Proceedings of IEEE-RAS/RSJ International Conference on Humanoid Robots (Humanoids 2004)
November 10-12, 2004, Los Angeles, California, USA

A FULL BODY HUMAN MOTION CAPTURE SYSTEM USING PARTICLE FILTERING AND ON-THE-FLY EDGE DETECTION

PEDRAM AZAD [1,2]
azad@ira.uka.de

ALEŠ UDE [2,3,4]
aude@atr.jp

RÜDIGER DILLMANN [1]
dillmann@ira.uka.de

GORDON CHENG [2,3]
gordon@atr.jp

[1] *IRF, Universität Karlsruhe*
Haid-und-Neu-Strasse 7, D-76131 Karlsruhe, Germany

[2] *Department of Humanoid Robotics and Computational Neuroscience, ATR CNS*
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0288, Japan

[3] *ICORP Computational Brain Project, Japan Science and Technology Agency*
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0288, Japan

[4] *Jozef Stefan Institute, Department of Automatics, Biocybernetics and Robotics*
Jamova 39, 1000 Ljubljana, Slovenia

In this paper we present a full body human motion capture system based on particle filtering operating on monocular image sequences. Distinguishing our approach from others is that edge detection is not carried out globally on the whole picture in a preprocess manner, but is done locally on-the-fly during the calculation of the likelihood function. This approach is effective and flexible at the same time, allowing the use of various edge detection algorithms with only small modifications. Another special feature of our system is a highly optimized occlusion test based on a fast point-in-triangle test. Our system has been designed carefully with respect to serve as a basis for various research activities in the near future, including the incorporation of stereo vision. A general framework architecture for particle filters and an extension for appliance to edge tracking has been developed with which it is possible to test a whole range of search space decomposition variants with minimum implementation effort.

Keywords: Human Motion Capture; Particle Filter; On-The-Fly Edge Detection; Fast Occlusion Test.

1. Introduction

Markerless human motion capture based on image sequences is a challenge. The general problem definition is to find the correct configuration of the 3D human model for each image. The main difficulty lies in the fact that the search space increases exponentially with the number of Degrees Of Freedom (DOF) of the human model. A realistic model of the human body usually has at least 25 DOF, the model used in our system has 28 DOF.

While marker-based human motion capture is the proper technique for application in the animation industry due to its higher accuracy, the idea of a markerless human motion capture system is that it can be applied without any additional arrangements. An ideal markerless human motion capture system can be applied in any environment with sufficient lighting conditions.

A good overview of visual tracking can be found in [1] and [2]. In [3] a human motion capture system is presented which can capture human motion using three fixed positioned cameras in the corners of a room and achieves very good results. A multi-view approach is presented in [4]. In [5] and [6] systems are presented using monocular image sequences only and therefore lack the ability to capture full 3D motion. In [7] a constant angle of view of the subject is assumed, which greatly restricts the resulting trackers generality [3].

Our aim is to create a system for application on an active head for a (moving) humanoid robot, allowing the use of a stereo camera system only in which the two cameras are positioned at a comparatively small distance. In this paper, we want to present a basic system running with one camera, with several special features included such as on-the-fly edge detection, a highly optimized occlusion test and the ability to track through the elbow singularity. On-the-fly edge detection guarantees that during the search for edge features always the full information provided by both the input picture and the current configuration of the human model is applied. As will be shown this advantage comes at no significant additional computational cost.

This system is currently serving as a solid basis for various research projects on a human motion capture system to be implemented on the active head of a humanoid robot. In particular, the incorporation of stereo vision and runtime optimizations is current work in progress.

2. Particle Filters

Our system uses a Bayesian framework called *particle filtering* to capture human motion from monocular image sequences. Particle filtering is also known as the Condensation algorithm introduced by Michael Isard and Andrew Blake [8]. The Condensation algorithm was designed to track curves in clutter where Kalman filtering fails in general, since it is based on Gaussian densities which are not capable of representing simultaneous alternative hypotheses.

However the ability of the Condensation algorithm to cope with clutter is achieved at a high computational cost. The probability distribution is modeled by a set of finite particles. Each particle is a pair

$$(\mathbf{s}_\mathrm{n}, \pi_\mathrm{n}) \tag{1}$$

where s_i represents one configuration vector of the model's configuration space and π_i the probability associated with it. The dimension of configuration space is equal to the number of DOF of the object model. The probability function is then modeled as a set of a constant number of particles *N*:

$$S = \{\ (\mathbf{s}_1, \pi_1), \ldots, (\mathbf{s}_N, \pi_N)\ \}\ . \tag{2}$$

Particle filtering is an iterative process operating on this set, thus the set of particles after each iteration *k* is denoted as:

$$S_k = \{\ (\mathbf{s}_{k,1}, \pi_{k,1}), \ldots, (\mathbf{s}_{k,N}, \pi_{k,N})\ \}\ . \tag{3}$$

The general algorithm consists of four steps in each iteration:

- (i) Preprocess data
- (ii) Predict new configurations
- (iii) Apply Bayesian rules
- (iv) Estimation of the current configuration

The first step consists of preprocessing the input data. In a system for tracking curves this would usually be the application of an edge detection algorithm.

In the second step the new set of configurations (particles with no probability assigned) is predicted by incorporation of the underlying dynamic model and adding Gaussian noise to the configurations of the particles of the last generation. The particles of the last generation used for this purpose are picked proportional to their probability, which can be efficiently solved by binary subdivision.

In the third step, the new set of particles is formed by assigning probabilities to the new set of configurations. This is done by applying Bayesian rules and in particular evaluating a central likelihood function $p(\mathbf{z}|\mathbf{s})$ for each new base $\mathbf{s}_i$ predicted, where $\mathbf{z}$ denotes the output of step one. More precisely $p(\mathbf{z}|\mathbf{s})$ calculates the likelihood that $\mathbf{s}$ is the proper configuration to a given measurement $\mathbf{z}$. The estimation of the current configuration in the final step is usually done by calculating the mean configuration of the probability distribution formed by the new set of particles.

3. Particle Filtering in a Human Motion Capture System

In this section we describe how particle filters are commonly used in a human motion capture system. The components of the particle filter which are application dependent are:

- Model of the object of interest
- Likelihood function

These components will be described in the following for the specific use in a human motion capture system.

3.1. *Human model*

A common human model, as used in our system, on which the calculations are based can be seen in figure 1. The modeled body parts consist of the head, torso, upper arm, forearm, thigh and lower leg.

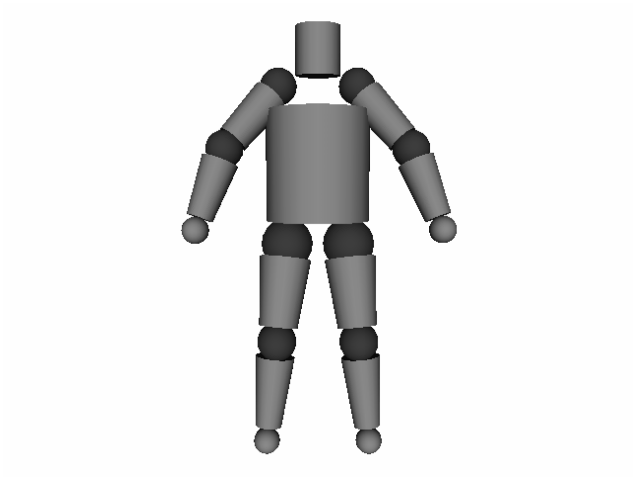

Fig. 1: 3D human model

Hands and feet are not modeled; elbows and knees are modeled as hinge joints. Thus the total number of DOF is 28:

- Basic rotation and translation: 6
- Joint for moving the upper body: 3
- Head: 3
- Shoulder Joints: 3 + 3
- Elbow joints: 1 + 1
- Hip Joints: 3 + 3
- Knee Joints: 1 + 1

3.1.1. *Modeling body parts*

Each body part is modeled as a section of a cone. Such a section is described by four parameters H, W, L and R, as illustrated in figure 2. H and W specify the radii of the base ellipse, L specifies the length of the section and R the ratio, thus the upper ellipse is described by $h = RH$, $w = RW$.

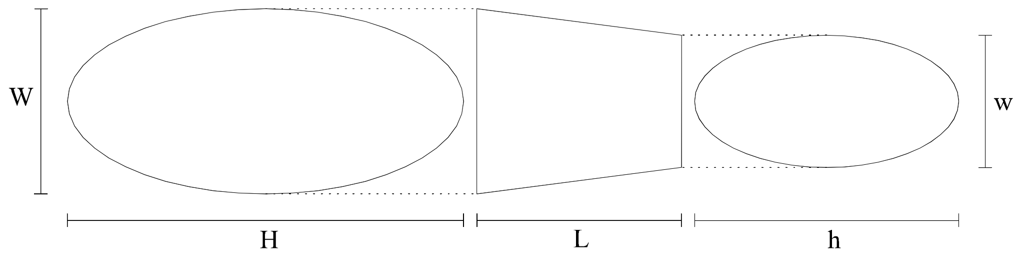

Fig. 2: 2D visualization of a section of a cone

3.1.2. *Projecting the contour*

In order to compare one configuration of the human model to a given image i.e. evaluating the likelihood function $p(\mathbf{z}|\mathbf{s})$, the contour of the human model has to be projected into the image with respect to the position of the camera. As described in [9] the occluding contour of each section is formed by two segments and can be calculated as follows.

A section of a cone is defined in the 3D world coordinate system by its origin $\mathbf{c}$, the orientation of the model coordinate system defined by the unit vectors $\mathbf{n}$, $\mathbf{u}_1$ and $\mathbf{u}_2$, and the parameters H, W, L and R which have been introduced in 3.1.1 (compare figures 2 and 3). All points on the surface of this section are defined by the equation:

$$P(\theta,t) = c + t\mathbf{n} + H(t)\mathbf{u}_1 \cos\theta + W(t)\mathbf{u}_2 \sin\theta \tag{4}$$

with

$$H(t) = H \cdot \left[\frac{(R-1)t}{L} + 1 \right] \text{ and } W(t) = W \cdot \left[\frac{(R-1)t}{L} + 1 \right] \cdot \qquad (5)$$

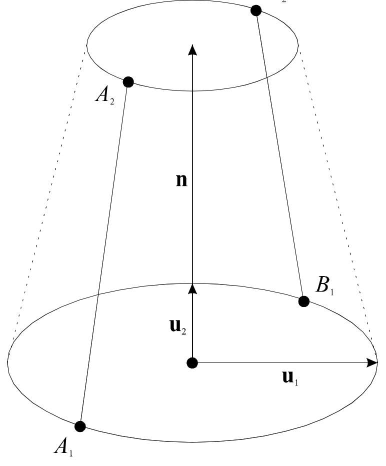

Fig. 3: 3D visualization of a section of a cone

To obtain the two point pairs A_1, A_2 and B_1, B_2 which define the two segments of interest, the following equation has to be solved:

$$H + [(\mathbf{c}-\mathbf{v}) \cdot \mathbf{u}_1] \cos\theta + [(\mathbf{c}-\mathbf{v}) \cdot \mathbf{u}_2] \frac{H}{W} \sin\theta - [(\mathbf{c}-\mathbf{v}) \cdot \mathbf{n}] \frac{H(R-1)}{L} \sin\theta = 0 \cdot \qquad (6)$$

Substituting with

$$\cos\theta = \frac{1-g^2}{1+g^2}; \quad \sin\theta = \frac{2g}{1+g^2}; \quad g = \tan\frac{\theta}{2} \qquad (7)$$

leads to the following equation of degree two:

$$g^2 - 2H \frac{[(\mathbf{c}-\mathbf{v}) \cdot \mathbf{u}_2] \frac{1}{W} - [(\mathbf{c}-\mathbf{v}) \cdot \mathbf{n}] \frac{R-1}{L}}{(\mathbf{c}-\mathbf{v}) \cdot \mathbf{u}_1} g - 1 = 0 \cdot \qquad (8)$$

By defining

$$r := H \frac{[(\mathbf{c}-\mathbf{v}) \cdot \mathbf{u}_2] \frac{1}{W} - [(\mathbf{c}-\mathbf{v}) \cdot \mathbf{n}] \frac{R-1}{L}}{(\mathbf{c}-\mathbf{v})\mathbf{u}_1} \qquad (9)$$

the two solutions g_1 and g_2 are specified as

$$g_{1/2} = r \pm \sqrt{r^2 + 1} \ . \tag{10}$$

Since both the position and the orientation of the camera relative to the base coordinate system of the human model are already taken into account by the configuration vector (base translation and rotation), therefore extrinsic calibration of the camera is not needed. Thus, the vector **v** can be set to the zero vector **0**. By doing this, the extrinsic calibration of the camera can be understood as part of the starting condition, i.e. an initial value for the configuration vector, which is discussed in section 3.3.

Using equation (10) the solutions g_1 and g_2 lead to the two angles θ_1 and θ_2 which can be used together with equation (4) to obtain the four points of interest A_1, A_2 and B_1, B_2. In the special case of $\mathbf{c}\cdot\mathbf{u}_1 = 0$, θ_1 and θ_2 are set to the values $\theta_1 = \pi$ and $\theta_2 = 0$. For obtaining the contour of the section defined by H, W, L and R the parameter t has to be set to $t_1 = 0$ for the points A_1, B_1 and to $t_2 = L$ for the points A_2, B_2. The 3D points A_1, A_2, B_1, B_2 are transformed into the image plane using the intrinsic parameters of the calibrated camera.

3.1.3. *Kinematic model*

For the implementation of the kinematic model basically any common technique known in robotics can be used such as 3×3 rotation matrices and three component translation vectors as it is the case in our implementation, 4×4 transformation matrices operating on homogenuous coordiantes or unit quaternions.

3.1.4. *Dynamic model*

Commonly used dynamic models are first order or second order models. The model is a stochastic differential equation in discrete time, which is in the first order case

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\boldsymbol{\omega}_t \tag{11}$$

where A defines the deterministic component of the model and $\boldsymbol{\omega}_\mathbf{t}$ is a vector of independent standard normal random variables scaled by the diagonal matrix B so that BB^T is the process noise covariance. A second order model is obtained by replacing $\mathbf{x}_t$, A and B by

$$\begin{pmatrix} \mathbf{x}_t \\ \mathbf{x}_{t+1} \end{pmatrix}, \quad \begin{pmatrix} 0 & I \\ A_0 & A_1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & 0 \\ 0 & B \end{pmatrix} \tag{12}$$

where I is the identity matrix. [10]

3.2. *Likelihood function*

Given the projected edges of the human model and the edge image computed from the current input image, a likelihood function $p(\mathbf{z}|\mathbf{s})$ is to be defined to calculate the likelihood that the configuration leading to the set of projected edges is the proper configuration i.e. matching the edge image the most.

The basic technique is to go through the projected edges and search at fixed distances Δ for high-contrast features perpendicular to the edge within a fixed search distance δ i.e. finding edge pixels in the generated edge image as illustrated in figure 4 [8], [3].

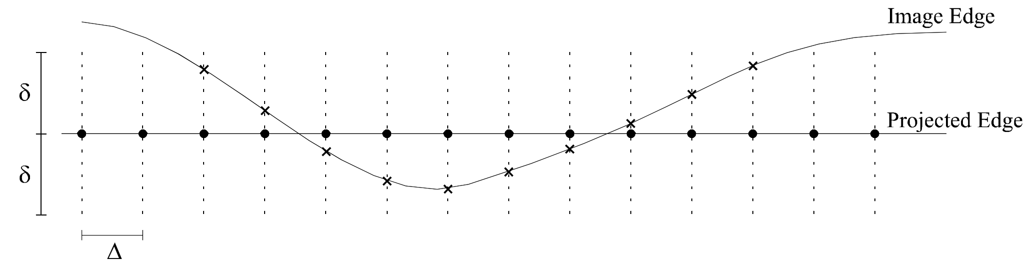

Fig. 4: Illustration of the search for edges

In [3], Deutscher proposes to use a gradient based edge detector, threshold the result to eliminate spurious edges followed by the application of a Gaussian smoothing mask to produce a pixel map, in which the value of each pixel is related to its proximity to an edge.

The likelihood is calculated on the base of the Sum of Squared Differences (SSD). For convenience of notation, it is assumed that all edges are contained in one contiguous spline with M discretizations observed. d_m denotes the distance at which an edge feature was found for the mth point and μ denotes a constant maximum error which is applied in case no edge feature could be found. The likelihood function can then be formulated as:

$$\mathrm{p}(\mathbf{z} \mid \mathbf{s}) \propto \exp\left\{-\frac{1}{2\sigma^2 M}\sum_{m=1}^{M} f(d_m, \mu)\right\} \tag{13}$$

where $f(\nu, \mu) = \min(\nu^2, \mu^2)$. [8]

3.3. *Starting condition*

The starting condition for the particle filter is usually set manually. Automatically finding an initial state can be regarded as a separate research theme. One possibility would be to use skin color maps for finding the hands and the head as done in [11], which are sufficient for the determination of an initial configuration for an upright standing person. Using a user friendly Graphical User Interface (GUI) with scale controls one can find an initial state which is good enough easily, as done in our system. The particle set is initialized by assigning this initial configuration to each particle with the probability $1/N$.

4. Special Features

In this section, we want to show details of our implementation, in particular the likelihood function with an integrated on-the-fly edge detector. Other features include a fast occlusion test and a two state model for handling elbow singularity.

4.1. *On-the-fly edge detection*

As described in section 3.2 the general approach for a likelihood function is to operate on an edge image generated in a preprocess step from the input image. This approach has several drawbacks:

- The use of common gradient based edge detection masks on the *whole* image such as the Sobel or the Laplace operator either consider edges in only one *fixed* direction or allow any direction, whereas in the outlined human motion capture system *each* projected edge has exactly *one specific* search direction of interest.
- Higher level edge detection algorithms such as the Canny edge detector in general produce lines of one pixel width, which makes the test for a specific pixel being part of an edge complicated respectively inefficient.
- In general, preprocessing the image with standard techniques neglects any information provided by the human model.

The generation of pixel maps as described in section 3.2 [3] solves the second problem mentioned, since the test for one pixel being part of an edge can be realized with one operation. However no information provided by the human model is used. Our approach is to incorporate the gradient calculation into the walk through the projected edges of the human model in the likelihood function. By doing this the information provided by the human model can be used extensively, since the gradient can always be calculated perpendicular to each projected edge.

4.1.1. *Algorithm*

The combined algorithm for the walk through a projected edge and the edge detection is presented as Pseudo code.

```
GoThroughLine(x1, y1, x2, y2)
begin
    ssd := 0;
    l := sqrt((x2 - x1)^2 + (y1 - y2)^2);
    (nx, ny) := ((y1 - y2) / l, (x2 - x1) / l);
    M := ⌊l / Δ⌋ + 1;
    (dx, dy) := ((x2 - x1) / M, (y2 - y1) / M);
    (ox, oy) := (x1, y1);

    for m = 0 TO M do
          (p1x, p1y) := (ox - δ·nx, oy - δ·ny)
          (p2x, p2y) := (ox + δ·nx, oy + δ·ny);

          if IsNotInPicture(p1x, p1y, p2x, p2y) then
                ssd := ssd + C1;
          else if IsOccluded(ox, oy) then
                ssd := ssd + C2;
          else
                (px, py) := (p1x, p1y);
                N := 2·δ + 1;
                $j_0$ := -1;
                $g_0$ := G_MIN;

                for n = 1 TO N do
                      i1 := image[round(py) · WIDTH + round(px)];
                      i2 := image[round(py + ny) · WIDTH + round(px + nx)];
                      g := |i1 - i2|;

                      if (g > $g_0$) then
                            $j_0$ := j;
                            $g_0$ := g;
                      endif

                      (px, py) := (px, py) + (nx, ny);
                endfor

                if ($j_0$ = -1) then
                      ssd := ssd + C3;
                else
                      ssd := ssd + $|j_0 - δ|^2$;
                endif
          endif

          (ox, oy) := (ox, oy) + (dx, dy);
    endfor

    return ssd / $M^2$;
end
```

All the constants for return values have to be assigned very carefully; in fact even non constant return values, e.g. depending on the last iteration of the particle filter, should to be considered. In the current implementation C1 = 0, C2 = δ^2 and C3 = $(\delta + 5)^2$. According to section 3.2, the constant δ defines the maximum search distance in each direction and is currently set to 15, Δ defines the constant distance between observed points of a projected edge and is currently set to 5. The threshold for the minimum gradient is set to G_MIN = 12. The constant WIDTH defines the image width.

It has to be pointed out that usually the value ssd is returned, whereas in our implementation ssd/M^2 is returned. This modification assigns a higher likelihood to *long* visible edges to avoid very short visible edges since in such configuration the probability cannot be reliable. For this implementation the value σ^2 in equation (13) is set to 0.0025. Although this approach achieves very good results, a more sophisticated approach for this problem will be implemented in the future.

4.1.2. *Flexibility*

Our approach is effective – since it searches for edge features perpendicular to the edge – and flexible at the same time, allowing with only small variations the implementation of various edge detection strategies. In the algorithm shown in section 4.1.1 the biggest gradient within the search region is picked, if it is bigger than a pre-defined threshold. Another strategy is to pick the nearest gradient bigger than a threshold within the search region. Also combinations of both strategies should be considered, which can all be applied to the full information contained in the original picture.

4.1.3. *Efficiency considerations*

Compared to the application of a 3×3 edge detection mask on the whole image this technique comes at no or only few extra cost. Assuming an image resolution of 640×480, at least $640 \cdot 480 \cdot 9 \approx 2.8 \cdot 10^6$ basic operations are needed for calculation of the edge image. With a total length of approximately 1500 pixels for all projected edges, Δ=5 and δ=15, $30 \cdot (1500/5)N \approx 9 \cdot 10^3 N$ additional basic operations are needed, where N is the number of particles. Thus, for N = 307 computational cost can be considered as approximately equal, for fewer particles on-the-fly detection is even faster.

In fact these considerations are of theoretic nature, in practice one also has to consider the additional cost caused by the two nested loops over the whole image, which are needed for the convolution with the edge detection mask. In contrast, in our implementation the gradient calculation is incorporated into the walk through of the projected edges, which has to be performed in any case, and therefore no extra cost for loops is caused. Measurements have shown that in fact on-the-fly edge detection is only slower by a factor of 1.4 for a large number of particles. Thus one can conclude that our on-the-fly edge detection approach does not differ significantly in terms of efficiency.

4.2. *Dynamic model*

Currently a very simple but yet effective implementation of the dynamic model is applied. In the prediction step for each new configuration *i*, being calculated on the base of a particle *j* from the last generation, the weighted difference between the two previous mean configurations is added to the components, in addition to the Gaussian noise:

$$\mathbf{s}_{k,i} = \mathbf{s}_{k-1,j} + A(\overline{\mathbf{s}}_{k-1} - \overline{\mathbf{s}}_{k-2}) + B\boldsymbol{\omega} \tag{14}$$

where $\mathbf{s}_{k,n}$ denotes the configuration of the *n*th particle after the *k*th iteration as introduced in section 2. The components of the diagonal square matrix *A* are currently all set to the same constant value:

$$\alpha_{ij} = \begin{cases} 0.7 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} . \tag{15}$$

The components of the diagonal square matrix *B* are currently set to values between 0.01 and 0.06 for rotational and to 3.0 for translational components. Each component of the vector $\boldsymbol{\omega}$ is calculated with the Gaussian random number generator algorithm from [12].

4.3. *Fast occlusion test*

When using the term occlusion, one has to distinguish between *occluded* and *occluding* points. While occluded points cannot be seen in the image under any circumstance, occluding points only cause problems when the occluded and occluding body part are of the same color. Since this is often the case, e.g. the arms of a person wearing a solid colored long sleeved shirt occluding the body, we currently treat both cases in the same way. Thus the occlusion test can be based on a point in polygon test applied on the projection of the edges of the human model into 2D space.

The basic implementation sums the angles between consecutive edge vectors and compares the result to 360°, which needs to evaluate the square root and the inverse cosine function from the Floating Point Unit (FPU), and is therefore slow. Our implementation is based on a fast point in triangle test for the 3D case, which uses FPU multiplications only, as described in [13]. The algorithm presented in [13] will be shown as Pseudo code, where `p` is the point to be tested and `a`, `b` and `c` define the three corners of the triangle.

```
SameSide(p1, p2, a, b)
begin
    cp1 := (b - a) × (p1 - a);
    cp2 := (b - a) × (p2 - a);
    return cp1 · cp2 >= 0;
end

PointInTriangle(p, a, b, c)
begin
    return  SameSide(p, a, b, c) AND
            SameSide(p, b, a, c) AND
            SameSide(p, c, a, b);
end
```

In the 2D case (z = 0) the function `SameSide` can be optimized even further:

```
SameSide2D(p1, p2, a, b)
begin
    ba0 := b.x - a.x;
    ba1 := b.y - a.y;

    cp1 := ba0 · (p1.y - a.y) - ba1 · (p1.x - a.x);
    cp2 := ba0 · (p2.y - a.y) - ba1 · (p2.x - a.x);

    cp1 · cp2 >= 0;
end

PointInTriangle2D(p, a, b, c)
begin
    return  SameSide2D(p, a, b, c) AND
            SameSide2D(p, b, a, c) AND
            SameSide2D(p, c, a, b);
end
```

The final point in quadrangle test `IsOccluded` which is applied in our occlusion test for each observed point of each projected edge is then:

```
IsOccluded(p, a1, a2, b1, b2)
begin
    return PointInPoly2D(p, a1, a2, b1) OR PointInPoly2D(p, b1, b2, a1);
end
```

Depending on which body part the currently processed projected edge belongs to, this test has to be performed only on a specific subset of all body parts, e.g. the head does not have to be tested for occlusion with the lower legs.

4.4. *Tracking through the elbow singularity*

The elbow singularity is a kinematic singularity of the human model which occurs when the angle of the elbow joint θ_e is near zero [14]. In this case it is practically impossible to determine the correct rotational component θ_s of the shoulder joint around the axis of the upper arm. Problems arise when the elbow singularity is left and a wrong value for θ_s has been assumed, then the distance between the last θ_s and the value in the new correct configuration is potentially too far.

One solution is to apply the idea of automatic model-switching as described in [15], where two dynamic models are used to track a bouncing ball. We define for each elbow joint two states depending on the angle: is θ_e smaller than a threshold φ then state one is applied otherwise state two. The only difference between these two states is the amount of noise added to θ_s in the prediction step. A higher amount of noise in the state of the elbow singularity permits to find the correct configuration even at a far distance for θ_s when the singularity is left. With *s* denoting the component of the configuration vector specifying θ_s respectively the corresponding index, and *e* denoting the component specifying θ_e, this two-state model can be rewritten, according to equation (14), as follows:

$$s_{k,i} = \begin{cases} s_{k-1,j} + \alpha_{ss} \cdot (\bar{s}_{k-1} - \bar{s}_{k-2}) + \hat{\beta}\omega_s & \text{if } e_{k-1,j} < \varphi \\ s_{k-1,j} + \alpha_{ss} \cdot (\bar{s}_{k-1} - \bar{s}_{k-2}) + \beta_s \omega_s & \text{otherwise} \end{cases} . \tag{16}$$

Currently $\hat{\beta} = 0.4$ (radians) and $\varphi = 0.2$ (radians).

5. Software Architecture

We have chosen a modular software architecture implemented with Object Oriented Programming (OOP). Using fixed interfaces, which have been designed carefully, allows the change of specific parts of the implementation by simply replacing the corresponding module, e.g. replacing the video capture module for the use of a different driver. For ongoing research in the area of search space decomposition and in particular *dynamic* search space decomposition we have implemented a *Framework* architecture for particle filters using *Template Methods* according to the design patterns described in [16], including an extension for human motion capture. This architecture is illustrated as a class diagram in figure 5.

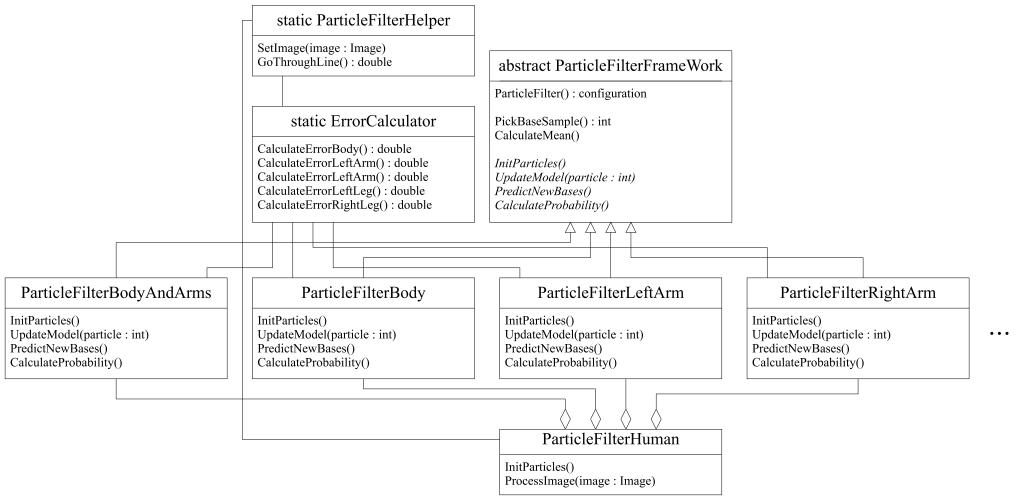
static ParticleFilterHelper
SetImage(image : Image)
GoThroughLine() : double
abstract ParticleFilterFrameWork
ParticleFilter() : configuration
PickBaseSample() : int
CalculateMean()
InitParticles()
UpdateModel(particle : int)
PredictNewBases()
CalculateProbability()
static ErrorCalculator
CalculateErrorBody() : double
CalculateErrorLeftArm() : double
CalculateErrorLeftArm() : double
CalculateErrorLeftLeg() : double
CalculateErrorRightLeg() : double
ParticleFilterBodyAndArms
ParticleFilterBody
ParticleFilterLeftArm
ParticleFilterRightArm
InitParticles()
UpdateModel(particle : int)
PredictNewBases()
CalculateProbability()
...
ParticleFilterHuman
InitParticles()
ProcessImage(image : Image)

Fig. 5: UML class diagram of the framework architecture implemented for particle filters.

6. System Setup

6.1. *Hardware setup*

The software has been developed and tested on an Intel Pentium 4, 2.8 GHz CPU with 1 GB RAM. The camera used was a Pulnix TMC 6700-CL (Progressive Scan).

6.2. *Software Environment*

The software has been implemented in C++. For image capturing the Open Computer Vision Library from Intel (OpenCV) has been used, which provides a very clean and comfortable interface to images and capturing devices – including AVI-files. The OpenCV library is also used for camera calibration. The 3D visualization has been implemented with the OpenGL library. The GUI has been implemented with Tcl/Tk.

7. Results

In our experiments, we activated all DOF except the legs and the joint for moving the upper body, resulting in 17 DOF. The video was captured at 60 Hz and a resolution of 640×480 with 24 bits per pixel RGB. All computations were applied to the corresponding grayscale image. To cope with the varying width of the torso over time, depending on the position of the arms due to loose clothing, we modeled the width of the torso for each side as a linear function (at a fixed interval) of the elbow's position of the same side. The number of particles used was 250, which resulted in a processing rate of 3.0 Hz. Screenshots from this experiment are shown in figures 6-9.

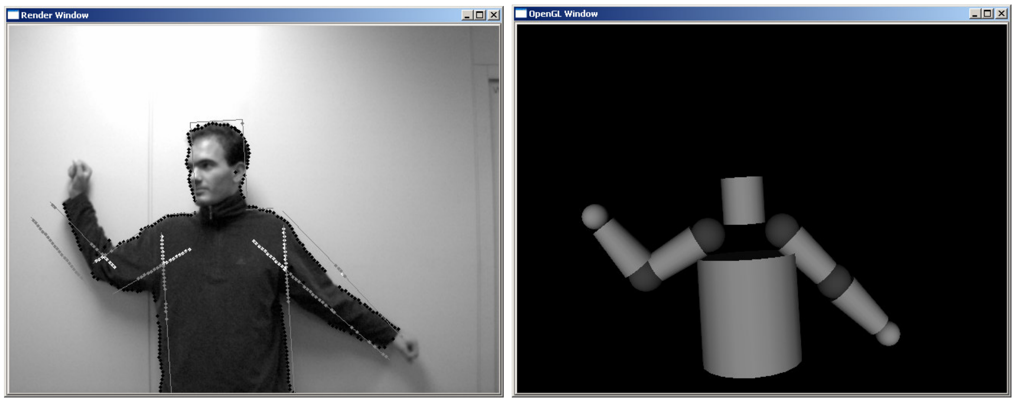

Fig. 6/7: 2D and 3D visualization at frame 0, the initial configuration has been roughly chosen. White dots identify occluded points, black dots identify found edge points and gray dots identify observed points from the projected edge for which no edge feature could be detected.

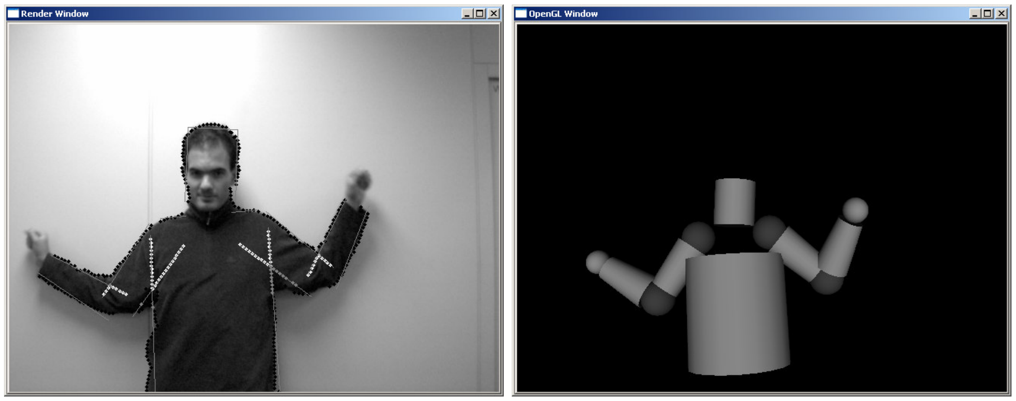

Fig. 8/9: 2D and 3D visualization at frame 30, the human motion tracker has found the correct configuration and is performing very well. White dots identify occluded points, black dots identify found edge points and gray dots identify observed points from the projected edge for which no edge feature could be detected.

In figure 10 the average SSD error per observed point over all projected edges is illustrated for the video sequence which was subject to figures 6-9. The correct configuration was found after 20 frames and was not lost at any time. Starting at frame 20, a standard deviation of ≈6.3 and an average error of ≈83 are observed, mainly caused by loosely fit clothing.

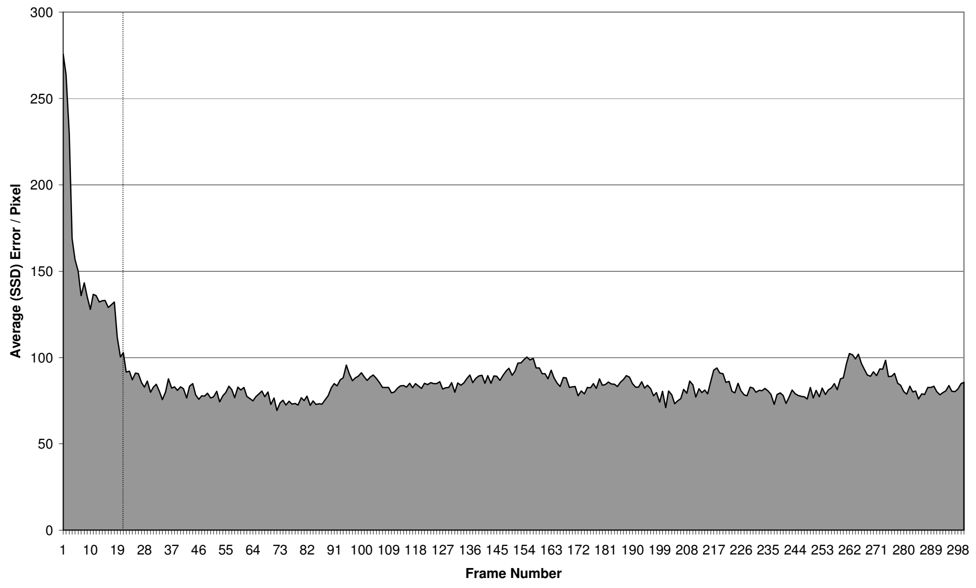

Fig. 10: Average SSD error for a video sequence of 300 frames captured at 60 Hz with 250 particles used.

8. Conclusion and Future Work

The motivation of our work is to create a full body human motion capture system for application on an active head. The input information for such a system is restricted on the images of a stereo camera system only, with the two cameras being positioned at a comparatively small distance. To our best knowledge, research activities for an optimized system – in terms of effectiveness *and* efficiency – for this specific application are new. Because a realistic human model has at least 25 DOF and thus the size of the configuration space is immense, all available information has to be included into the calculation process for a successful tracking.

Our basic system, among other features including an on-the-fly edge detector approach and a fast occlusion test, has been finished and successfully tested. A modular software architecture and a framework architecture for particle filters, including an extension for human motion capture, which is currently used for research on dynamic search space decomposition, has been presented. With the use of a single camera and as few as 250 particles, simple planar movements, body, arm and head movement resulting in a 17-dimensional search space could be tracked reliably. However using only one camera, significant depth information cannot be acquired reliably and accurately enough, thus not allowing the tracking of full 3D motion.

Also the incorporation of 3D information into the likelihood function can potentially decrease the effective search space and thus diminish the number of particles needed. Therefore we have started research on the incorporation of stereo vision. Our strategy is to combine all additional information provided by a stereo camera system i.e. using state-of-the-art correlation and disparity maps, extending the likelihood function for application on both images and incorporating the human model for solving the correspondence problem, resulting in an algorithm without search. However the great challenge and purpose of our work will be to make the most of a system limited to the use of a stereo camera system only.

Our current work in progress is also in the area of runtime optimizations to tackle the high dimensional search space. Currently a highly optimized likelihood function is being developed, which can achieve a two to three fold speedup. We also want to use the benefits from the theory on partitioned sampling [1] and annealed particle filtering [3], serving as a general approach for dealing with high dimensional search spaces. Our final goal is to attain a real-time human motion capture system for application on an active head of a humanoid robot.

References

1. J. MacCormick and M. Isard, Partitioned sampling, articulated objects, and interface-quality hand tracking, in *European Conf. Computer Vision (ECCV)* (Dublin, Ireland, 2000), pp. 3--19.

2. J. MacCormick, *Probabilistic models and stochastic algorithms for visual tracking*, PhD thesis, University of Oxford, UK, 2000.

3. J. Deutscher, A. Blake and I. Reid – Articulated Body Motion Capture by Annealed Particle Filtering, in *Conf. Computer Vision and Pattern Recognition (CVPR)* (Hilton Head, USA, 2000), pp 2126--2133.

4. D. Gavrila and L. Davis, 3-D model-based tracking of humans in action: a multi-view approach, in *Conf. Computer Vision and Pattern Recognition (CVPR)* (San Francisco, USA, 1996), pp. 73--80.

5. H. Sidenbladh, *Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences*, PhD Thesis, Royal Institute of Technolog, Stockholm, Sweden, 2001.

6. S. Wachter and H.-H. Nagel – Tracking Persons in Monocular Image Sequences, *Computer Vision and Image Understanding*, **74**(3), pp. 174--192 (1999).

7. C. Bregler and J. Malik, Tracking people with twists and exponential maps, in *Conf. Computer Vision and Pattern Recognition (CVPR)* (Santa Barbara, USA, 1998), pp. 8--15.

8. M. Isard and A. Blake, Condensation – conditional density propagation for visual tracking, *International Journal of Computer Vision,* **29**(1), pp. 5--28 (1998).

9. J. Foley, A. van Dam, S. Feiner and J. Hughes – *Computer Graphics: Principles and Practice*, 2nd edn. (Addison Wesley, 1990).

10. M. Isard and A. Blake, Contour tracking by stochastic propagation of conditional density, in *European Conf. Computer Vision (ECCV)* (Cambridge, UK, 1996), pp. 343--356.

11. K. Nickel and R. Stiefelhagen, Real-Time Person Tracking and Pointing Gesture Recognition for Human-Robot Interaction, *Int. Workshop on Human-Computer Interaction (HCI)* in conjunction with *European Conf. Computer Vision (ECCV)* (Prague, Czech Republic, 2004), pp. 28--38.

12. W. Press, S. Teukolsky, W. Vetterling and B. Flannery – *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edn. (Cambridge University Press, 1992), pp. 288--290.

13. http://www.blackpawn.com/texts/pointinpoly/default.html, Point in triangle test.

14. J. Deutscher, B. North, B. Bascle and A. Blake, Tracking through singularities and discontinuities by random sampling, in *Int. Conf. Computer Vision (ICCV)* (Kerkyra, Greece, 1999), pp. 1144--1149.

15. M. Isard and A. Blake, A mixed-state Condensation tracker with automatic model-switching, in *Int. Conf. Computer Vision (ICCV)* (Bombay, India, 1998), pp. 107--112.

16. E. Gamma, R. Helm, R. Johnson and J. Vlissides – *Design Patterns – Elements of Reusable Object-Oriented Software* (Addison Wesley, 1995).