Appendix A

Experiment

We used a uniformly colored background behind the subject. We traced three lines but the subject had to follow the middle one. The other two allowed the subject to have a better reference of the central one.

An orange sphere, situated at the subject's wrist, must be kept in the line. A video camera recorded a side view of the subject. The subject did not look at the line directly but at a TV monitor placed in front of him/her. In front of the subject there was a mirror so that he could check that the motion was in the sagital plane. The purpose of the mirror was that the subject could prevent movements like abduction.

Experimental Settings

This planning needed a video camera, a monitor, a mirror, a chair and a smooth surface where the line that the subject had to follow was situated. The experiment was carried out in a room with the following description (see Figure A.1, Figure A.2):

Camera: 2.5 m to the wall

Monitor: 1m to the subject

Temperature: 24ºC

Uniform Artificial Illumination, Uniform grey paint on wall

Figure A.1. Experiment settings

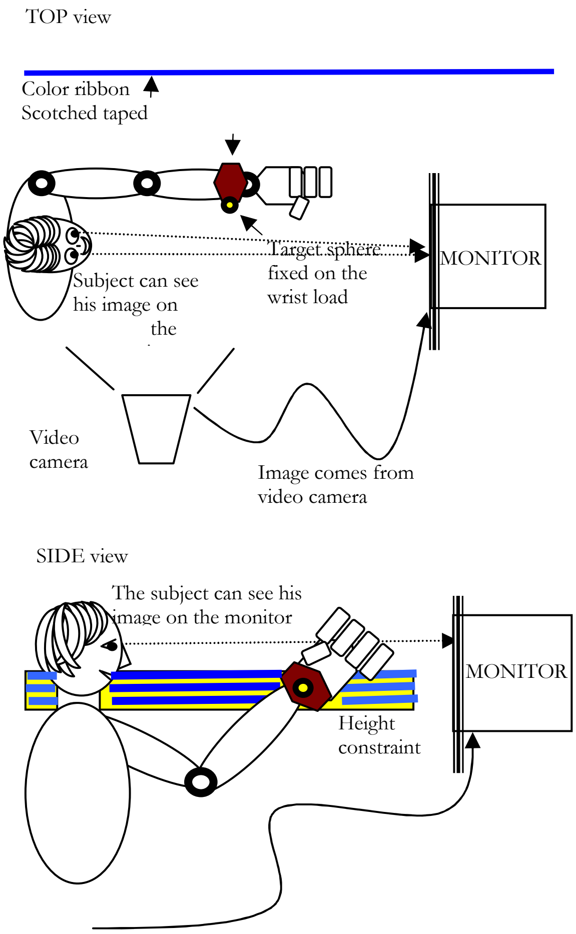


Figure A.2. Top and side view of the experiment

Protocol

The experiment consisted of three well differentiated phases: explanation and performance of a previous training experiment, performance of the definitive experiment and finally get feedback from the subject.

The training test took short time and the subject did not carry load. He only had to explore the middle line using the orange sphere. The objectives were that the subject became familiar with the experiment settings (monitor, mirror, etc) and also to define the initial posture by placing a red marker in the initial position.

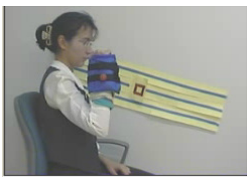
In the training phase, the instructions given to the subjects were: "Stand/sit comfortably because only the arm should move. Your back must be in slight contact with the surface behind it; this is to avoid back movements. Use your right arm to follow a line with the orange marker slowly. Look at the monitor so that the orange marker stays on the line. Adjust the arm in the mirror so that it moves in a vertical plane. Freely explore the middle line slowly during one minute".

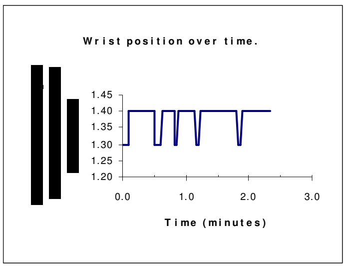
Pictures

In the following, we show pictures taken from the experiment (see section 6.2.1.1)

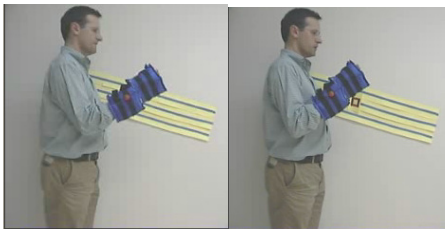
OBLIQUE (-25º)

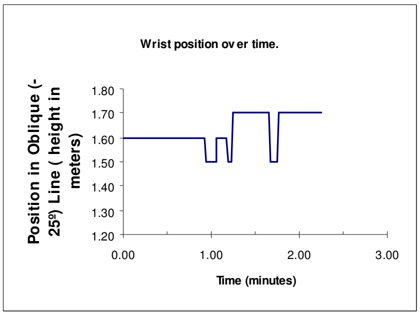
Subject 1. Load: 2.5 kg.
Initial posture Recovery found



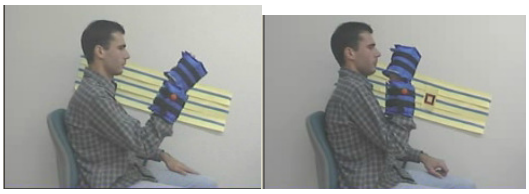


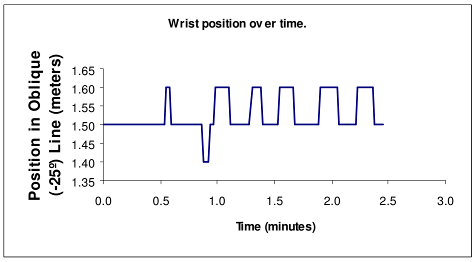
Subject 2. Load 5kg.
Initial posture Recovery found





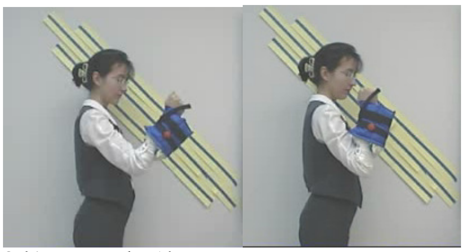
Subject 3. Load 5 kg.
Initial posture Recovery found

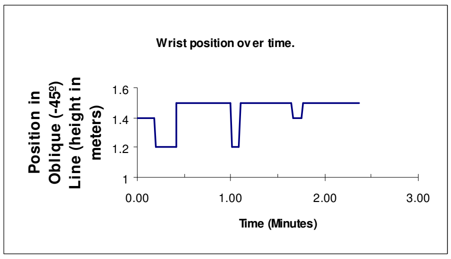




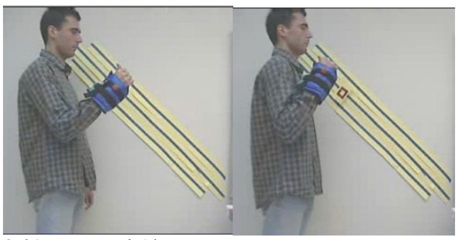
OBLIQUE (-45º)

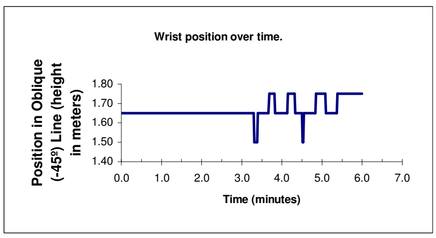
Subject 1. Load 2.5Kg.
Initial posture Recovery found



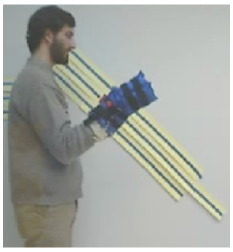


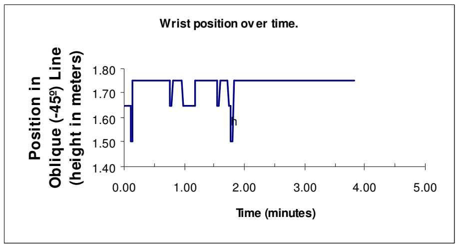
Subject 2. Load 2.5 kg.
Initial posture Recovery found





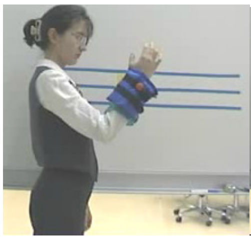
Subject 3. Load 5 kg.
Initial posture Recovery found

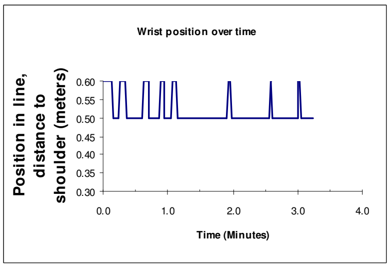




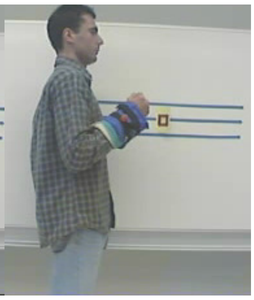
HORIZONTAL

Subject 1. Load 2.5 kg.
Initial posture Recovery found





Subject 2. Load 2.5 Kg.
Initial posture Recovery found



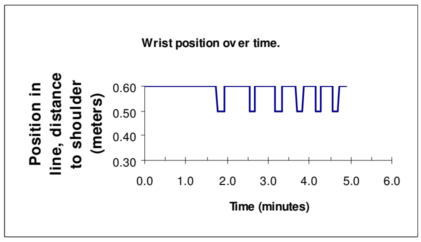


Figure A.3. Some of the pictures taken from the experiment

Appendix B

Contraposto Pictures

In the following we show different contraposto poses taken from the art history and daily adopted.

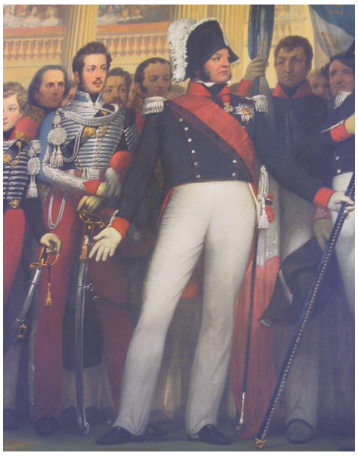


Figure B.1. Contraposto poses

Appendix C

Approximating Volumes using Box Trees

We have designed a data structure that let us approximate a certain volume. It is called Volume Approximation Tree (VATree) and it is used to represent human reachable regions. The proposed data structure is an octal tree (nodes have eight children). Octal trees (usually referred to as *octrees*) are commonly used for representing volumes or surfaces.

Each node in a VATree stores the parameters of its box (origin, width, height, depth) and its type. There are four types of nodes (see Figure C.4):

Inner node(I): The eight vertices of its box are inside the volume that is being approximated. We assume that if the eight vertices can be reached, then the entire box is reachable.

Outer node(Out): Its box is totally outside the volume.

Parent node(P): Some of the vertices of its box lay inside the volume and the rest lay outside. To achieve a tighter approximation of the volume, parent nodes are subdivided yielding a set of eight child nodes or *subnodes*.

Final nodes(F): They are related to parent nodes as they lay on the surface of the volume as well. However, they are treated differently because their depth in the tree is maximum, and thus they can't be subdivided.

Note that only nodes of parent type have pointers to its eight child nodes. Children are obtained by subdividing their parent's box.

Construction Procedure

The following parameters are necessary for constructing a VATree:

- Dimensions and position of the initial box B_0. This box must be chosen so that it completely contains the volume being approximated.

- Maximum tree depth, D_{max}.

- Minimum tree depth, D_{min}.
- V, the volume to be approximated.

Given a certain node N_i, the following algorithm shows how to construct the branch hanging from N_i:

1. If the current depth D is less than D_{min}, go to step 7.
2. Compute the position of the eight vertices of N_i's box.
3. Apply the function *isPointInside* to those eight vertices, to find whether each one lies inside or outside V.
4. If every vertex is inside V, label N_i as an INNER node and finish.
5. If every vertex is outside V, label N_i as an OUTER node and finish.
6. It the current depth equals the maximum depth, label N_i as a FINAL node and finish.
7. Label N_i as a PARENT node and allocate memory for its eight child nodes. Repeat from step 1 eight times, taking as root each of those eight child nodes

.

Figure C.1. VATree construction Algorithm

As many vertices are shared by different subnodes, many calls to *isPointInside* are redundant. We have provided a mechanism in which, during its subdivision, each parent node calls *isPointInside* for the different vertices of its descendants, storing the result so that this information can be used when building the child nodes. This way, only one call is made for each different vertex.

The success of the algorithm strongly depends on how correctly the initial box B_0 is selected. Extreme situations can arise, such as those shown in Figure C.2. The solution to this problem is based on forcing the tree to have a minimum depth D_{min}. During the tree construction, all nodes laying at depths greater than D_{min} are unfailingly labeled as parents and subdivided, regardless of their location in relation to the volume.

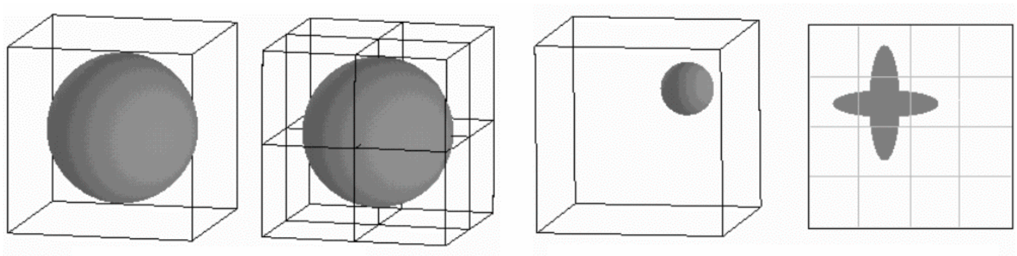


Figure C.2. Initial Box

(a) A case in which is needed Dmin=1 (b) A case in which D_{min}=1 is also needed (c) The box is so big that D_{min}=1 is not enough (d) A complex volume can make D_{min}=2 insufficient

Several factors must be taken into account when choosing an appropriate value for D_{min}:

- The size ratio of the initial box B_0 and the volume V. In the example of Figure C.2.a, one subdivision sufficed because B_0 was chosen with a similar size to V's. In Figure C.2.c, however, B_0 is too big when compared to V and thus D_{min} should be greater than 1 for the procedure to work as intended.

- The complexity of the volume V. In Figure C.2.d, for instance, the volume requires $D_{min} > 1$ because after the first subdivision all sampling points lay in its exterior. The construction algorithm works better when V is a simple volume, such as a sphere. When this is not true, it is recommendable to choose a greater D_{min}.

Construction of an Example Tree

A two-dimensional version of the algorithm provides an intuitive representation. In such a case, each parent node has only four children, and the tree approximates an area instead of a volume.

Note that this simplification implies no loss of generality for symmetric volumes such as a sphere. Figure C.3 illustrates the construction of a tree that approximates a circular area, with D_{min}=1 and D_{max}=3.

In Figure C.3.b the initial box B_0 can be seen, as well as the circle being approximated. Note that B_0 has been properly selected so that it tightly fits the circle. In the first iteration of the algorithm the root node is labeled as PARENT, despite all of its vertices lying outside the circle. This is correct because the depth D is 0 at this stage, lesser than D_{min}.

Figure C.3.b depicts the situation after subdiving the root node. This subdivision yields four subnodes of depth D=1, which are all labeled as *parent* because they have some vertices inside the circle and others outside. It should be remarked that to label these four nodes no calls to *isPointInside* are performed, because all relevant information has been previously collected by their parent.

As seen, the four nodes that resulted after subdividing the root node are themselves *parent* nodes. Thus they must be subdivided, giving the situation shown in Figure C.3.c, this figure shows the sixteen nodes of depth D=2. Four of these nodes are classified as inner and won't be further subdivided. The remaining twelve nodes have vertices inside and outside the circle, and thus are labeled as parent nodes and are subdivided.

After performing the second subdivision, the situation is as shown in Figure C.3.d. Note that nodes labeled *inner* in the previous step are now absent, since they were not subdivided. Also note that from all nodes of depth three, 28 are labeled *outer* because they are completely outside the circle. The remaining nodes are labeled *final* since they belong to the frontier of the circle (i.e. some of their vertices lay inside an others lay outside) but can not be subdivided, being their depth equal to D_{max}.

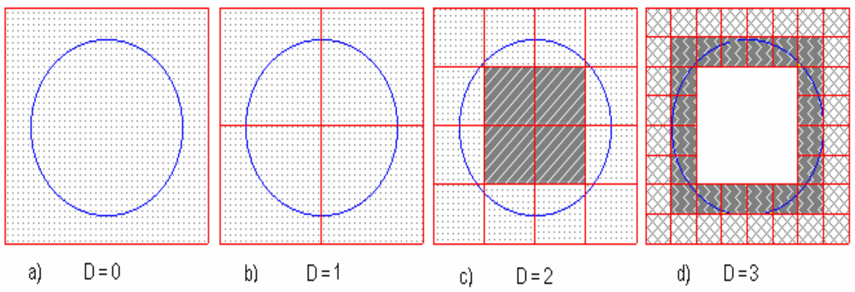


Figure C.3. Construction of a 2D version of the tree

The resultant tree is represented in Figure C.4. Some branches have been omitted for the sake of clarity. In Figure C.4 nodes labeled as *P* are Parent nodes, *Inn* are Inner nodes, *F* are Final nodes and *Out* are outer nodes.

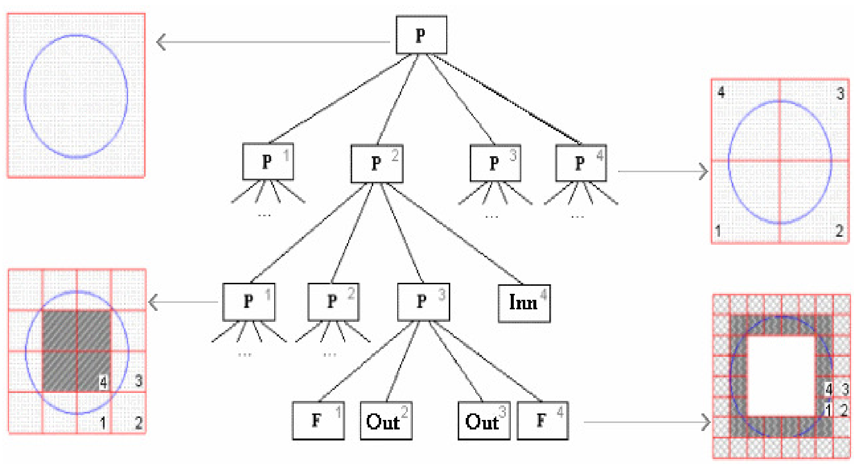


Figure C.4. Structure of an example tree