEY, GOTO, EXIT` These commands can be used for keyboard input and to influence the flow of control.

Other GRASP commands include simple graphics as well.

Pictures and clippings to be used with GRASP have to be stored in **PIC** format. The necessary filters for translating from and to the portable color image format are provided with the `xviewgl` package. In addition, `xviewgl` supports the reading of pictures in the **GIF** format.

6 Technical Notes

The current version of the *Vitra* system is written in Common Lisp and CLOS, with the graphical user interface implemented in CLIM.

The *Vitra* system is availlable for CLIM 1.1 with Genera 8.1.1, on Symbolics 36xx Lisp Machines and Symbolics UX1200S Lisp Coprocessors, Allegro Common Lisp 4.1, on SPARC Workstations, and Lucid Common Lisp 4.0.2, on Hewlett Packard 9720.

Additional tools, all written in C, are availlable in the Unix environments. These include the `pbmplus` toolkit, by Jef Poskanzer, `rayshade 4.0`, by Craig Kolb and Rod Bogart, `xv`, by John Bradley, and `xviewgl`, written by Brad Daniels.

Acknowledgements

The use of the programs mentioned above is gratefully acknowledged. Dieter Koller and Karl Rohr, both being involved in our partner project in Karlsruhe, gave helpful comments and suggestions throughout the development of the *Vitra* workbench. This strong support made our cooperation a close one. Jörg Baus, Guido Bosch, Markus Bolz, Clemens Huwig and Alassane Ndiaye provided help for finding and installing the public domain software. Klaus-Peter Gapp assisted in implementing the *Vitra* workbench as it is described here.

References

P. Burger, D. Gillies. *Interactive Computer Graphics*. Addison-Wesley, Reading, MA, 1989.

CLIM. *Common Lisp Interface Manager (CLIM)*. Symbolics Inc., Cambridge, MA, 1991.

J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes. *Computer Graphics. Principles and Pratice*. Addison-Wesley, Reading, MA, 2nd edn., 1990.

G. Herzog. Ein Werkzeug zur Visualisierung und Generierung von geometrischen Bildfolgenbeschreibungen. Memo 12, Universität des Saarlandes, SFB 314 (VITRA), Saarbrücken, 1986.

G. Herzog, C.-K. Sung, E. André, W. Enkelmann, H.-H. Nagel, T. Rist, W. Wahlster, G. Zimmermann. Incremental Natural Language Description of Dynamic Imagery. In: C. Freksa, W. Brauer, eds., *Wissensbasierte Systeme. 3. Int. GI-Kongreß*, pp. 153–162, Springer, Berlin, Heidelberg, 1989.

D. Koller. *Detektion, Verfolgung und Klassifikation bewegter Objekte in monokularen Bildfolgen am Beispiel von Straßenverkehrsszenen*. Infix, St. Augustin, 1992.

B. Neumann. Natural Language Description of Time-Varying Scenes. Report 105, Fachbereich Informatik, Univ. Hamburg, 1984.

K. Rohr. Auf dem Wege zu modellgestütztem Erkennen von bewegten nicht-starren Körpern in Realweltbildfolgen. In: H. Burkhardt, K. H. Höhne, B. Neumann, eds., *Mustererkennung 1989, 11. DAGM Symposium*, pp. 324–328, Springer, Berlin, Heidelberg, 1989.

K. Rohr, H. H. Nagel. Modellgestützte Bestimmung des Bewegungszustandes von Fußgängern in Realweltbildfolgen. In: R. E. Großkopf, ed., *Mustererkennung 1990; 12. DAGM Symposium*, pp. 52–58, Springer, Berlin, Heidelberg, 1990.

J. R. J. Schirra, G. Bosch, C.-K. Sung, G. Zimmermann. From Image Sequences to Natural Language: A First Step Towards Automatic Perception and Description of Motions. *Applied Artificial Intelligence*, **1**, 287–305, 1987.

R. Ulichney. *Digital Halftoning*. MIT Press, Cambridge, MA, 1987.

T. A. Welch. A Technique for High-Performance Data Compression. *Computer*, **17**(6), 8–19, 1984.