Detection and recognition of movement

7th term project

Winter 2001-2002

EPFL - ISR

Assistant 1:	Sebastien Grange
Assistant 2:	Terry Fong
Professor:	Roland Siegwart

Table of contents

List of figures

1. Summary

This project consists in using computer vision for detecting and recognising human gestures. A model, specific to human beings, should describe the human being and the gestures should be recognised in real-time.
For achieving this goals I used a pair of fix mounted stereo cameras and a standard personal computer (Intel Pentium 4, 1800 Mhz). Figure 1 shows the cameras.

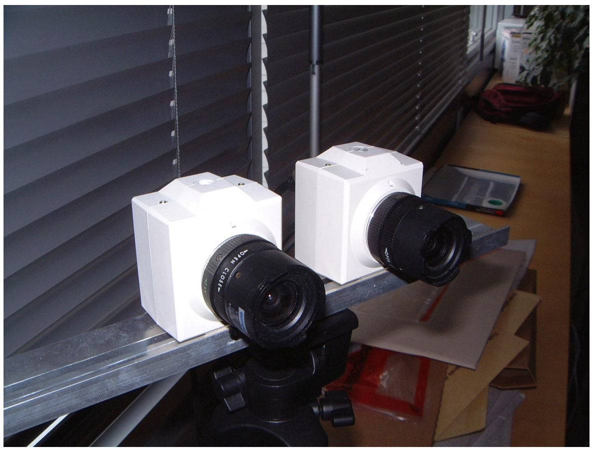

Figure 1: The stereo cameras

This system allowed me to calculate depth information. As the cameras are mounted fix, the background can be subtracted.
I chose a rather simple model of the human being, describing head, body, upper and lower arms. This static model, completed by time information (i.e. “when was the corresponding image taken”) allows the description of the dynamic of the movement recognised, that is, their velocity and frequency.
For the gesture detection and recognition I implemented a program working in two modes, detection and tracking mode. The idea is to first trying to detect the whole human being and as the whole human is detected to track the detected features using a less resource consuming algorithm and considering some human specific restrictions (e.g. arms are within a certain distance to the head).
The detection algorithm is based on detecting a human contour, partitioning this contour and searching for the best matching combination between the parts of the contour and the features of the human model. This approach allows me to have a different number of parts of the

contour and of human features (i.e. occlusion of some parts of the human being or noise detected as part of the human).
Unfortunately I did not manage to completely finish the implementation.
Figure 2 shows results of background subtraction and human contour detection.

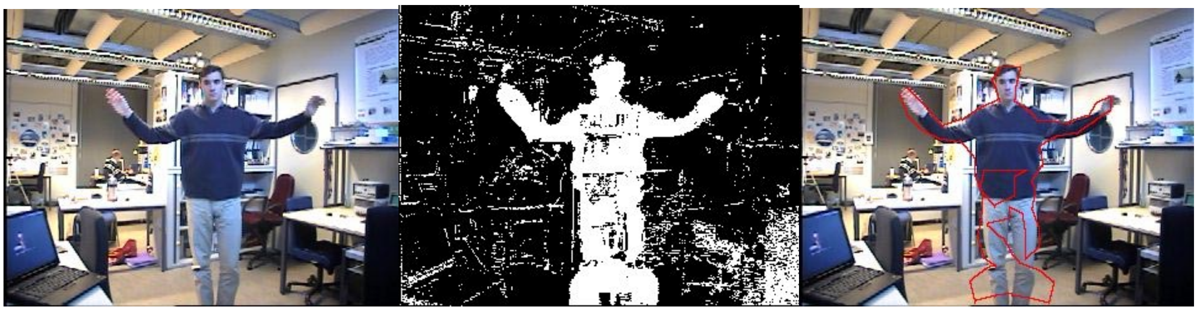

Figure 2: Input image, background subtracted, contour found

2. Introduction

2.1. Problem context

2.1.1. General context

Today Human–Computer Interaction (HCI) is generally restricted to the use of pointing devices (e.g. mouse, joystick), keyboards and displays. Communication using these instruments is very different from the ways of communication used between human beings, to say very unnatural.
This kind of communication is sometimes very difficult, very slow and/or very restricted. Especially for certain groups of human beings, like children, handicapped or elderly people communication with computers is often difficult if not impossible.
Also considering the very complex actions that can be performed by today's computers in everyday life these input and output devices seem somewhat rudimentary.
The goal of this project is to use computer vision to detect and recognise human gestures and posture to provide a more natural interaction between human beings and computers.
When communicating with our fellow human beings, our sense of vision provides us mainly with information about what the person is doing (gestures), where he is (position) and about his general degree of activity (studying, manually working, etc).
Providing this information to the computer would certainly enhance communication between the computer and humans.
I will direct my project mainly in the direction of vision based feature extraction but in a way that adding other facilities won't be difficult.
To have a more complete solution for interpreting human actions voice or sound recognition and perhaps haptic sensors could be added.
Vision based control can be of two main kinds from the human point of view:

- Active: people can control a computer system using specific gestures
- Passive: the computer system interprets the peoples natural behaviour and gestures and reacts in function of his interpretation

In a passive vision based control system the human isn't directly commanding it, but it will enhance nevertheless Human-Computer Interaction dramatically, by making the machine aware of the task the human is trying to perform.

The goal of this project will be a vision-based system that detects human beings and creates a usable model of his/her posture in real-time. If the output of the system is accurate enough, this movement could be interpreted by an active or by a passive vision based control system. Because I don't think that the output will be accurate enough in the project environment that I will describe later, I will design my system more specifically for an active vision system.

2.1.2. Three examples

I will base the development of the system mainly in regard of three examples. They are all of different kind but they have also some things in common. I will base my hypotheses on the points of each example, which I think are essential.
The three examples are all examples of smart spaces. Smart spaces are spaces (mostly rooms) that interact with the people inside to enhance the functionality of the room. This interaction is done using computing devices. We could imagine for example controlling the curtains and the video projector in a conference room by gestures.
In the following I will shortly present and discuss the three examples:

- Smart surgery room
- Smart conference room
- Smart play room

Today's surgery rooms are highly complicated rooms with a variety of different facilities (changing lighting, adjusting the bed, etc). The surgeon has several foot switches to control some of the facilities and several assistants, that also partly control the rooms' facilities, surround him. The goal here is to help the surgeon control the environment by using a vision based gesture detection system. Controlling some of the rooms' facilities using natural gestures would facilitate the surgeons' task.
In the smart conference room the goal also is to control some of the rooms' facilities (curtains, video projector, presentation control) using a vision based gesture detection system.
The smart playroom isn't really an example of a smart space. It's more about controlling a computer using gestures. Especially small children aren't able to interact with computers using today's highly artificial ways of communication between a computer and a human being. Nevertheless many of the future toys will be computer based and will need a more natural communications interface. As a solution we could imagine a vision-based system

detecting some of the child's gestures and acting accordingly. That would provide even small children with a possibility to interact with computers and so get used to one of today's most used tools. Besides that the system could also provide some safety by monitoring the children.
As we can see the specifications of the applications to be used in the three examples will be quite different. As a consequence of that the price of the systems will vary strongly.
The system used in the smart operating theatre should be very reliable (I could imagine using several pairs of cameras) and will probably cost ten times more than the other two systems. The smart office system should also be quite reliable, but probably have less functionality (will detect fewer gestures) and is slower. The system used in the play room must not be very reliable, it could even be more fun if the computer does not behave the same way every time or it could be challenging to learn the exact gestures the computer really is able to detect.
Also the environmental conditions (lighting conditions, clothing of the people in the smart space, dimensions of the smart space) will be quite different in the three examples.
But nevertheless the basic structure of the program will be the same and above all interpreting of the output will change. The operating theatre system will have better hardware (higher resolution cameras, faster computers) that will enhance the system performance, but the basic detection routine won't change much, perhaps there will be additional evaluation functions.

2.2. Problem description

The first part of the problem consists in developing a model of the human being with which it would be possible to describe the most important human gestures. This model should be applicable in most situations where gesture detection is needed. The model should allow the tracking of the movement of head, upper and lower arms. But it should also be evolutive, in that it should be designed in a way that makes it possible to extend it as more sensing modalities, computing power or features are added to the system.
The second part is to define a strategy for detecting the features of the model and to detect their movement.
Finally this solution should be implemented on a standard personal computer using a pair or stereo cameras and ImageNation PXC200 frame grabbers. The cameras are shown in figure 3. The tracking should run in real-time, it should be possible to track gestures of a normal speed. Although this vision system works at 20 Hz and the maximum frequency of a human gesture

is 8 Hz, I don't think that it would be possible to track very fast gestures, because the system will probably slow down due to image processing and tracking fast gestures also means searching in large areas of the images for the tracked features, what additionally will slow down the system.
The resolution of the system should be sufficient to allow basic interpretation of body language, that is it should detect small movements as well as making possible the calculation of their velocity and/or frequency.

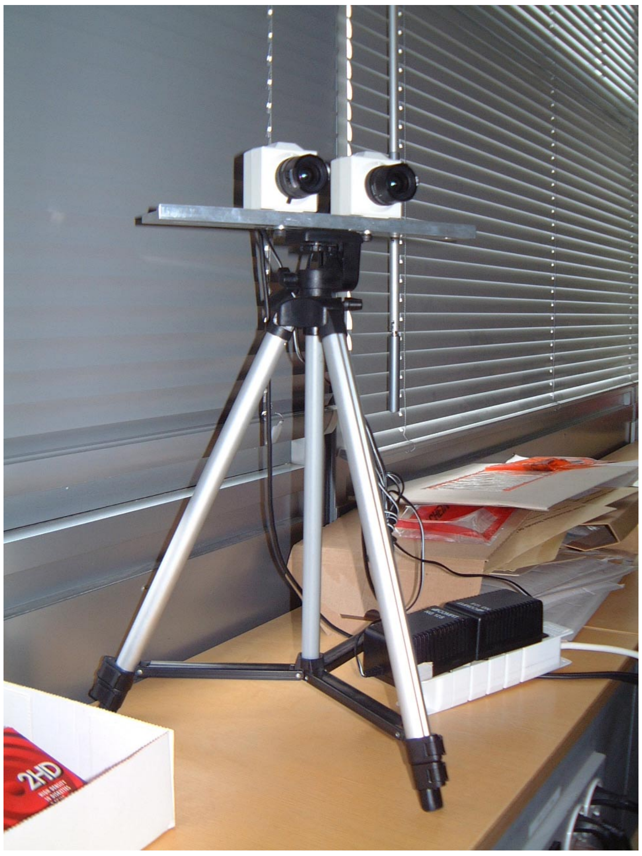

Figure 3: The stereo cameras

It isn't the goal to find an interaction system (i.e. gesture command interpreter), but to develop a generic solution and a program that just detects and recognizes gestures in an image sequence and makes this information available to another application, which will interpret this information specifically to a given situation.

2.3. Hypotheses

For limiting and specifying the problem I made the following hypotheses:

Fixed camera position:

I assume the camera position fixed in the room. That will allow me to work with background subtraction. This hypothesis is quite reasonable, because the system will be designed for use in smart spaces.

One person at a time in the field of vision:

It is extremely difficult to differentiate and to track two people at a time; therefore I make the hypothesis that there is just one person at a time in the field of vision. This restriction poses no problems in the case of a smart conference room, because there is mostly just one person "performing" at a time. But in the two other cases it won't be easy to keep this restriction, the surgeon won't operate alone and playing alone isn't fun.

No skin colour detection:

Skin colour detection is an extremely helpful modality. However I won't use it in this project. I won't use it mainly because it can't be used in the case of a surgery room because the surgeon will be wearing gloves and a mask will cover his face. Also it is not always reliable, because of clothing, lighting condition changes and occlusion (e.g. using tools).

The person is upright and faces the camera:

I will assume the hypothesis that the person tracked is standing or sitting. That allows me to make considerable simplifications detecting the contour of the person.
This hypothesis is realistic in most situations, because people are normally gesticulating only while standing or sitting. As I can't use skin colour detection it won't be possible to decide whether the detected person is facing the camera or not. So it is necessary to assume that every person detected is facing the camera. This hypothesis could eventually be relaxed later on (person should only face the camera while entering the field of vision).

No occlusion:

In a first version I will consider that there is no occlusion. That means that there aren’t any gestures to recognise where some parts of the body occlude others. The tracker shouldn’t loose the person tracked, when some parts are occluded, but it won’t be possible to estimate the position of the parts occluded.
This quite strong restriction is only made to validate the level of detail of the model I am building, but occlusion can be acceptable in the future since I am working with a stereoscopic vision system. For example the authors of [3] describe a method that can handle occlusion.

3. Related research

3.1. L. Zhao, C. Thorpe: Recursive context reasoning for human detection and parts identification (2000)

L. Zhao and C. Thorpe describe in this paper a recursive context reasoning approach for human detection. The method consists in dividing a detected human contour and registering the parts with a given set of features of the human model (e.g. head, arms, legs, etc) respecting some consistency rules. The best match is chosen and the edges of the features found in the contour are aligned with the outlines of the detected contour. This procedure is repeated recursively using the features found and the human model as input for the next iteration until they can make a reliable decision on whether the image contains a person or not. L. Zhao and C. Thorpe defined a TRS – invariant body model to encode the shapes of the body parts and the context information (size and spatial relationships between body parts).
This problem is quite similar to the one I’m about to solve, but I will nevertheless show you another approach. I will use another, simpler, model of the human, because I think the implementation of the model described in this would be very time consuming and I think a simpler model will suffice to detect and recognise gestures in the given context.
I also will implement another tracking strategy, because I think that tracking all features of my human model in real-time using this algorithm will be to resource consuming.

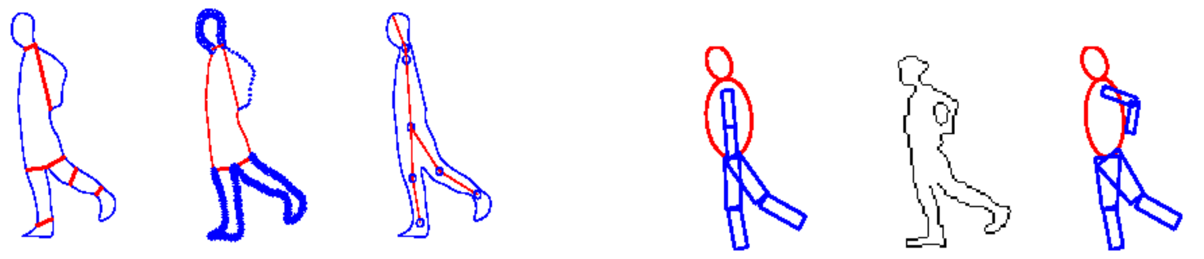

Figure 4: Body part identification and localisation, from left to right: contour partition, main body part identification, updated locations of the identified body parts, reconstructed outlines of the body parts, edge images, aligned body parts

3.2. R. Cutler, M. Turk: View-based interpretation of real-time optical flow for gesture recognition (1998)

This paper by R. Cutler and M. Turk describes another approach of vision-based gesture detection. R. Cutler and M. Turk don't depart using a human model and trying to find the features of this model in the image sequence; they rather use optical flow to segment the dominant moving body parts into "motion blobs". Using motion blobs and their general characteristics, actions are recognised. Once an action is recognised, parameters of that action (e.g. frequency, velocity) are estimated.

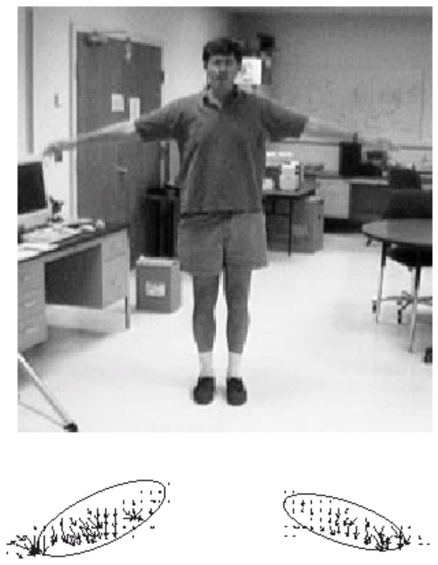

Figure 5: Flapping action, flapping flow and segmented blobs

Using optical flow isn't possible in my case, because it's not accurate enough. It won't be possible to recognise more complex gestures than waving for example and that won't be sufficient above all for the surgery room example.

3.3. N. Jojic, M. Turk, Th. S. Huang: Tracking self-occluding articulated objects in dense disparity maps (1999)

The authors of this paper describe a tracking approach usable for articulated objects (like human beings) using the disparity maps derived from stereo image sequences. The model takes into account occlusions and enforces articulation constraints among the parts of the objects. The results are encouraging as can be seen in figure 6. Unfortunately this approach needs a very accurate initialisation phase before starting to work, that won't be easy, especially in the case of the playroom system.

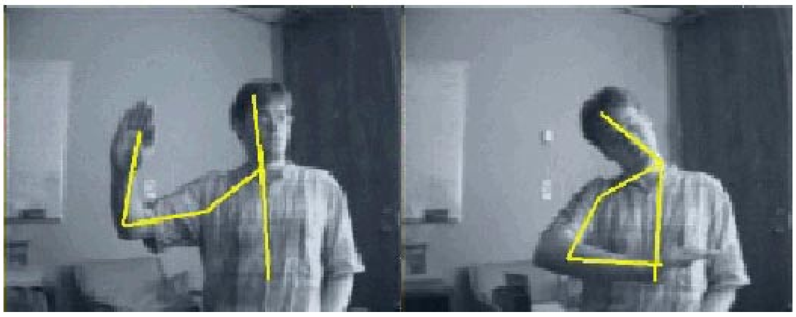

Figure 6: Upper body tracking in presence of occlusions

3.4. S. Grange: Vision based sensor fusion for active interfaces: H.O.T. – human oriented tracking (2000)

S. Grange uses stereovision and colour images to locate particular human features. From the tracking results he builds a model of the human pose and segments and parameterises human movements. This model is finally used to monitor the persons' activity. His approach is different from the solution I will have to find in his use of skin colour detection. Also H.O.T.

wasn't designed for gesture interpretation in the first place, but for activity monitoring, so the model isn't suitable for achieving the goals of my project.

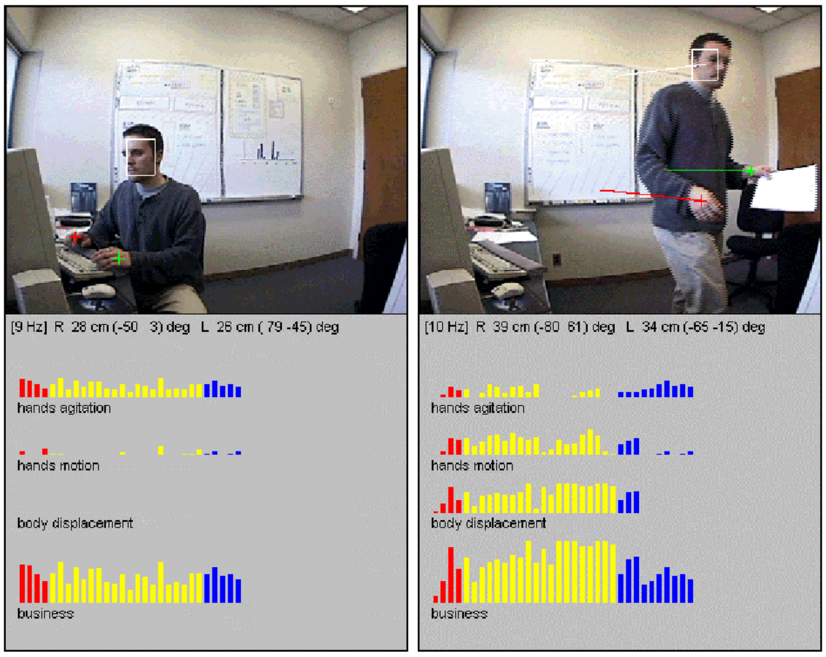

Figure 7: Activity monitoring using H.O.T.

3.5. M. Turk: Visual interaction with lifelike characters (1996)

In this paper M. Turk describes all the basic techniques that can be used in computer vision for detecting and tracking features in an image sequence. He describes techniques like background subtraction, colour and motion measuring. He also describes a basic head tracking routine and a head counting routine. These and other features described were developed for enabling a simple game of Simon Says.

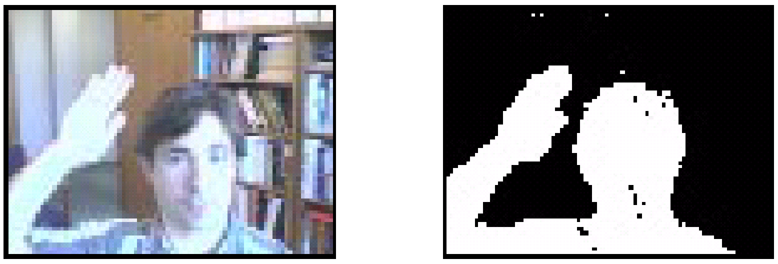

Figure 8: Background subtraction

I used some of the techniques he describes in this paper within my project, sometimes in a slightly changed way and the paper gave me a general over view, of what is possible in computer vision and how it basically works.

3.6. A. Mulder: Human movement tracking technology (1994)

In this paper A. Mulder describes different human movement tracking systems using different technologies. He also describes the different technologies that can be used. Especially interesting for me were his descriptions of, so-called, outside-in tracking system (systems that employ an external sensor that senses artificial sources, markers or natural sources on the body) and the specifications he gives for such systems as well as descriptions of available technologies for implementing those systems. This information also gave me a general idea of computer vision. It was also quite interesting for me to see, for example, what resolution to expect from a vision based Human-Computer Interaction system.

3.7. H. Sawada, S. Hata, S. Hashimoto: Gesture recognition for human-friendly interface in designer – consumer cooperate design system (1999)

In this paper the authors describe optical flow based gesture recognition system able to determine hand motion. They use skin colour detection to extract the area of a hand, then they extract the contour of the hand area and determine the tips of the fingers. The fingertips are detected because of their sharp curvature. The 3D position of the hand is finally obtained by applying a stereo matching technique to each of the fingertips. They use an optical flow technique applied to the fingertip areas for detecting the 3D motion of the hand. The results are applied to a gesture driven design system, where customer and designer can interactively exchange their opinion for the prototyping. Commands such as pointing, movement, rotation and zooming are recognised by the vision system.
As it isn’t for the skin colour based contour extraction this project is situated in a similar field as my project. The main difference is that they track and detect gestures on another level of detail. They track for example the movement of the fingers of one hand.

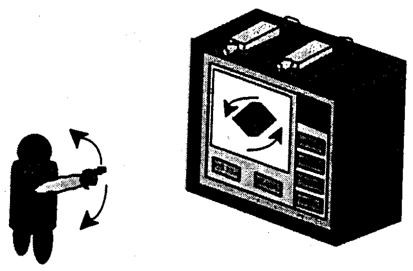

Figure 9: Gesture driven design system

4. The chosen solution

In the following section I will describe the solutions I chose in regard of the hypotheses mentioned in section one and in regard of the studies mentioned in section two.

4.1. Human model

Our way of recognising and interpreting other human beings gestures is a very complex process. We do not only detect "full-scale" gestures like waving hands, but also small gestures like a small degree tilt of the head or changing of face expression. We are able to notice someone being sad without speaking with him.
For this recognition and interpretation the different parts of the human body are of different importance. Arms for example don't do much else then carry hands; also we don't really use our upper body for expressing something, as it isn't for example for showing attention with leaning forward. For interpreting the gestures produced by the different body parts, we need different levels of details. Hand gesture recognition for example needs much more detail, than arm gesture recognition, because there is the movement of five fingers to recognise.
As I said in section 2.1.1., I will design my system with an interpretation system in mind, which interprets active gestures. For the reason of that I decided to describe the humans' head, his lower and his upper arms with a rather simple model. I won't include hands in my model although they express much more for us than arms, but the only thing that really carries information in active gestures is their position. This position can be estimated easily as being in the prolongation of the lower arm; and the arms are much easier to find in images

because of their size. Also, the hands cannot really be tracked reliably with the image resolution that the real-time constraint imposes.
The upper and lower arms will be described by rectangles, the head by a circle. This simple model also takes in consideration that I will mainly work with human contour information, as I can’t hope to have any reliable colour information. The existence of a body below the head (at the same distance) could be considered for checking reason or as the point of departure in the tracking algorithm (the body will be the easiest thing to find because of its size).

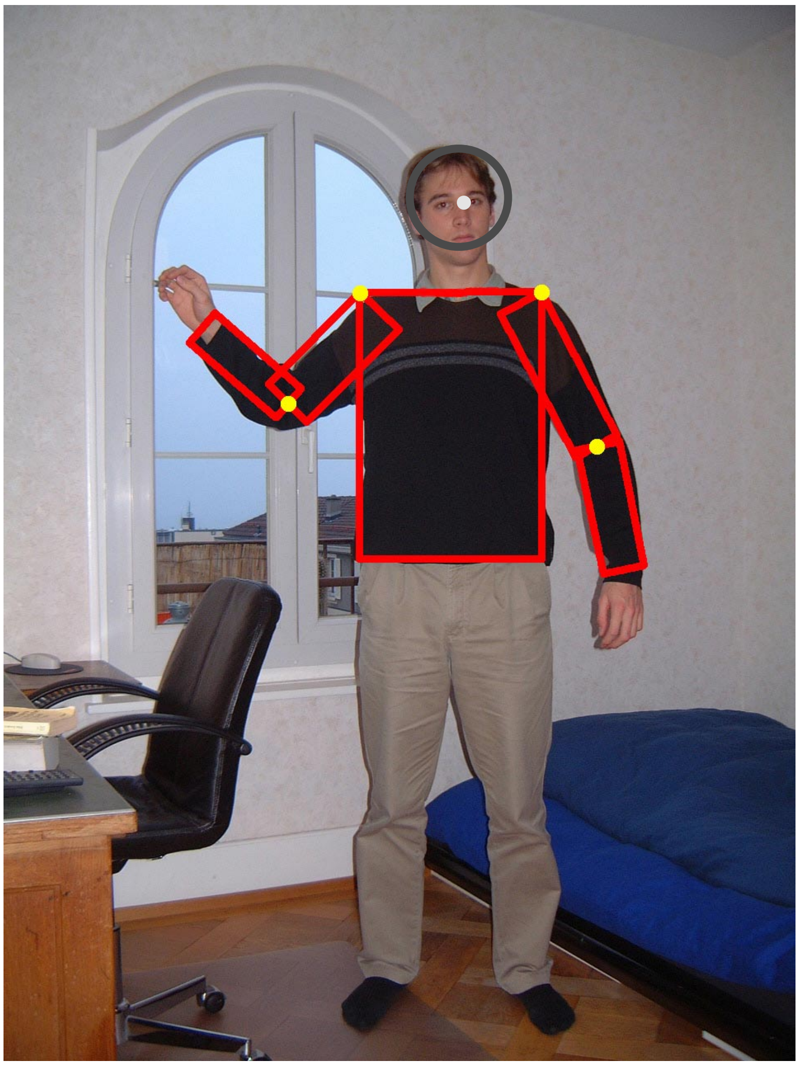

Figure 10: Model of the human being

This model is completed with information concerning consistency of the assembled parts, an arm should, hopefully, be attached to the shoulder, that means, it should be located within a certain distance to the head. This consistency information will be very human specific, that will also make the model very human specific.

Also the distance of the detected contour will be considered for adjusting the dimensions of the models features.

The model could, later on, be completed by colour information for every feature. The system could measure the colour of each feature after having detected it with certain reliability and use this information for easier finding the feature in the following images.

Each feature of the described model is defined by its size (width\height or diameter), its position within the image (coordinates of one specific point of the feature) and by its angle in the case of the rectangles.

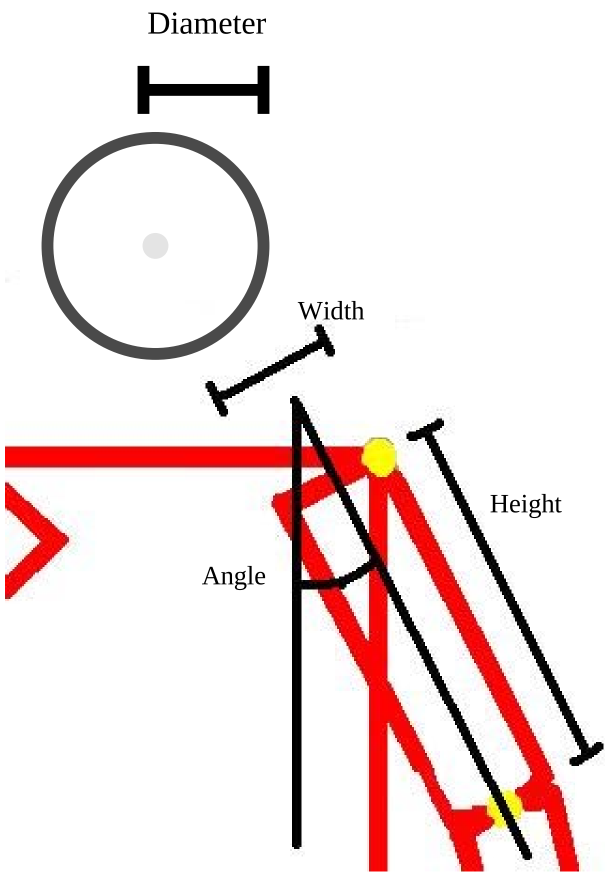

Figure 11: Features of the human model

This information will also be the output of the program; another application receiving this information periodically (knowing the time elapsed between two sets of the values) could interpret the movement according to a given context. Like that we have completely separated the context independent action of gesture detection and description and the context dependant action of gesture interpretation.

Although this is a static model of the human being, providing the interpreting application with information about the time the images where taken would make it possible to calculate also

velocity and frequency of the gestures. Adding this information to the output makes dynamic interpretation of the gestures possible.
I could also imagine adjusting the dimensions of each parameter not only accordingly to the detected distance but also accordingly to the dimensions of the detected contour after having matched the features with certain reliability.

4.2. The two program modes

The program will work using in two modes, detection mode and tracking mode.
Detection mode is used, when not all features of the human model are found with a sufficient reliability. That means, if there is no one in the field of vision, the program runs in detection mode until a person enters the scene and is completely recognised. If one part of the person is occluded the program will continue running in detection mode until it really has recognised all parts of the model.
As soon as all parts of the model are recognised the program will switch to tracking mode. In tracking mode the program simply tries to follow the objects found in the last image. For acceleration this process I will adjust the search window sizes and position in function of the features position in the last image and the position of the features already found. Is, for example, the head already found, the search windows of the upper arms can be adjusted accordingly to the heads position.
Both of the two modes will begin with adjusting the dimensions of the features of the human model in function of the distance of a detected object and finish by the same consistency check.
This two-mode algorithm combines the advantages of having an intelligent part-detection algorithm and a fast and simple tracking algorithm once the parts found in the image.
This two program modes are explained in more detail in the following two sections. The flow chart in figure 12 shows the rough program flow.

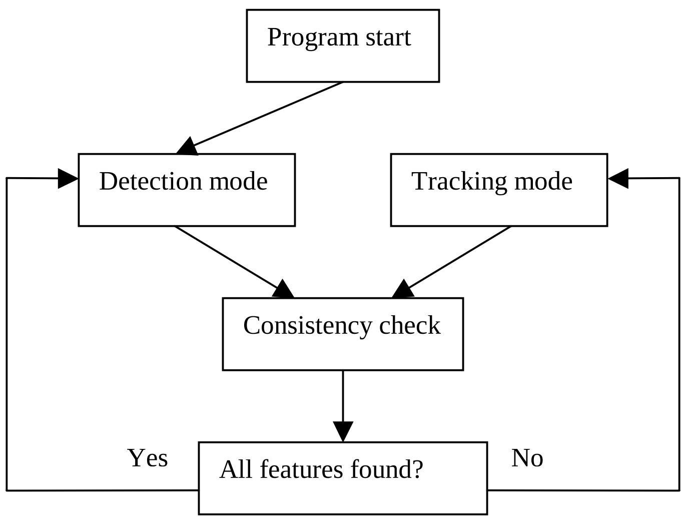

Figure 12: Program flow

4.3. Detection strategy

The system first initialises itself, that is it saves the background reference images for the background subtraction described in the following section. During this phase there shouldn’t be anyone in the field of vision.
After having done the initialisation the system waits for a person to enter the field of vision, that is it tries to detect a contour that is sufficiently big (in function of the contours distance) so that it could be a human being. Having detected such a contour the dimensions of the features of the human models are adapted to the distance of the contour and an algorithm similar to the one described in [1], but a little bit simplified, is applied.
The program tries to partition the contour using corners of the contour. The detected parts are then registered with a set of human features according to the human model. The program finally keeps the best matching combination. This algorithm works also if the number of human features and the number of segments found in the image aren’t the same.
After that, the chosen combination has to pass a consistency check, this check returns true if the combination is consistent and all features of the human model were detected. Even if the final consistency check fails, the features already found can compose an output.
This detection algorithm will run as long as the final consistency check fails.
The detection strategy is illustrated in figure 13.

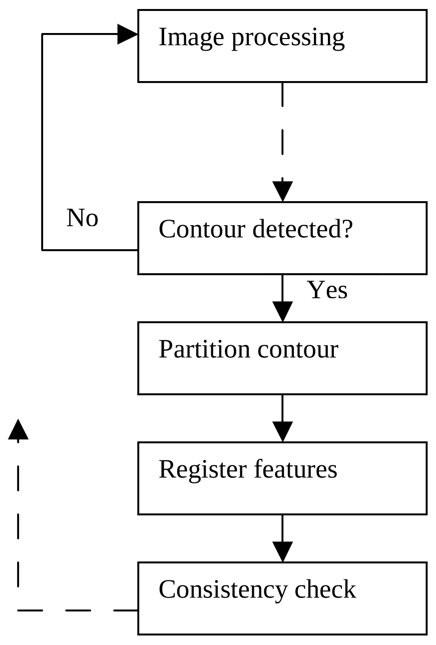

Figure 13: Detection strategy

4.4. Tracking strategy

Once the whole human model is found in a contour, the program switches into tracking mode. The tracking algorithm consists in searching for the features in regard of their position in the last image (later on perhaps also in regard of their motion, i.e. velocity and direction, in the last some images) and in regard of the constraints defined by the model (e.g. arms attached to the body, limitations in some angles).
Also the tracking will begin by adapting the dimensions of the features of the human model in function of the distance of the detected object. Then the program will search for the head in a search window centred on the position of the head in the last image and dimensioned accordingly to the distance of the object. After having found the head, the program will continue with searching for the upper arms and finally the lower arms. The dimensions of the search windows are always adjusted accordingly to the distance of the object and the position accordingly to the features already found and/or the position in the last image. Searching is always done for one feature at a time.
After having found all features the consistency is checked using the consistency check also used after detecting the features. As soon as the consistency check fails the program returns to detection mode.

4.5. Image processing

4.5.1. Background subtraction

The most powerful image processing technique used in this program is surely the background subtraction. I used background subtraction on both colour and disparity images, because background subtraction on colour images gets very unreliable when lighting conditions change, but is very accurate when the lighting conditions are stable. Unfortunately a person entering the field of vision sometimes changes lighting conditions sufficiently to get completely unusable results from colour image background subtraction (especially when he covers a light source). I found that combination of background subtraction on both images mentioned above produced the best results.

I used an algorithm similar to the one described in [8], but I do a background initialisation when the program starts, rather than using a background-learning algorithm. The field of vision shouldn't contain anything than the not moving background when the program takes the first three pictures. The mean value of those three pictures will be the reference picture until the next initialisation. In the case of the colour images the background is subtracted on normalised colour images because they are much less sensitive to lighting changes.

The background subtraction in the case of disparity images is a little bit different. I do a first subtraction considering the difference value as zero in case there was not enough texture to compute a disparity value for a given pixel. Then I check all the pixels that had in the reference image the error value but another value in the actual image. Each one of them, which has approximately the same value (in the present version +/- 5) as some of his adjacent pixels (in the present version three out of eight), keeps his value and will like that be considered as changed. This method helps preventing from groups of pixels with unreliable values within the human body.

Finally I take as foreground all the pixels, which were detected by one of the two methods as foreground.

I could also imagine using a background-learning algorithm, like the one described by [8] for initialisation. In this case the system could do the initialisation even if someone is in the scene. During this process the program would take some hundred frames and calculate the background on the base of those frames; that would take about one minute, but it would also be more comfortable in use, especially in the playroom example.

4.5.2. “Closing”

Another important image processing technique I used is “closing”. I used this technique for reducing noise after the background subtraction. This technique consists basically in taking the mean value of the pixels in squares and assigning this value to the pixels in these squares. I also used the two ingredients of “closing”, “erode” and “dilate”, separately, but using them separately normally changes the dimensions of the objects in the images, a very unwanted effect in regard of the human model, heavily depending on features dimensions, I’ve chosen.

4.5.3. Colour filtering

I haven’t yet used any colour filtering, because of the hypotheses I made (see section 2.3.). But as I also mentioned it could be used to make tracking more stable and more reliable, once the features detected with sufficient reliability (see section 4.1.). Tracking could be made more stable using the colour information of each feature to track. This colour information could also be updated regularly, for example every some hundred images. Colour filtering consists in just keeping pixels with values within a specific range; it’s usually done in normalised colour space.

4.6. Program flow

In figure 14 I show the complete program flow. It resumes all the things mentioned in this section and shows graphically the expected behaviour of the program. This program will provide an intelligent detection algorithm, which works more or less independently of the environment (no special cloths to wear, no constraints on the background, no skin colour has to be visible), as well as a tracking algorithm, which I expect to be fast.

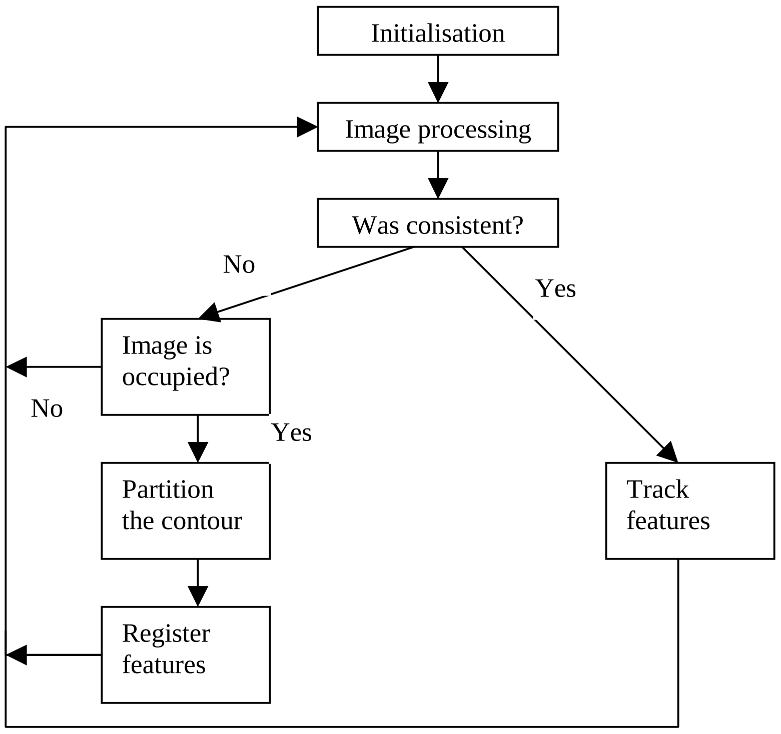

Figure 14: Flow chart of the program flow

5. Implementation

The program is essentially based on S. Granges TLIB. TLIB is a C++ library providing functions for accessing the frame grabbers, for doing some image treatment (e.g. format conversions, filtering, extraction of edges, etc), for calculating the disparity image, but also for finding shapes within images. TLIB does not directly implement tracking functions but it provides the user with the basic functions to implement tracking functions.
The implementation was done in C++ on a standard personal computer (Intel Pentium 4, 1800 MHz) using Microsoft Visual Studio.

5.1. Program structure

In this section I will describe the classes I implemented and I will give reasons why I chose to implement them this way.

5.1.1. The class tlGeneralRect

```
lass GeneralRect : public tlRect
-----------------------------------------------------------
int angle

tlGeneralRect  ()
tlGeneralRect  (int x, int y, int angle, int width, int height)

void set (int x, int y, int angle, int width, int height)
void rotate (int dangle)
void setAngle (int angle)
void setLocation (int x, int y)
void setDim (int width, int height)
void resize (int dwidth, int dheight)
void copyTo (tlGeneralRect *gen_rect)
void add (tlImage *image, tlPixel val)
void addFill (tlImage *image, tlPixel val)
void toMask (tlMask *mask)
void changeCoordSyst ()
```

Figure 15: Class tlGeneralRect

The class tlGeneralRect is a class which inherits from TLIB's class tlRect. TlRect is a class describing rectangles and providing methods to handle these rectangles unfortunately these rectangles are always aligned parallel to the image borders; they are therefore unusable for describing the upper and lower arms in my human model.
TlGeneralRect extends tlRect by an integer called "angle", which represents the angle shown in figure x, measured in degrees. Adding this angle already suffices to represent upper and lower arms of my human model.
I rewrote most of the methods of tlRect to work with the additional variable angle. I also added two methods, which draw the rectangle in the image passed as parameter. I added them directly in the class tlGeneralRect, because I did not want to change the source code of tlImage.

5.1.2. The class tlCircle

```
class tlCircle
```

```
int x;
int y;
int diam;

tlCircle ();
tlCircle (int x, int y, int diam);

void set (int new_x, int new_y, int new_diam);
void set (int new_x, int new_y);
void resize (int new_diam);
void shift (int dx, int dy);
void copyTo (tlCircle *circle);
void add (tlImage *image, unsigned char r, unsigned char g, unsigned char b);
void addFill (tlImage *image, unsigned char r, unsigned char g, unsigned char b);
```

Figure 16: Class tlCircle

The class tlCircle describes a circle, the object I chose to represent a humans head. The class consists of three integers representing the coordinates of the circles centre and the circles diameter.
I implemented constructors and methods to set the parameters, to resize and to move the circle, to copy the circle and to draw the circle in an image.

5.1.3. The class tlHuman

The class tlHuman is the main piece of the code. It is the class that represents a human being and that implements also the tracking strategy. I decided to join the human model and the tracking strategy in one class, because they are in the case of this project very closely related (the tracking strategy is based on the human model).
The class however does not implement any method concerning interpretation the tracking output.
TlHuman also includes methods that do all the image processing, the contour detecting and partitioning methods and the tracking methods.
TlHuman contains all images constructed during image processing.

```
class tlHuman
```

```
private:

tlCircle        *head
tlRect          *h_search
tlGeneralRect   *left_upper_arm
tlRect          *lua_search
tlGeneralRect   *right_upper_arm
tlRect          *rua_search
tlGeneralRect   *left_lower_arm
tlRect          *lla_search
tlGeneralRect   *right_lower_arm
tlRect          *rla_search
tlGeneralRect   *body
tlRect          *b_search
tlImage         *color_ref_image
tlImage         *disp_ref_image
tlImage         *color_image
tlImage         *disparity_image
tlImage         *contour_image
bool            consistent
int             rect_dim

void subtractBackground (tlImage *left_color_image, tlImage *right_color_image, tlImage
*disp_image)
bool isConsistent ()
void trackFeatures ()
void findAll ()
int   partitionContour ()
bool occupied ()
void setFeatureSize (int distance)
int   getDistance ()
bool isWhite (int x, int y, int dim)
void combine (int nb_segments)
void searchContour (int& counter, int x, int y, int dim)
void searchEnv (int x, int y)

public:

tlImage   *color_image
tlImage   *background_image
tlImage   *safety_image
bool      background_detected

tlHuman ()
~tlHuman ()

bool track (tlImage *left_image, tlImage *right_image, tlImage *disp_image, tlCircle *head,
tlGeneralRect *left_upper_arm, tlGeneralRect *right_upper_arm, tlGeneralRect *left_lower_arm,
tlGeneralRect *right_lower_arm, tlGeneralRect *body)
void getDispBackground  (tlImage *disp_ref_image1,  tlImage *disp_ref_image2,  tlImage
*disp_ref_image3)
void getColorBackground (tlImage *color_ref_image1, tlImage *color_ref_image2, tlImage
*color_ref_image3)
```

Figure 17: Class tlHuman

5.2. Expected behaviour

The expected behaviour is in fact the one described in section 4.2.. During normal operation (all features detected) the program should only run in tracking mode. This mode shouldn't be too much resource consuming; the tracking should be quite fast. As soon as the tracker looses some features (or even the whole person) the program should try to detect the lost features again using the image partitioning and registering methods (switching to detection mode). This process will be more time consuming; the program will run slower. If the person leaves the field of vision, the program will only run the image processing and contour detecting methods and restart detecting the person, using contour partitioning and registering, as soon as the contour detection method detects a contour that could represent a human being.

5.3. Problems and their solutions

5.3.1. Calibration of the stereo cameras

One of the problems during the initial phase of the project was the calibration of the stereo cameras.
Stereo systems still are very sensitive on what concerns geometry, if the cameras are not very accurately aligned the calibration program does not manage to detect the relative position of the cameras and the stereo algorithm won't be able to compute a reliable disparity images (very noisy disparity image). One of the problems using stereo vision is that the cameras aren't able to determine disparity within areas with no texture (as for example a wall). The stereo algorithm takes a pixel in the image produced by one camera and tries to find the corresponding one in the image produced by the second camera. Knowing the distance between these two pixels (within the image) and the relative position of the cameras the stereo algorithm is able to calculate the distance between the cameras and the object to which the pixel belongs. Isn't there any texture the algorithm isn't able to find the corresponding pixel in the second image, because there are many pixels with the same value.
If the system wasn't able to calculate the relative position of the cameras during calibration, the disparity values won't be correct.

After finally having managed to do the calibration the cameras produced a quite reasonable disparity image.

5.3.2. Narrow field of vision

As the field of vision of the cameras is quite narrow, there remained some quite large regions on the human body where the stereo processing method couldn't determine the disparity. These regions combined during background subtraction with the undetectable regions of the background behind the human body. The consequence of that was, that in the image after background subtraction still persisted large regions where no difference was detected. These regions are situated within the human contour.
For solving this problem I implemented a function which tries to fill enclosed regions that weren't detected. After that, detected contours resemble quite well human contours above all when the person is at a distance of at least two meters.

5.3.3. Camera saturation

Another problem was camera saturation. The cameras used to saturate already at normal indoor lighting conditions. Although the contour extraction worked well one evening, results could be completely different the next morning. Above all skin parties of the human being proved to reflect incoming light quite strongly. The consequence of that is, that there can't be calculated any reliable disparity image and that makes any reliable background subtraction nearly impossible.
Just closing the camera lenses isn't possible, because the calibration had to be done with fully opened lenses. Therefore the disparity image won't be reliable if the lenses of the two cameras don't have exactly aperture.
The only solution I found to solve that problem was to darken the room.
The cameras also proved quite sensitive to lighting changes. Already a person entering the field of vision or just passing by can change the lighting conditions sufficiently that any background subtraction using colour images becomes impossible. A solution to that problem is to convert the colour images to normalised colour for background subtraction and to use a combination of background subtraction on both disparity and colour image.

5.3.4. Trade-off between noise reduction and image resolution

All methods I tried for eliminating noise had one thing in common: they reduced the image resolution. That wouldn't really be a problem if there weren't already the holes I described in section x within the human body. The combination of bad resolution and not closed contour sometimes happened to cut the human body in two.
Also the number of times noise reduction techniques can be applied is often very limited, because they slow down the system considerably.
For solving this problem it could be considered applying filters just to parts of the image, but that would mean that I already know the position if the noise within the image.
I finally found parameters that worked well with the filters and combined with the advanced background subtraction algorithm I managed to produce quite noise-less images.

6. Results

In this section I will present the results of my project. Unfortunately I can't present many results visually because I didn't really manage to finish the implementation of a function situated in a key position. But I nevertheless can show the results of some parts of my implementation.

Representation of the human model:

The two classes I used for representing the human model, tlGeneralRect and tlCircle, work well. The implemented functions work well and the objects can be displayed in images.

Background subtraction:

I implemented a background subtraction method, which works well, that is it manages to extract the foreground even on a background having very similar colour as the foreground. Figure 18 shows an example of an image before and after background subtraction.

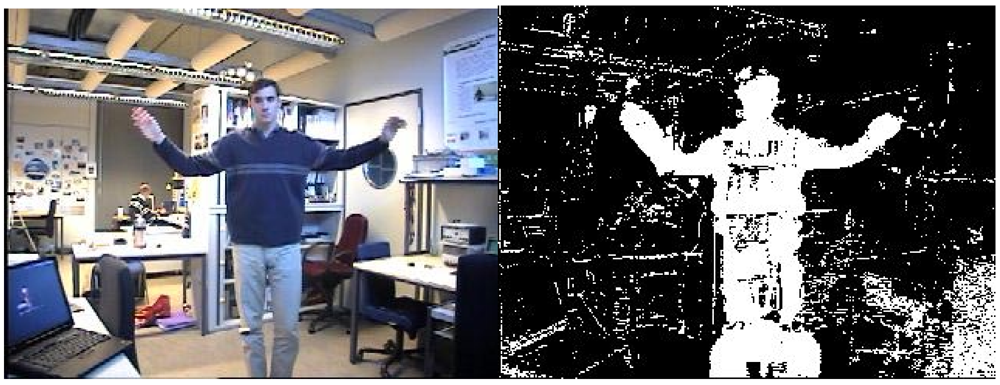

Figure 18: Image before (left) and after (right) background subtraction

As we can see there is almost no undetected area within the human body left. The algorithm also manages to detect parts of the human being, which have almost the same colour as the background (e.g. head).

Contour detection:

As the background detection works well, the extraction of the human contour seems to be quite easy. There were nevertheless some problems, above all because the human contour is often divided in two parts by the noise in the image.
Another problem was to decide which parts of the foreground really belong to the human being.
The results of the contour detection are shown in figure 19 and 20. The images show the input image, the image after background detection and the detected contour.

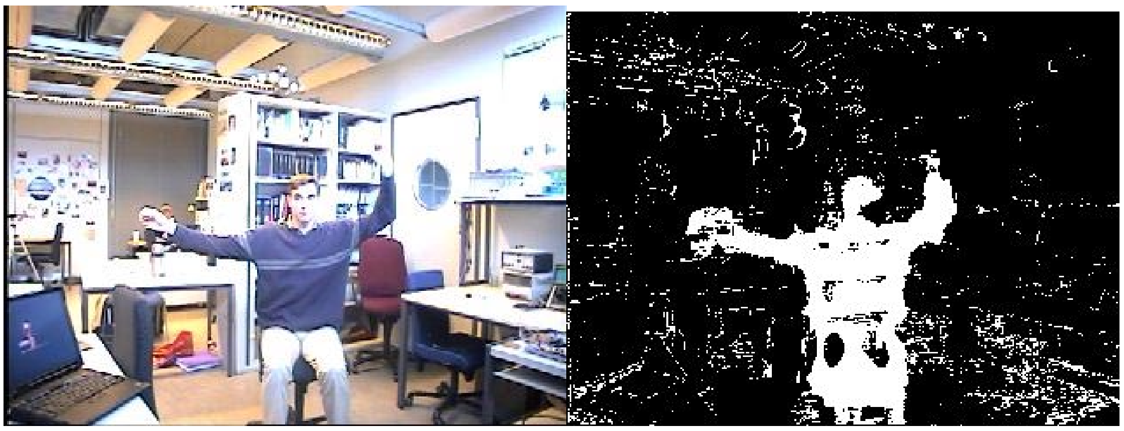

Figure 19: Input image, background subtracted

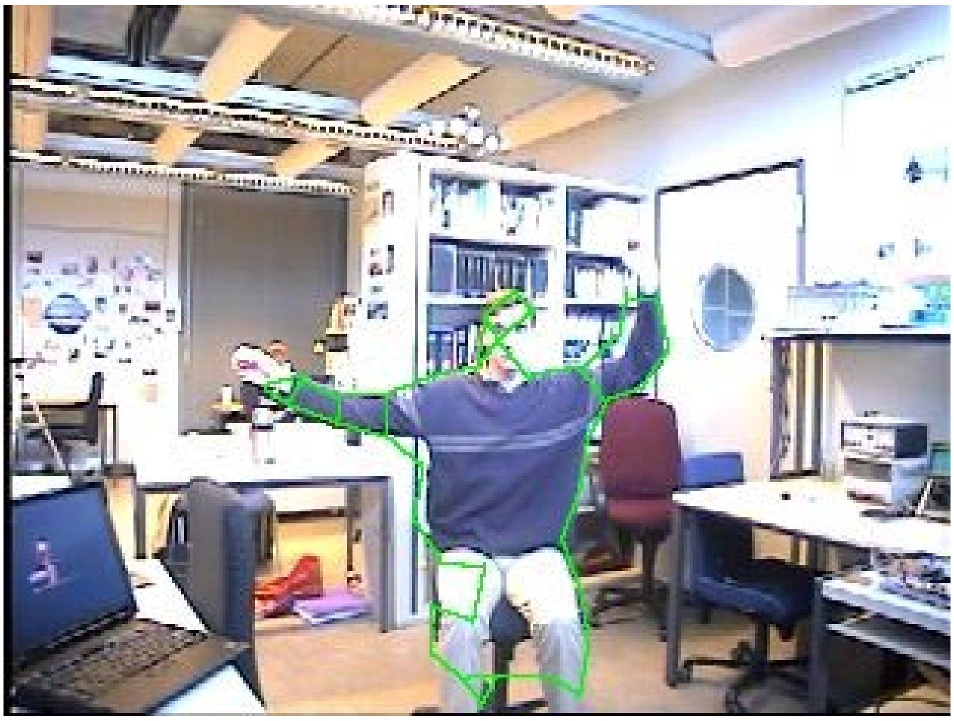

Figure 20: Contour detected

As it can be seen, the background subtraction works also well, when the cameras are already saturated in some areas. The contour segmentation on the contrary doesn't detect really the whole human being, the hands, for example, aren't detected. That doesn't really hinder the gesture detection because the model doesn't describe the hands.
In figure 21 I show the contour detected after in the images of figure 18.

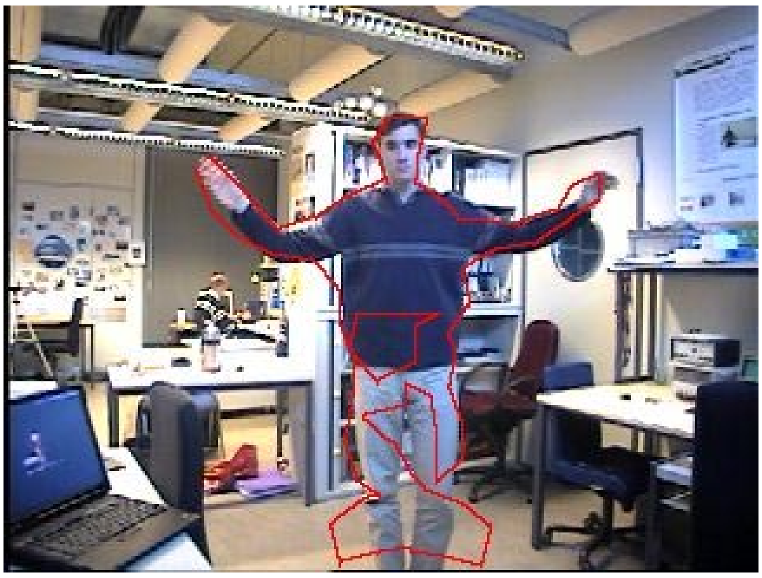

Figure 21: Contour detected in the images of figure 18

As the conditions are better in this case the contour detection manages to extract the whole human contour.

Performance of the system:

I measured the refresh rate of the system with and without object in the images. Without object the program has a refresh rate of 22 frames per second and the CPU usage at 85% to 90%. That means the system runs with the maximum refresh rate.
With a person in the images the CPU usage is at 100% and the refresh rate of the system is about 10 frames per second. As we can see does the image processing done for detecting the contour remarkably slow down the system.
If the lighting conditions are good (no camera saturation) and there is just one person in the field of vision the system managed always to extract the persons contour, sometimes overlooking the head and/or the hands.
Changing lighting conditions sometimes remarkably changed the performance of the contour detection.

7. Future work

7.1. Parts not yet finished

7.1.1. Implementation

I did not really manage to finish the implementation. There rests mainly one function to complete, the function doing the contour partitioning, “tlHuman::partitionContour”. I already managed to extract a quite reliable contour, but I couldn’t partition this contour.

7.1.2. Testing

I implemented all the other functions, but I couldn't test all of them because the output of the contour partitioning function missed. There are also some parameters to adjust in these functions.
The untested functions are namely: "tlHuman::isConsistent", "tlHuman::trackFeatures", "tlHuman::setFeatureSize", "tlHuman::combine", "tlGeneralRect::toMask", "tlGeneralRect::changeCoordSyst", "tlCircle::toMask".
After having finished the implementation of the contour partitioning function, the whole system could be tested and all the parameters used could be adjusted.

7.1.3. More dynamic limit between tracking and detection mode

In the moment the system switches only to tracking mode, once all features of the human model are found. This switching could be made more dynamic in detecting only the features not detected in the last images and tracking the features already detected.

7.2. Features that could be added

In this section I will mention some of the features that could be added to the program to build a complete smart environment. Building a complete smart environment for this program was, naturally, not possible even with today's affordable computing devices.

7.2.1. Voice detection and recognition

One of the most important parts of a natural Human-Computer Interaction would be voice detection (for people recognition) and, in a later stage, voice recognition (interaction with the computer using voice commands).
Often human beings also use their voice to explain gestures or gestures to explain what they said (i.e. "look at that" and pointing in a direction). A combined interpretation of voice and gestures could prove to be a very powerful tool in Human-Computer Interaction.
But it will be very difficult to interpret combinations like the one mentioned above, because adding voice information can sometimes change the meaning of gestures completely.

This feature could also be easily included in tlHuman by adding methods which recognise voice commands. The recognised commands could be added to the output produced by tlHuman and could also be interpreted by the same application interpreting the rest of the output.

7.2.2. Human activity monitoring

Another feature for completing a natural Human-Computer Interaction system is human activity monitoring. Methods detecting the general degree of activity of the person tracked could be added to tlHuman. The degree of activity could be classified and also added to the output generated by tlHuman. This additional information would help the interpreting application to better understand the gesture and voice information detected, by providing information about the human beings preoccupation. Adding this context information would help the application for example to decide whether to interrupt the human being or not.

7.2.3. Position detection

The last additional feature I will describe here is position detection with respect to the environment. A method detecting the position in space (relatively to known landmarks) of the tracked person could also be added to tlHuman. This additional information would also help the interpreting application to better understand gestures, by adding context information. Knowing if the human being is in front of the telephone or near a piece of furniture would also provide information about the human beings preoccupation.

7.3. Adaptability to other problems

The system could be used in almost any area viewable with fixed cameras.
It could for example be used in manufacturing plants for tracking people working near machines. That could help to avoid dangerous situations by adapting the machines activities to the humans' activities.
The system could also be used throughout a whole apartment not only in the playroom. The gesture detection and recognising program will be the same in every room, but the interpreting program will interpret the gestures in function of the room where they were

detected (i.e. the same gesture closing the curtains in the living room would stop the water tab, when exercised in the kitchen.
Another idea is to use the system in apartments of elderly people, because they often aren't able any longer to do every thing on their own. The system would help them to stay independent.

8. Conclusion

At this stage of the project I can clearly say that vision is a very powerful tool for enhanced Human-Computer Interaction. Vision allows a more natural communication between human beings and computer devices.
Vision is very powerful, used for activity monitoring (passive vision based control, see section 2.1.1.) to provide the computer with additional information about the human beings preoccupation, but also used for gesture driven control (active vision based control, see also section 2.1.1.).
This project was situated more in the active vision based control area and I think I can say that it is possible to develop a vision-based system, using stereo vision, which is able to detect and to recognise human gestures in real-time.
I designed a human model and I described in this report a detection and tracking algorithm, which would make the implementation of such a system possible.
This implementation proved more complicated than I thought. The first problem was the calibration of the stereo cameras; these stereo cameras depend still on a very accurate alignment and a precise calibration.
Then there was the handling of the noise in the image; often it was quite complicated to decide whether something is noise or important information.
It was also quite an experience to see that things which are very clear for our sense of vision aren't that easy to describe for a computer. The partitioning of the contour, for example, seems for our eye very easy, but it isn't that easy at all to describe this partitioning in rules that can be handled by the computer.
But nevertheless, the performance measures I've made indicate, that it should be possible to finish the implementation and to have a system tracking human features in real-time.

9. References

[1] L. Zhao, C. Thorpe: “Recursive context reasoning for human detection and parts identification”, IEEE Workshop in human modelling, analysis, and synthesis, June 2000.

[2] R. Cutler, M. Turk: „View-based interpretation of real-time optical flow for gesture recognition”.

[3] N. Jojic, M. Turk, Th. S. Huang: “Tracking self-occluding articulated objects in dense disparity maps”, IEEE international conference on computer vision, 1999.

[4] S. Grange: “Vision-based sensor fusion for active interfaces: H.O.T – Human Oriented Tracking”, 2000.

[5] M. Turk: “Visual interaction with lifelike characters”, IEEE computer society press, October 1996.

[6] A. Mulder: “Human movement tracking technology”, 1994.

[7] H. Sawada, S. Hata, S. Hashimoto: “Gesture recognition for human-friendly interface in designer – consumer cooperate design system”, IEEE international workshop on robot and human interaction, 1999.

[8] G. Gordon, T. Darrel, M. Harville, J. Woodfill: “Background estimation and removal based on range and color”, IEEE conference on computer vision and pattern recognition, June 1999.