CHALMERS

Limb movement tracking and analysis of neonates using multi-camera videos: towards automatic analysis of neurological dysfunctions

Master's Thesis in Biomedical Engineering

GRZEGORZ SOWULEWSKI

Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2015
Master's Thesis EX053/2015

Master's Thesis EX053/2015

Limb movement tracking and analysis of neonates using multi-camera videos: towards automatic analysis of neurological dysfunctions

Grzegorz Sowulewski

Supervisor and Examiner: prof. Irene Yu-Hua Gu

Department of Signals and Systems
Chalmers University of Technology

Gothenburg, Sweden 2015

Abstract

Neurological dysfunctions of newborn babies are a common problem. When diagnosed early, this issue can be mitigated through medical treatment. The main focus of this project is on video analysis of limb movement of neonates. Videos were measured at Östra hospital in Göteborg using 3 web-cameras. In the thesis work, detection of markers is performed by Scale-Invariant Feature Transform (SIFT) with post-processing for enhancement. Epipolar geometry is employed to back-project 2D markers in different views to 3D space, where camera calibration is done by using an existing software. Particle filters are used for tracking trajectories of detected markers. Finally, some preliminary analysis is performed for quantifying the movement through analyzing the limb motion trajectories. Performance of tracking is evaluated by projecting the points from 3D tracked trajectories in one of the 2D image views. Manually selected points are then used as the ground truth. Visual observation and error analysis are also shown.

Acknowledgements

I would like to express my gratitude to prof. Irene Yu-Hua Gu, my supervisor. Her patience and vast knowledge on signal processing was very appreciated. She was often able to point the right direction when I encountered some problems. Additionally, I want to thank Yixiao Yun, a PhD student at Signals and Systems department who provided me with useful advice. I would like to also thank Anders Flisberg and Magnus Thordstein for providing the subjects and videos measured with single camera. Moreover, I would like to thank Mr. Xu Long from Shanghai Jiao Tong University for providing Matlab codes for enhanced marker detection. Last but not least, thanks go to my parents who supported me in many ways during the time of studies here.

Grzegorz Sowulewski, Göteborg, August 2015

Contents

1

Introduction

Neurological dysfunctions are fairly common in newborns. Infants born very early (less than 32 weeks after conception) are in the group of highest risk of having these problems [1]. Cerebral palsy is the most common disease of nervous system and leads to motoric problems in young children. It occurs typically in 2 to 100 in 1000 infants. Usually, these problems are hard to see until the child is 1 or 2 years old. However, early detection of cerebral palsy (and other neurological problems) is needed as soon as possible. This would allow not only clinicians to give infants necessary treatment (and thus curing the problem or improve quality of life of the patient) but also to prevent treatment in the case of a false positive [2].

Experienced clinicians working with newborn babies are able to observe occurrence of abnormal movements through a careful and quite lengthy observation. Unfortunately, there are usually many babies in a ward and such careful examination is not possible for every newborn. Such an exam is still only and indicator of a potential neurological problem. To have the whole picture, more advanced techniques need to be employed such as Magnetic Resonance Imaging (MRI).

Evaluation of newborn's motoric behavior is often subjective. There is a need to systematize this method of measuring the quality of spontaneous movements to assess the state of the nervous system. It was found that general movements (GM) assessment can be a great marker for brain impairment [3]. The most significant disadvantage of this method is that it is highly qualitative. Thus, it is hard to measure the changes in the movement and systematize them [1]. General movements are spontaneous and vary for different stages of baby's development (Figure 1.1). Infants' motion was often observed in terms of angular motion in flexion and extension. However, it was oversimplified as rotations along the axes of the limbs and small changes in the direction of movements indicate complexity. Healthy infants usually move their limbs fluently and the complex-

ity of these movements is high. Babies with neurological problem have rigid movements that also lack in complexity [3].

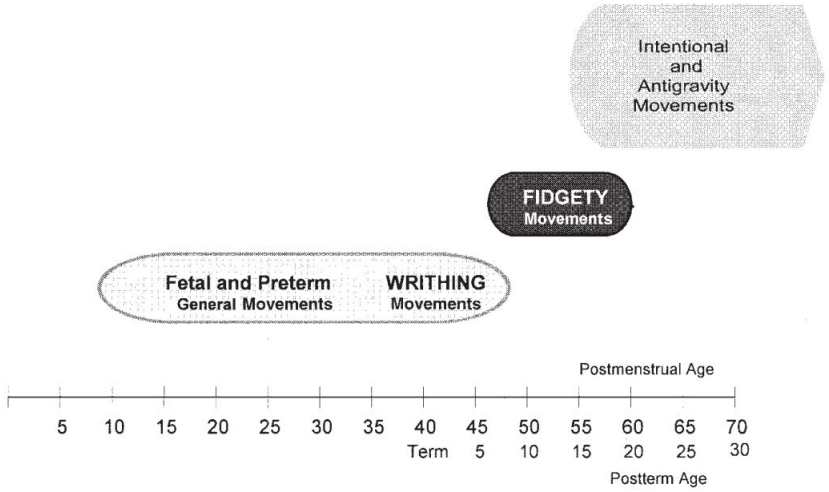


Figure 1.1: Development of general movements [3]

Image processing in medical applications has a long history. It is an important means of both qualitative and quantitative evaluation of different features. Images can be obtained in different ways. For example, X-ray pictures represent differences in attenuation of gamma radiation in different tissues of the body. Other techniques, like MRI, indirectly show concentration of hydrogen atoms which represents structure of the tissue.

Regular video camera can be a useful diagnostic tool too. It is basically image processing with time being additional dimension. Videos are usually processed frame by frame. Features are detected in each frame and their change in time can be observed. Video camera measurements do not require complicated and expensive equipment in contrary to, for example, X-ray and MRI. Most of the work here is done by the computer using various image processing and analysis methods. Some of the motion tracking methods revolutionised other industries like Computer-generated imagery (CGI) in modern movies.

1.1 Problems addressed

There are some problems which arise when creating a motion tracking system for newborn babies. They mostly concern technical issues and computer algorithms but one should not neglect organisational problems and nature of the measurement.

1. **Data acquisition**: Recording a video of a newborn baby requires taking into account some legal concerns and asking clinicians for help. What is more, the

optimal setup for the measurement along with measurement protocol has to be specified.

2. **Camera calibration**: Cameras should be calibrated to fully utilize their potential.
3. **Object detection**: Objects need to be detected successfully and without spurious points.
4. **Getting 3D position of the objects**: Coordinates in each view should be used to put the objects in 3-dimensional space.
5. **Obtaining the trajectories**: Movement of the registered objects has to be tracked for later evaluation.
6. **Pattern recognition**: Movement patterns should reflect the state of the patient. They have to be categorized as healthy or not.

1.2 Aim of this thesis

This thesis project aims to introduce a system to automatically classify newborns which would be a helpful tool for clinicians. The project developed in cooperation with 2 clinicians from neonatal wards in different hospitals in Gothenburg. They provided the patients and their expertise. They also asked parents for their consent as it was legally necessary to have it before performing measurements. Legal issues are also the reason why no images from actual measurements are included in this thesis.

Three calibrated cameras were used during measurements. They were placed on tripods in different angles around the patient's bed but still leaving some place for parents and clinician. Markers were dots in different colors on white paper bands. The band were placed on patient's limbs, 2 on each. Cameras were calibrated separately for every video as they had to be moved often. Detection and placing point in 3D was done using enhanced version of known algorithms and geometrical correspondences between calibrated cameras. Markers' trajectories were obtained with a use of tracking algorithms.

Hopefully, this research will be used in the future to develop a successful camera system and introduce it in hospitals. It may significantly improve diagnosis of neurological problems and thus improve the quality of newborns' lives.

2

Background

This chapter sheds light on theories which were taken into consideration when proceeding with the project. They are well-established methods with strong mathematical basis. Studying them was necessary to decide how the problems should be solved. The presented algorithms were later adapted to the needs of the project. This chapter is divided into 3 parts: Scale Invariant Feature Transform (SIFT), Particle Filter and 3D reconstruction. These sections address the following problems:

- marker detection,
- object tracking,
- projecting 2D object into 3D world.

To detect markers, it was necessary to study Scale Invariant Feature Transform. Then, when markers are successfully detected, they have to be tracked frame by frame. This is where idea of Particle Filter is employed. Additionally, it is crucial to calibrate views from multiple cameras so tracking can occur in 3D space.

2.1 Scale Invariant Feature Transform

Scale Invariant Feature Transform (SIFT) is a method used for image matching and object recognition. In this project, it proved to be very effective in detecting markers on infant's body. SIFT is a common algorithm in computer vision, especially in matching points between different 2D views of a 3D world. As the name might suggest, SIFT descriptor is invariant to rotations, translations and scaling. It is also quite resistant to some changing illumination and some perspective transformations [4].

Originally, the SIFT descriptor was used just for detecting points of interest in a gray-scale image. This is also the reason why it is used in this project. Local gradient direction

of pixel intensities would give information on the local structures in a neighbourhood around each point of interest. The descriptor could be later used to match corresponding points in other images. Briefly, SIFT transforms image data into coordinates relative to local features.

Algorithm was developed by David Lowe and can be divided into 4 parts: scale-space extrema detection, keypoint localization, orientation assignment and keypoint descriptors [5].

2.1.1 Scale-space extrema detection

Firstly, one must identify locations and scales that might be assigned to the object in different views. It is necessary for keypoint detection. Identifying these locations is done by searching for stable features in all possible scales, using a function called scale space. All scales and locations are searched with variable-scale Gaussian. It is done through convolution:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y), \tag{2.1}$$

where $L(x,y,\sigma)$ is a scale space function of the image, $I(x,y)$ is the image and $G(x,y,\sigma)$ is a variable-scale (σ) Gaussian defined:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}}. \tag{2.2}$$

To increase efficiency, difference-of-Gaussian (D) function is convolved with the image which is the difference of 2 nearby scales separated by a factor k:

$$D(x,y,\sigma) = (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma). \tag{2.3}$$

The same process is repeated for each scale (octave). Firstly, the original image is convolved with Gaussian function. Then, the adjacent scale space images resulting from previous operation are subtracted to get Difference-of-Gaussian. After, doing this for first octave, Gaussian image is downsampled by 2. The process is illustrated in Figure (2.1). Difference-of-Gaussian gives a good approximation to the scale-normalized Laplacian of Gaussian, $\sigma^2\nabla^2 G$, which is needed for scale invariance:

$$G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G. \tag{2.4}$$

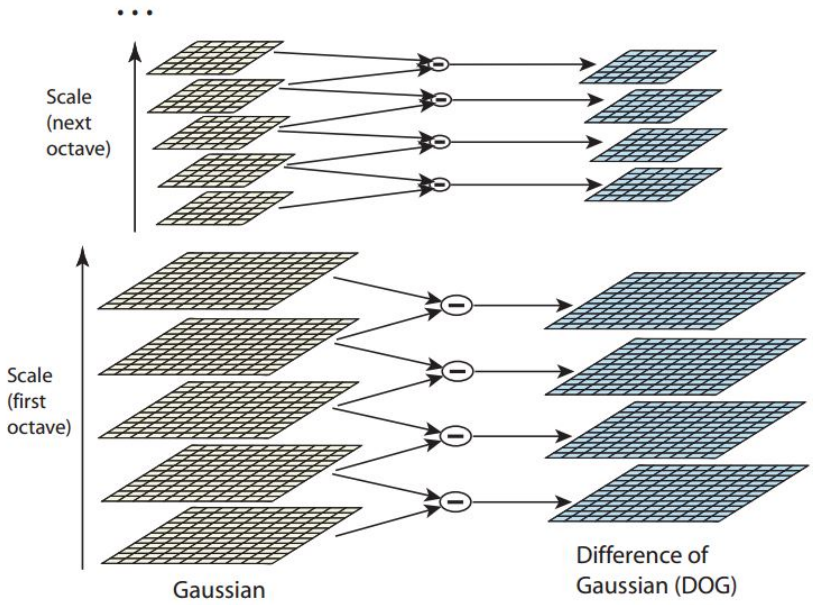


Figure 2.1: Computing Difference of Gaussian (DoG) [5].

Factor k is constant throughout all scales. Calculating D is done for every octave in scale space which is doubling of σ. Octave is then divided into integer number of intervals (s) and $k = 2^{1/s}$. Thus, multiple images in each octave in Figure (2.1). Extrema are compared to their 26 neighbours: 8 in the current image, 9 neighbours in the scale above and 9 neighbours in the scale below, to reject local extrema. The extremum is selected only if it is larger or smaller than all of the considered neighbouring pixels. Minimum/maximum detection is presented in Figure (2.2) [5].

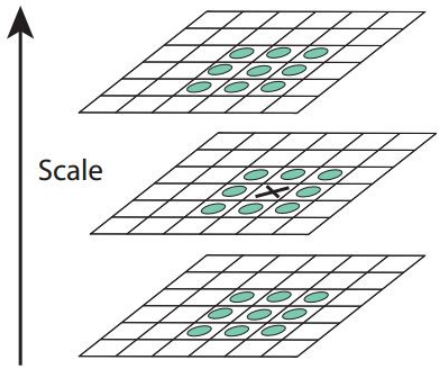


Figure 2.2: Maxima and minima detection in DoG images [5].

2.1.2 Keypoint localization

When a point of interest is found, it should be compared with nearby data. Points having low contrast need to be rejected as they are sensitive to noise. The same applies to points not localized along the edge.
Initially keypoints were located at the location and scale of the central sample point. More effective approach determines interpolated location of maximum by fitting a quadratic function to the local sample points. Taylor expansion of the scale-space function (D) is used and the origin is at the sample point:

$$D(x) = D + \frac{\partial D^T}{\partial x}x + \frac{1}{2}x^T\frac{\partial^2 D}{\partial x^2}x. \tag{2.5}$$

$x = (x,y,\sigma)^T$ is the offset from sample point. Location of extremum ($\hat{x}$) can be calculated:

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x}. \tag{2.6}$$

Value at extremum is calculated and values below a certain threshold are discarded (because of their low contrast):

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x}. \tag{2.7}$$

There is another criterion of rejection of candidate points: edge response. The DoG funtion is very sensitive to edges even if the point's location on the edge is not determined accurately. Such badly defined peak in DoG function has a large principal curvature across the edge but the one in perpendicular direction i significantly smaller. Hessian matrix $\mathbf{H}$ of size 2x2 computed at the scale and location of the candidate point is used to calculate principal curvature:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}, \tag{2.8}$$

where derivatives are differences in neighbouring pixels.
The eigenvalues of the matrix $\mathbf{H}$ are proportional to principal curvatures of D. It is not necessary to compute the eigenvalues, their ratio is enough. The sum of 2 eigenvalues (α - the bigger and β - the smaller) can be calculated from the trace of $\mathbf{H}$ while their product is the determinant:

$$Tr(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta, \tag{2.9}$$

$$Det(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta. \tag{2.10}$$

If the determinant is negative the point will still be discarded because of different signs of curvatures. Such point cannot be an extremum. When r is the ratio between the largest and the smallest eigenvalue, $\alpha = r\beta$. It leads to:

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}. \tag{2.11}$$

Therefore, the expression above depends only on the ratio of eigenvalues and α and β do not have to be calculated explicitly. The last part of equation (2.11) increases with r and is at minimum when α and β are equal. Hence, checking if the ratio of principal curvatures is below r, one can compare [5]:

$$\frac{Tr(\mathbf{H})}{Det(\mathbf{H})} < \frac{(r+1)^2}{r}. \tag{2.12}$$

2.1.3 Orientation assignment

Assigning orientation to every keypoint is necessary to achieve rotation invariance. Image sample has to be selected with the closest scale to scale of keypoint. This is necessary to maintain scale-invariance. After that, gradient magnitude (m) and orientation (θ) is calculated for each sample image (L):

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}, \tag{2.13}$$

$$\theta(x,y) = tan^{-1}(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}). \tag{2.14}$$

Orientation is quantized and samples are weighted by their gradient magnitudes [5].

2.1.4 Keypoint descriptor

In this step, descriptor for each image region is computed. It must be distinctive but invariant to changes of illumination and point of view.
Gradient magnitude and orientation from the previous step is necessary to create a keypoint descriptor for each sample point in a region around its location. These descriptors (represented with an arrow) are weighted by a Gaussian window with σ equal to half the width of the window of descriptor. Then, each sample point is assigned a weight. The window is used to smooth the descriptor so big changes in descriptor with small changes in the position are avoided. Additionally, it assigns smaller weights to gradients far from the center of the descriptor which are most affected by errors. Area under this window is divided into subregions. Each subregion is assigned an 8-arrow descriptor which represents the sum of gradient magnitudes in corresponding directions. It is illustrated in Figure 2.3.

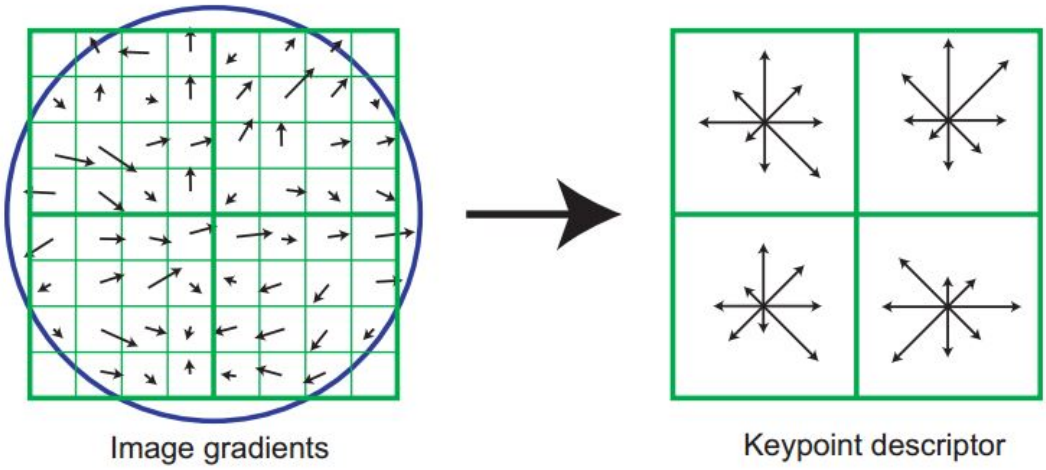


Figure 2.3: Computing the keypoint descriptor [5].

Boundary effects and sharp changes must be avoided. Due to this, the value of each gradient is interpolated and distributed into adjacent bins of histogram. The vector containing the values off all components of orientation histogram related to lengths of arrows on the right side of Figure (2.3) form the descriptor. After that, the feature vector is changed to lessen the influence of illumination changes.

2.2 Detection enhancement

Scale Invariant Feature Transform is a powerful tool but it produces points which are not of interest. The outliers should be excluded so (ideally) only markers remain in each image of the video.
There are 4 parameters to take into account: size of the object, connectedness, difference in intensity between pixels and distance between detected objects. The used algorithm [6] uses 2 windows of arbitrarily defined sizes: inner and outer. The windows are imposed on every detected point of interest. Object should entirely cover inner window and should not exceed the outer. This ensures that detected entity is of a right size. It is visualized in Figure 2.4. Only target 2 in the figure satisfies the condition.
Using these 2 windows on every point detected in SIFT also ensures that the object is not part of a bigger one. To satisfy intensity constraint one must compare mean intensity in inner window with the outer band.

$$mean(I^{out} - I^{in}) - mean(I^{in}) > T_i, \tag{2.15}$$

where T_i is the chosen intensity threshold.

Finally, points lying too close too each other are discarded. The idea is presented in Algorithm 1.

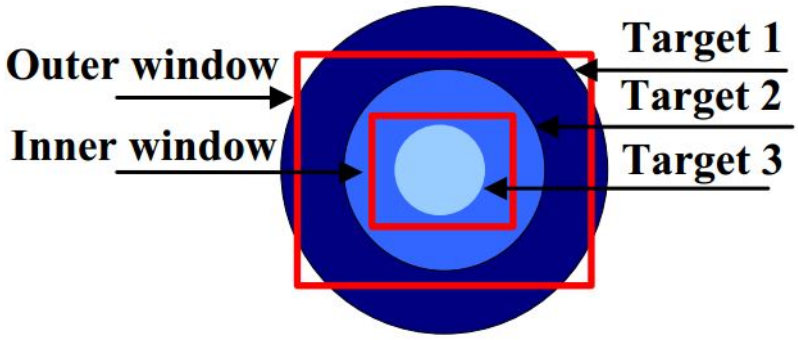


Figure 2.4: Idea behind usage of 2 windows [6].

Algorithm 1: Effective Markers Detection

Data: Image I, position of candidate points from SIFT P, radius of inner window r^{in}, radius of outer window r^{out}, binarizing threshold T_b, intensity threshold T_i.

Result: Position of candidate points without outliers P_{new}.

1 Exclude points P closer than r^{out} to the border of the image
2 **for** each point P
3 Place windows W^{in} and W^{out} around the point and binarize are within W^{out} (threshold T_b)
4 **if** Object fills entire W^{in} **and** is not adjacent to border of W^{out}
5 Connectedness condition=1
6 **end if**
7 **if** difference between mean pixel intensities of area in W^{in} and W^{out}-W^{in} is higher than T_i
8 Intensity condition=1
9 **end if**
10 **if** Connectedness condition=1 **and** Intensity condition=1
11 $P_{new} = P$
12 **end if**
13 **end for**

2.3 Particle filter

Particle filter belongs to Monte Carlo algorithms and represents probability densities with ”particles” that can be applied to state-space model. This filter is a generalization of Kalman filtering methods [7]. Kalman filter can only be applied in linear and Gaussian systems while Particle filter does not bear these constraints. The particle filter employs a hidden Markov Model, where the system consists of hidden and observable variables. Observable variables are simply observations and hidden variables are found in state-

space model of the system. Their relationship is described with a known function.

2.3.1 Hidden Markov Process

There is an X-valued discreet time Markov process $\{\mathbf{x}_n\}_{n\geq 1}$ such that:

$$\mathbf{x}_1 \sim \mu(x_1); \mathbf{x}_n|(\mathbf{x}_{n-1} = x_{n-1}) \sim f(x_n|x_{n-1}) \tag{2.16}$$

where $\mathbf{x}$ is a state vector, "$\sim$" means distributed according to, $\mu(x)$ is a probability density function and $f(x|x')$ is a probability density associated with moving from x' to x. The goal is to estimate $\{\mathbf{x}_n\}_{n\geq 1}$ with access to Y-valued process $\{\mathbf{y}_n\}_{n\geq 1}$. It is assumed that given $\{\mathbf{x}_n\}_{n\geq 1}$, the observations $\{\mathbf{y}_n\}_{n\geq 1}$ are statistically independent and their marginal densities are given by:

$$\mathbf{y}_n|(\mathbf{x}_n = x_n) \sim g(y_n|x_n). \tag{2.17}$$

The vector $\mathbf{y}$ is called a measurement vector. Models that can be described with (2.16) and (2.17) are hidden Markov models (HMM) or general state-space models. (2.16) defines the prior distribution of the process $\{\mathbf{x}_n\}_{n\geq 1}$ and (2.17) is the likelihood function:

$$p(x_{1:n}) = \mu(x1) \prod_{k=2}^{n} f(x_k|x_{k-1}) \tag{2.18}$$

and

$$p(y_{1:n}|x_{1:n}) = \prod_{k=1}^{n} g(y_k|x_k) \tag{2.19}$$

where, for any sequence $\{z_n\}_{n\geq 1}$, and any $i \leq j, z_{i:j} := (z_i, z_{I=1},...,z_j)$.
For this kind of Bayesian system, inferring $\mathbf{x}_{1:n}$ given $\mathbf{y}_{1:n} = y_{1:n}$ relies on the posterior distribution

$$p(x_{1:n}|y_{1:n}) = \frac{p(x_{1:n},y_{1:n})}{p(y1:n)}, \tag{2.20}$$

where

$$p(x_{1:n},y_{1:n}) = p(x_{1:n})p(y_{1:n}|x_{1:n}), \tag{2.21}$$

and

$$p(y_{1:n}) = \int p(x_{1:n},y_{1:n})dx_{1:n}. \tag{2.22}$$

Integral in (2.22) is a finite sum for finite state-space HMM models and all discrete probability distributions can be calculated exactly. For most non-linear and non-Gaussian models, however, it is necessary to use numerical methods to compute these distributions. Particle methods are simulation-based and use samples distributed according to

posterior distributions $p(x_{1:n}|y_{1:n})$ and allow to approximate $p(y_{1:n})$. They belong to Sequential Monte Carlo (SMC) methods [8].

2.3.2 Monte Carlo Methods

Firstly, some generic probability density $\pi_n(x_{1:n})$ is considered for some fixed n. Sampling N independent random variables $\mathbf{x}^i_{1:n} \sim \pi_n(x_{1:n}$ for $i = 1,...,N$ allows The Monte Carlo method to approximate $\pi_n(x_{1:n})$:

$$\hat{\pi}_n(x_{1:n}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x^i_{1:n}}(x_{1:n}), \tag{2.23}$$

where $\delta_{x_0}(x)$ is the Dirac delta mass located at x_0. With this approximation, it is possible to estimate, for example, $\pi_n(x_k)$, using:

$$\hat{\pi}_n(x_k) = \frac{1}{N}\sum_{i=1}^{N} \delta_{x^i_k}(x_k), \tag{2.24}$$

and the expectation of any test function $\varphi_n : \mathbf{x}^n \to R$ written as

$$I_n(\varphi_n) := \int \varphi_n(x_{1:n})\pi_n(x_{1:n})dx_{1:n}, \tag{2.25}$$

which is estimated by

$$I^{MC}_n(\varphi_n) := \int \varphi_n(x_{1:n})\hat{\pi}_n(x_{1:n})dx_{1:n} = \frac{1}{N}\sum_{i=1}^{N} \varphi_n(\mathbf{x}^i_{1:n}). \tag{2.26}$$

Variance of this estimate is calculated by

$$V[I^{MC}_n(\varphi_n)] = \frac{1}{N}(\int \varphi^2_n(x_{1:n})\pi_n(x_{1:n})dx_{1:n} - I^2_n(\varphi_n)). \tag{2.27}$$

The biggest advantage of Monte Carlo methods is that the variance of approximation decreases with increasing number of particles [8].

2.3.3 Importance Sampling

Importance Sampling is used in most Monte Carlo methods. It requires introduction of an importance density $q_n(x_{1:n})$ such that

$$\pi_n(x_{1:n}) > 0 \implies q_n(x_{1:n}) > 0. \tag{2.28}$$

It leads to:

$$\pi_n(x_{1:n}) = \frac{w_n(x_{1:n})q_n(x_{1:n})}{\mathbf{z}_n} \tag{2.29}$$

and

$$\mathbf{z}_n = \int w_n(x_{1:n}) q_n(x_{1:n}) dx_{1:n} \tag{2.30}$$

where $w_n(x_{1:n})$ is the unnormalized weight function

$$w_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{q_n(x_{1:n})}. \tag{2.31}$$

Importance Sampling gives an estimate of the normalizing constant with relative variance:

$$\frac{V_{IS}[\hat{\mathbf{z}}_n]}{\mathbf{z}_n^2} = \frac{1}{N}\left(\int \frac{\pi_n^2(x_{1:n})}{q_n(x_{1:n})} dx_{1:n} - 1\right). \tag{2.32}$$

It is also possible to use this method to compute $I_n(\varphi_n)$ [8]

$$I_n^{IS}(\varphi_n) = \int \varphi(x_{1:n}) \hat{\pi}_n(x_{1:n}) dx_{1:n} = \sum_{i=1}^{N} \mathbf{w}_n^i \varphi_n(\mathbf{x}_{1:n}^i). \tag{2.33}$$

2.3.4 Sequential Importance Resampling

The degeneracy of algorithm without resampling cannot be avoided. Therefore, it is critical to eliminate trajectories with small normalized importance weights and focus on the ones with large weights. A good measure of degeneracy is the effective sample size n_{eff} which can be estimated by [9]

$$\widehat{n_{eff}} = \frac{1}{\sum_{i=1}^{N} (\mathbf{w}_n^i)^2}. \tag{2.34}$$

When effective sample size is much lower than original sample size, some weights are very small. They can be neglected and, for example, substituted with samples of higher weights. There are 3 popular resampling methods [8]

- systematic resampling,
- residual resampling,
- multinomial resampling.

2.3.5 Filtering

The aim of particle filter is to compute a numerical approximation of the distribution $p(x_{1:n}|y_{1:n})_{n\geq 1}$ sequentially in time. Pseudocode for Sampling Importance Resampling (SIR) particle filter looks as follows [8].

At time $n = 1$

- Sample $\mathbf{x}_1^i \sim q(x_1|y_1)$.

- Compute the weights $w_1(\mathbf{x}_1^i) = \frac{\mu(\mathbf{x}_1^i)g(y_1|\mathbf{x}_1^i)}{q(\mathbf{x}_1^i|y_1)}$ and $\mathbf{w}_1^i \propto w_1(\mathbf{x}_1^i)$.
- Resample $\{\mathbf{w}_1^i, \mathbf{x}_1^i\}$ to obtain N equally-weighted particles $\{\frac{1}{N}, \bar{\mathbf{x}}_1^i\}$.

At time $n \geq 2$

- Sample $\mathbf{x}_1^i \sim q(x_n|y_n, \bar{\mathbf{x}}_{n-1}^i)$ and set $\mathbf{x}_{1:n}^i \leftarrow (\bar{\mathbf{x}}_{1:n-1}^i, \mathbf{x}_n^i)$.
- Compute the weights $\alpha_n(\mathbf{x}_{n-1:n}) = \frac{g(y_n|\mathbf{x}_n^i)f(\mathbf{x}_n^i|\mathbf{x}_{n-1}^i)}{q(\mathbf{x}_n^i|y_n, \mathbf{x}n-1^i)}$ and $\mathbf{w}_n^i \propto \alpha_n(\mathbf{x}_{n-1:n}^i)$.
- Resample $\{\mathbf{w}_n^i, \mathbf{x}_{1:n}^i\}$ to obtain N new equally-weighted particles $\{\frac{1}{N}, \bar{\mathbf{x}}_{1:n}^i\}$.

Therefore, at time n we get

$$\hat{p}(x_{1:n}|y_{1:n}) = \sum_{i=1}^{N} \mathbf{w}_n^i \delta_{\mathbf{x}_{1:n}^i}(x_{1:n}), \tag{2.35}$$

$$\hat{p}(y_n|y_{1:n-1}) = \sum_{i=1}^{N} \mathbf{w}_{n-1}^i \alpha_n(\mathbf{x}_{n-1:n}^i). \tag{2.36}$$

2.4 Epipolar geometry

Epipolar geometry is often used when dealing with projection of a scene into 2 views. It depends only on intrinsic parameters of the cameras and their relative positions. Independence of the projected scene makes it simple and easy to perform various mathematical operations between the views.
Epipolar geometry is defined on intersection of image planes with their pencil. The baseline is a line between the camera centres. The correspondence between some point X in the projected scene and 2 views is shown in Figure 2.5. Points C and C' are camera centres and x and x' are projections of X on image planes. All these points lie in a plane π.

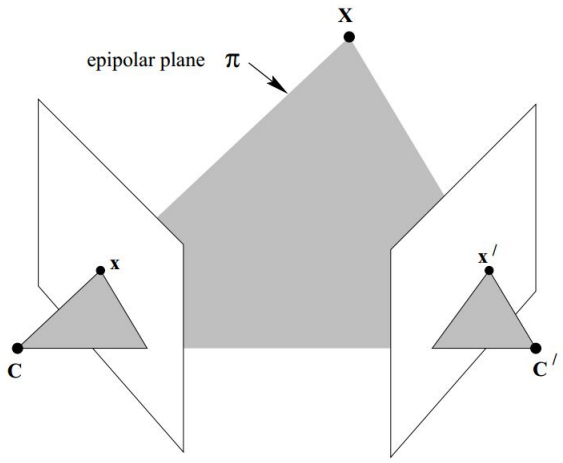


Figure 2.5: Epipolar plane [10]

Points in each image plane intersecting with a baseline are called epipoles. In Figure 2.5 there would be epipole for the left and right plane called e and $e'e$ respectively. The point X is projected on the left hand side plane with a ray defined by the camera center. This ray is represented by a line l' in the right hand side view. It is presented in Figure 2.6. Line l' is the epipolar line which corresponds to projection of the object to the other view (x).

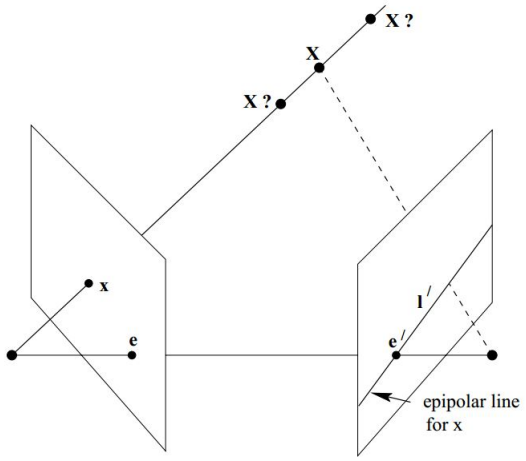


Figure 2.6: Point to line correspondence [10]

There are few important terms used when talking about epipolar geometry. They are necessary to understand when performing all sorts of operations [10].

- Epipole - a point on the image plane where it intersects with a line joining 2 cameras.

- Epipolar plane - a plane connecting camera centres. There is a family of such planes which rotate around the baseline.
- Epipolar line - line marking the intersection of image plane and epipolar plane.

2.4.1 Fundamental matrix

The epipolar geometry can be easily represented in a matrix. It is called the fundamental matrix. It is used to derive an epipolar line of a point. In the Figure 2.6 it is visible that projection of a point from one view can only correspond to a line in the other. This happens because one view gives no information about the depth. Therefore the mapping is point-to-line:

$$x \rightarrow l', \tag{2.37}$$

and this is done with a use of fundamental matrix $\mathbf{F}$

$$\mathbf{Fx} = \mathbf{l}, \tag{2.38}$$

where $\mathbf{x}$ and $\mathbf{l}$ are 1 x 3 vectors (homogeneous coordinates) and $\mathbf{F}$ is a 3 x 3 matrix [10]. Another interesting property of the fundamental matrix is that when 2 views of the same point in the scene correspond to each other, then:

$$\mathbf{x}'^T\mathbf{Fx} = 0. \tag{2.39}$$

2.5 3D reconstruction

In this project, obtaining position and tracking of markers is done in 3 dimensions. Therefore, it is necessary to use at least 2 cameras to get 2D coordinates plus depth. Calibration is performed to know relative positions of cameras and compensate for single camera distortions. In this process, camera matrices are calculated which make it easy to get different relations between 2-dimensional views and 3D scene. Such relations are derived with a use of epipolar geometry.

2.5.1 Camera parameters

Extracting camera parameters is a first step to do calibration. Camera matrix is composed of intrinsic and extrinsic camera parameters. They represent the model of how world coordinates of the object correspond to camera coordinate system. Pin-hole camera model is considered in this chapter [11].

Intrinsic parameters

Extraction of intrinsic parameters can be done independently for each camera. These parameters are:

- focal length (f_c),
- principal point (c_c),
- skew coefficient (α_c),
- distortions (k_c).

Let P be a point in space of coordinate vector $\mathbf{x}_c = [x_c, y_c, z_c]^T$ in the camera reference frame. It can be projected to image plane according to intrinsic parameters. Let $\mathbf{x}_n$ be normalized image projection:

$$\mathbf{x}_n = \begin{bmatrix} x_c/z_c \\ y_c/z_c \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.40}$$

Let $r^2 = x^2 + y^2$. After including lens distortion, the new normalized point coordinate $\mathbf{x}_d$ is defined:

$$\mathbf{x}_d = \begin{bmatrix} x_d(1) \\ x_d(2) \end{bmatrix} = (1 + k_c(1)r^2 + k_c(2)r^4 + k_c(5)r^6)\mathbf{x}_n + \mathbf{dx} \tag{2.41}$$

where $\mathbf{dx}$ is the tangential vector:

$$\mathbf{dx} = \begin{bmatrix} 2k_c(3)xy + k_c(4)(r^2 + 2x^2) \\ k_c(3)(r^2 + 2y^2) + 2k_c(4)xy \end{bmatrix}. \tag{2.42}$$

Vector $\mathbf{k}_c$ contains both radial and tangential distortion coefficients. Once distortion is applied, the final pixel coordinates $[x_p; y_p]$ of the projection P on the image plane are:

$$x_p = f_c(1)(x_d(1) + \alpha_c * x_d(2)) + c_c(1) \tag{2.43}$$

$$y_p = f_c(2)x_d(2) + c_c(2) \tag{2.44}$$

The pixel coordinate vector and the normalized (distorted) coordinate vector $\mathbf{x}_d$ are related to each other through the linear equation:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_d(1) \\ x_d(2) \\ 1 \end{bmatrix}. \tag{2.45}$$

$\mathbf{K}$ is defined as follows:

$$\begin{bmatrix} f_c(1) & \alpha_c * f_c(1) & c_c(1) \\ 0 & f_c(2) & c_c(2) \\ 0 & 0 & 1 \end{bmatrix} \tag{2.46}$$

Parameters $f_c(1)$ and $f_c(2)$ are the focal distance expressed in units of horizontal and vertical pixels. They are usually similar. The ratio $f_c(2)/f_c(1)$ is called aspect ration and is different from 1 if the pixels in sensor array are not square. The coefficient α_c holds the angle between x and y sensor axes. This model is called "affine distortion model" [12].

Extrinsic parameters

Extrinsic parameters are rotation ($\mathbf{R}$) and translation ($\mathbf{t}$) and describe placement of camera with regard to image. It is best to extract them with calibration grid 2.7.

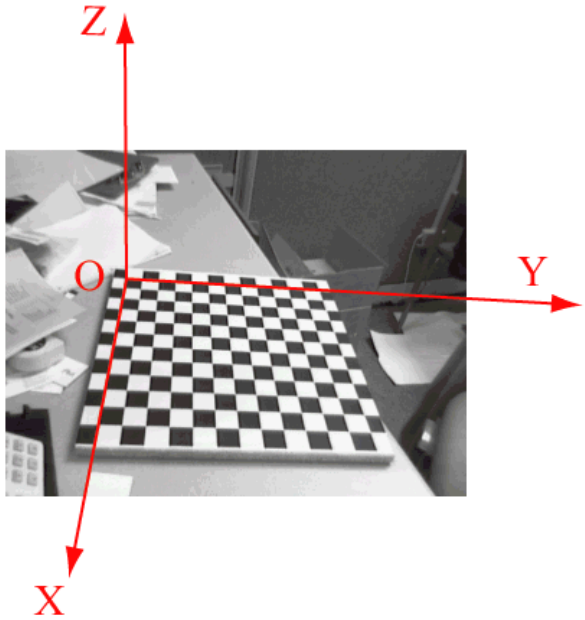


Figure 2.7: Calibration grid with reference frame [12].

Let P be a point space of coordinate vector $\mathbf{x} = [x, y, z]^T$ in the grid reference frame. Let $\mathbf{x}_c = [x_c, y_c, z_c]^T$ be the coordinate vector of P in the camera reference frame. $\mathbf{x}$ and $\mathbf{x}_c$ are related to each other through the following rigid motion equation:

$$\mathbf{x}_c = \mathbf{R} \cdot \mathbf{x} + \mathbf{t} \tag{2.47}$$

The translation vector $\mathbf{t}$ is the coordinate vector of the origin of the grid pattern (O) in the camera reference frame and the third column of the matrix $\mathbf{R}$ is the surface normal vector of the plane containing the planar grid in the camera reference frame [12].

Camera matrix

Computation of intrinsic and extrinsic parameters is a crucial process as they provide all necessary information that is possible to get about the camera and its placement with

regard to the object. To make this information useful and easily translate it for the computer, one combines it into camera matrix **P**. It is the ultimate description of the camera according to the chosen model. Camera matrix is calculated simply by:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}], \tag{2.48}$$

where **R** indicates the rotation matrix (of size 3x3) and **t** is the translation vector (of length 3). The result is, therefore, a 3x4 matrix **P** [10].

2.5.2 Two-View Triangulation

Having calibrated cameras allows to reconstruct the scene in 3 dimensions even though each camera provides information for only 2 dimensions. It is possible with the use of camera matrices which contain both intrinsic and extrinsic parameters.

Given the camera matrices P and P', let $\mathbf{x}$ and $\mathbf{x}'$ be two points in the two images that satisfy the epipolar constraint, $\mathbf{x}'^T\mathbf{F}\mathbf{x} = 0$. This constraint might be interpreted geometrically in terms of the rays in space corresponding to the two image points. It means that $\mathbf{x}'$ lies on the epipolar line $\mathbf{F}\mathbf{x}$. And this indicates that the two rays back-projected from image points $\mathbf{x}$ and $\mathbf{x}'$ lie in a common epipolar plane - a plane passing through the two camera centres. Since the two rays lie in a plane, they will intersect in some point. This point X projects via the two cameras to the points $\mathbf{x}$ and $\mathbf{x}'$ in the two images. It is shown in Figure 2.8.

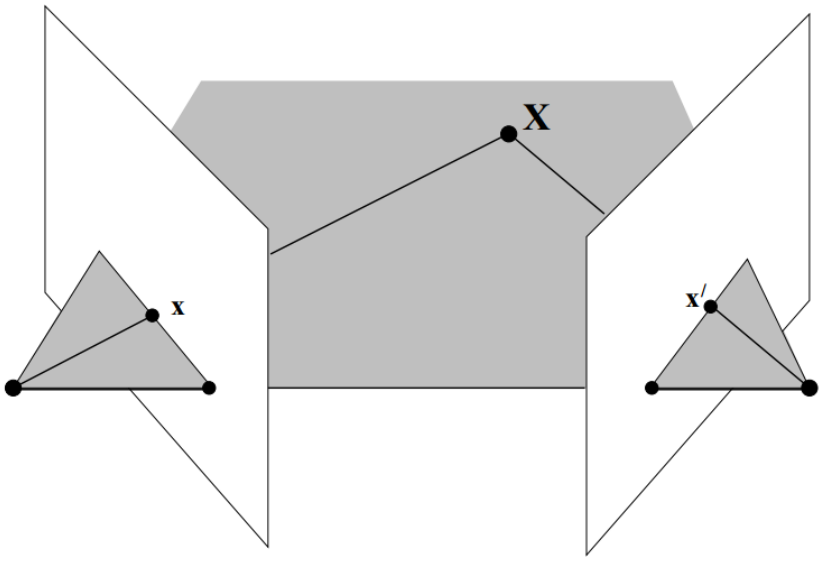


Figure 2.8: Triangulation [10].

The only points that cannot be determined from their images are points on the baseline between two cameras. In this case the back-projected rays are collinear and intersect

along their whole length. Therefore, the point X cannot be uniquely determined. Points on the baseline project to the epipoles in both images. The basis of reconstruction is projective reconstruction theorem (Theorem 1)[10].

Theorem 1. *Suppose that $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}'_i$ is a set of correspondences between points in two images and that the fundamental matrix $\boldsymbol{F}$ is uniquely determined by the condition $\boldsymbol{x}'^T_i \boldsymbol{F}\boldsymbol{x}_i = 0$ for all i. Let $(\boldsymbol{P}_1, \boldsymbol{P}'_1, \{X_{1i}\})$ and $(\boldsymbol{P}_2, \boldsymbol{P}'_2, \{X_{2i}\})$ be two reconstructions of the correspondences $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}'_i$. Then there exists a non-singular matrix $\boldsymbol{H}$ such that $\boldsymbol{P}_2 = \boldsymbol{P}_1\boldsymbol{H}^{-1}$, $\boldsymbol{P}'_2 = \boldsymbol{P}'_1\boldsymbol{H}^{-1}$ and $X_{2i} = \boldsymbol{H}X_{1i}$ for all i, except for those i such that $\boldsymbol{F}\boldsymbol{x}_i = \boldsymbol{x}'^T_i \boldsymbol{F} = 0$ [10].*

2.5.3 Three-View Triangulation

With 3 views, it convenient to calculate trifocal tensor. It is a set of 3 matrices which is independent on the scene but describes relations between the cameras. It allows to transfer points from 2 views to a third.
Figure 2.9 presents a line L which is imaged as triplet $l \leftrightarrow l' \leftrightarrow l''$ in 3 views indicated by their centres, C, C', C'', and image planes. Corresponding lines back-projected from images intersect in a single line in 3D space.

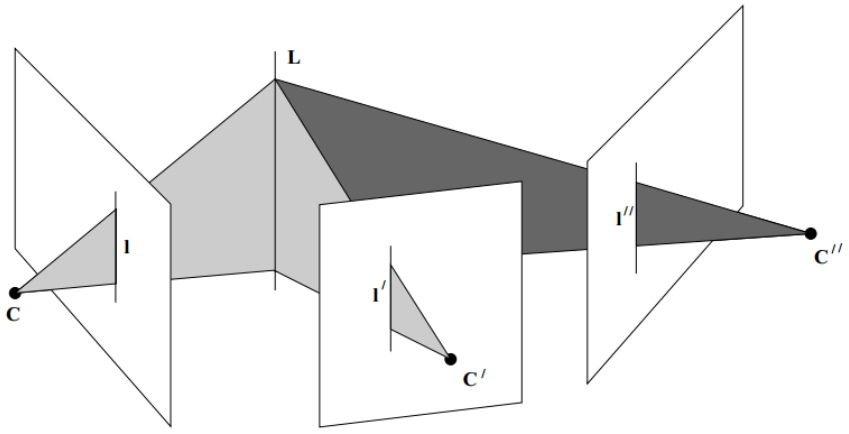


Figure 2.9: Projections of a line in 3 views [10].

Figure 2.9 also shows that each line can be back-projected to a plane. These planes can be presented as:

$$\pi = \mathbf{P}^T\mathbf{l} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{2.49}$$

$$\pi' = \mathbf{P}'^T\mathbf{l}' = \begin{pmatrix} \mathbf{A}^T\mathbf{l}' \\ \mathbf{a}_4^T\mathbf{l}' \end{pmatrix}, \tag{2.50}$$

$$\pi'' = \mathbf{P}''^T\mathbf{l}'' = \begin{pmatrix} \mathbf{B}^t\mathbf{l}'' \\ \mathbf{b}_4^T\mathbf{l}'' \end{pmatrix}, \tag{2.51}$$

where $\mathbf{P} = [\mathbf{I}|0]$, $\mathbf{P}' = [\mathbf{A}|\mathbf{a}_4]$ and $\mathbf{P}'' = [\mathbf{B}|\mathbf{b}_4]$ camera matrices for 3 views.
$\mathbf{A}$ and $\mathbf{B}$ are 3x3 matrices and vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ represent i-th columns of respective camera matrices. Vectors $\mathbf{a}_4$ and $\mathbf{b}_4$ denote epipoles in views 2 and 3 coming from the first camera. $\mathbf{A}$ and $\mathbf{B}$ are also infinite homographies respectively from the first to the second and the third camera.
The 3 image lines are obtained from a single line (L) in space. Therefore, the plains shown in equations 2.49, 2.50 and 2.51 are not independent and they must meet in the common line L in 3D space. There is, however, a requirement that the 4x3 matrix $\mathbf{M} = [\pi,\pi',\pi'']$ has rank 2. A relation among the lines l, l', l'' is reduced by this intersection constraint. If the rank of matrix $\mathbf{M}$ is 2, there must be a linear dependence between its columns. The matrix can be written as:

$$\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3] = \begin{bmatrix} 1 & \mathbf{A}^T\mathbf{l}' & \mathbf{B}^T\mathbf{l}'' \\ 0 & \mathbf{a}_4^T\mathbf{l}' & \mathbf{b}_4^T\mathbf{l}'' \end{bmatrix}. \tag{2.52}$$

Hence, the linear relationship is $\mathbf{m}_1 = \alpha\mathbf{m}_2 + \beta\mathbf{m}_3$. Observing that element (2,1) of $\mathbf{M}$ is 0, $\alpha = k(\mathbf{b}_4^T\mathbf{l}'')$ and $\beta = -k(\mathbf{a}_4^T\mathbf{l}')$ for some number k. Therefore:

$$\mathbf{l} = (\mathbf{b}_4^T\mathbf{l}'')\mathbf{A}^T\mathbf{l}' - (\mathbf{a}_4^T\mathbf{l}')\mathbf{B}^T\mathbf{l}'' = (\mathbf{l}''^T\mathbf{b}_4)\mathbf{A}^T\mathbf{l}' - (\mathbf{l}'^T\mathbf{a}_4)\mathbf{B}^T\mathbf{l}''. \tag{2.53}$$

This allows calculating i-th coordinate of l by:

$$l_i = \mathbf{l}''^T(\mathbf{b}_4 a_i^T)\mathbf{l}' - \mathbf{l}'^T(\mathbf{a}_4\mathbf{b}_i^T)\mathbf{l}'' = \mathbf{l}'^T(\mathbf{a}_i\mathbf{b}_4^T)\mathbf{l}'' - \mathbf{l}'^T(\mathbf{a}_4\mathbf{b}_i^T)\mathbf{l}''. \tag{2.54}$$

Simplifying:

$$\mathbf{T}_i = \mathbf{a}_i\mathbf{b}_4^T - \mathbf{a}_4\mathbf{b}_i^T, \tag{2.55}$$

the relation can be written as:

$$l_i = \mathbf{l}'^T\mathbf{T}_i\mathbf{l}''. \tag{2.56}$$

$\mathbf{T}$ is a trifocal tensor which can be presented as a set of 3 matrices $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$. The tensor has 27 elements but it has just 18 degrees of freedom which are independent. Therefore, one needs only 18 parameters to calculate the trifocal tensor as the others can be derived from ratios of elements.

3

Methods used in this thesis

The project had several steps. Firstly, preliminary processing on 1 camera video was made to assess if the approach is feasible. Then, the measurement setup was established considering number of cameras, placement of markers on infant's body, recording time, legal issues and hospital's possibilities. After that, marker detection was done using Scale Invariant Feature Transform (SIFT) [13] with additional code [6] to discard points of no interest. The detection was done separately for different colors of markers as they required different pre-processing methods. Later, Camera Calibration Toolbox [12] was employed to compute intrinsic parameters and synchronize all 3 cameras. This made localizing and tracking of markers in 3D possible. Views of markers from different cameras needed to be matched to be sure which points correspond to themselves. Then, algorithm was developed to form 3D trajectories of markers in consecutive frames. Particle filter of Sampling Importance Resampling (SIR) type with constant velocity model was chosen to estimate the true position of the marker in each time point. Finally, criterion for abrupt changes detection was defined after consulting with an expert. The process is presented in Figure 3.1.

3.1 Recording videos

In a previous study infant's movements were recorded with 1 camera placed above the bed and the video was provided for this project. While this setup was quite easy and convenient, it could not provide enough information. The markers placed on newborn's body could not be often visible because of occlusions related to movement and position. These occlusions posed a significant problem to markers tracking algorithm. What is more, study of medical literature proved that it is necessary to observe the complexity of child's movements to determine potential neurological problems [1]. Two-dimensional data was clearly insufficient to estimate, for example, rotations. Moreover, the frame rate was too low and motion blur artifacts were visible.

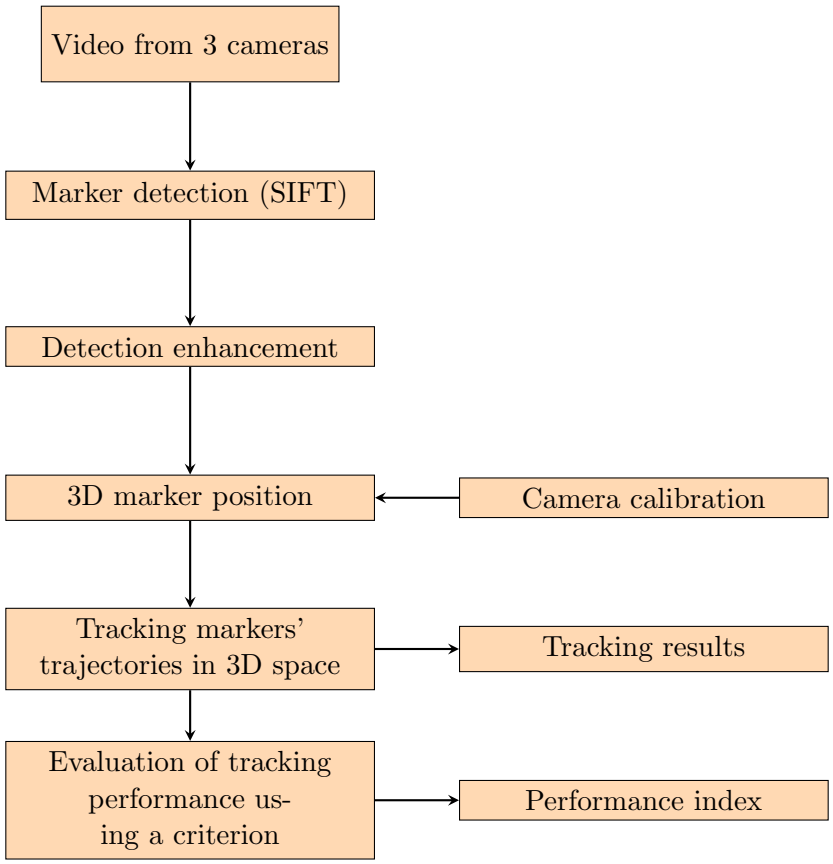
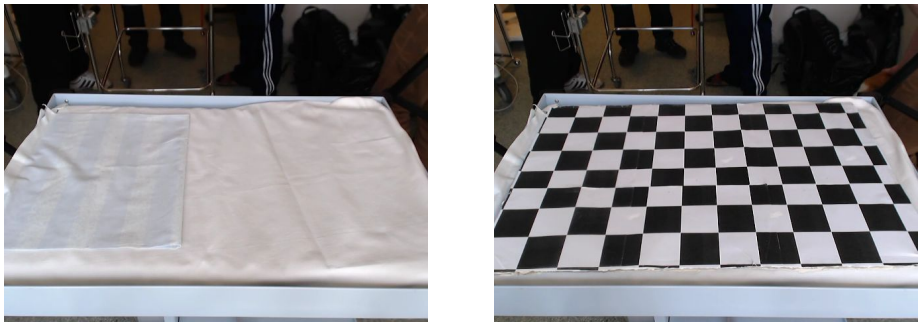


Figure 3.1: Stages of the project.

For reasons mentioned above, 3 cameras were used in this project. It increased the complexity of the system and added more stages to the work but it created possibility to solve the most important problems. Thanks to use of 3 cameras, occlusions in 1 or 2 views could be coped with by projecting markers from the view(s) where marker was visible to the ones where it was not. With the right placement of recording devices, the field of view was greater. Additionally, points of interest could be projected into 3-dimensional space with 3-view triangulation methods. Adding the third dimension to markers' positions drastically increased possibilities to track complex movements.
We could record subjects' movement by courtesy of a group of people, including medical doctors, students and subjects' parents who joined the measurement activities. Measurements were performed in 2 hospitals in Gothenburg. Sheets were chosen to be plain and in bleak colors. Infants wore paper strips dots of 2 different colors placed interchangeably, 2 on each limb. The videos were about 15 minutes long, depending on the subject's state. The bed and checker board used for calibration is presented in figure
Cameras were placed on tripods to see subject from different angles. They were connected to a laptop with VirtualDub program. The software does not allow to start



(a) (b)

Figure 3.2: Bed (a) and checker board (b)

cameras simultaneously so the starting frame in each video was marked with a quick movement of hand or checker board (used for camera calibration). Video resolution was set to 640x480 pixels and frame rate to 30 frames/second.

3.2 Marker detection

Marker detection is a necessary step to proceed. The basis of this step is Scale Invariant Feature Transform (SIFT). The markers were chosen in such a way as to be distinctive from the background. It is a critical feature which ensures that markers will be detected as candidate points. External Matlab toolbox [13] was chosen to perform the SIFT. Fortunately, authors of the mentioned toolbox left a high degree of freedom of settings to the user. The parameters need to be tailored to the application. Otherwise, points of interest might not be detected and algorithm may produce a lot of outliers. Relevant parameters to set are (with values that were proposed for this project):

- **Octaves** - number of octaves of the Difference-of-Gaussian (DoG) scale space, should be relatively high (3);
- **Levels** - number of levels per octave of DoG scale space, needs to be high to get more details (15);
- **FirstOctave** - index of the first octave, helps to control the size of detected objects (0);
- **PeakThresh** - threshold of intensity differences between objects and background, for high threshold only very distinctive objects (intensity-wise) will be detected (3);
- **EdgeThresh** - threshold of edge detection, for low values only points having clear edge will be found (5).

The detection was not successful in grayscale or RGB color space for red markers. Proposed markers have colors red and blue (or red and black) interchangeably but taking red and blue matrices of RGB image did not produce satisfactory results. Instead, RGB images were converted to YCbCr color space.
YCbCr is used in video and digital photography. It consists of 3 components:

- Y - luma component,
- Cb - blue-difference chroma, component
- Cr - red-difference chroma component.

Y is a luminance factor while Cb and Cr represent chrominance. In this color space Cb is a subtraction of color blue and luma $(B-Y)$ and Cr of red and luma $(R-Y)$. When converting from RGB space, the results are scaled and rounded and offsets are added. Therefore, standard 8-bit RGB image produces Cb and Cr matrices of range 16-240 [14]. Conversion from RGB to YCbCr color space is presented in equation 3.1.

$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256} \begin{bmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.1}$$

The pre-processing was needed to detect as many markers as possible but SIFT will produce more points than it is wanted. Excluding outliers was done using the method mentioned in section 2.2, where code was provided for tests in this thesis work. This resulted in only 1 keypoint per marker and rejection of most of spurious points.

3.3 Camera calibration

The aim of camera calibration was to obtain 3 camera matrices containing intrinsic and extrinsic camera parameters. It is necessary to estimate correspondences between different views and to back-project markers into 3D space.
Calibration is commonly done with checker board. The black and white squares form a grid which is suitable for corner detection. With the assumption that the lines are perpendicular, it is easy to estimate various distortions. It is suggested to use a board with even number of squares on one side and odd number on the other. Then, the same point of origin can be detected automatically in all images.
Camera calibration is a crucial part of this project. The 3 views cannot be effectively used without the knowledge of relative positions between cameras. It does not only allow to back-project the points into 3D scene. It is also necessary to check correspondence of the detected points using fundamental matrices. Finally, with known size of squares on the checker board, one can operate on physical units of distance when tracking objects.

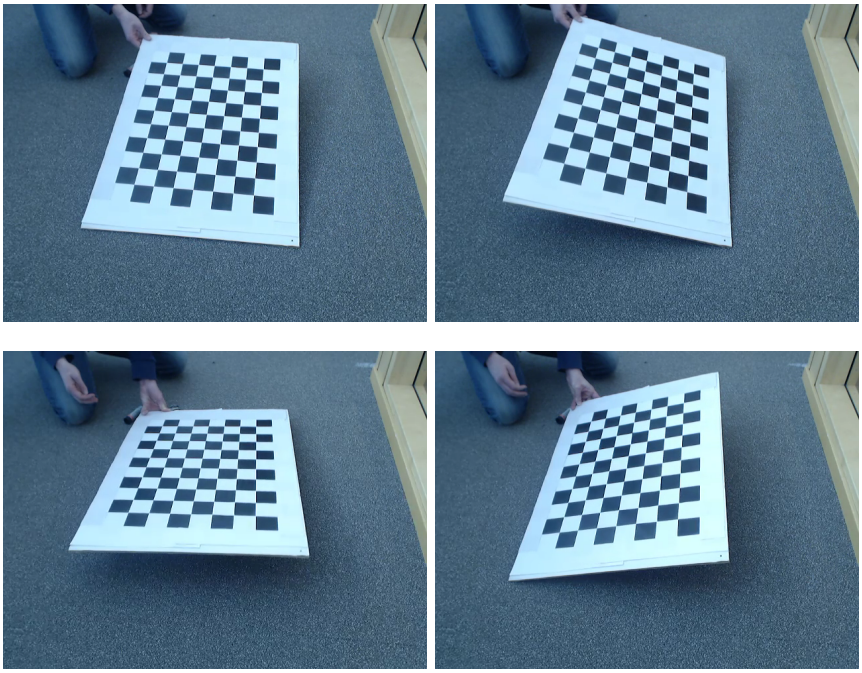


Figure 3.3: Calibration images

3.3.1 Obtaining intrinsic parameters

Extraction of intrinsic parameters can be done once for each camera. They describe the camera itself with no reference to the observed scene. It means that they will not change in different settings. Calibration was done with a checker board of size 8 by 11 black and white squares. Each square had a size of 53x53 mm. Measuring the square size allows to operate on real units in 3D. It is also required in calibration programs.
The board needed to be shown to camera from different angles for the process to be successful but the surface was visible in every image. It is done separately for each camera. To do the calibration of intrinsic parameters, one just needs a few photographs of the board seen from different angles with camera in the same place. After providing the size of checker board and single square size in millimeters all intrinsic parameters can be calculated. Four images were chosen here. They are visible on Figure 3.3 (1 camera). Camera Calibrator application was used which is a part of Matlab software. It automatically detects corner points and origin in provided images. Checker board needs to be, however, made according to suggested standard described before. After calibration, one can see reprojection errors to estimate if calibration was satisfactory. There is also Camera Calibration Toolbox for Matlab [12] which is more complicated to use but offers possibility to manually select region of interest and point of origin. Intrinsic parameters were saved for each camera for later use.
Intrinsic matrix is of size 3 x 3. It was computed using both Matlab's Camera Calibrator

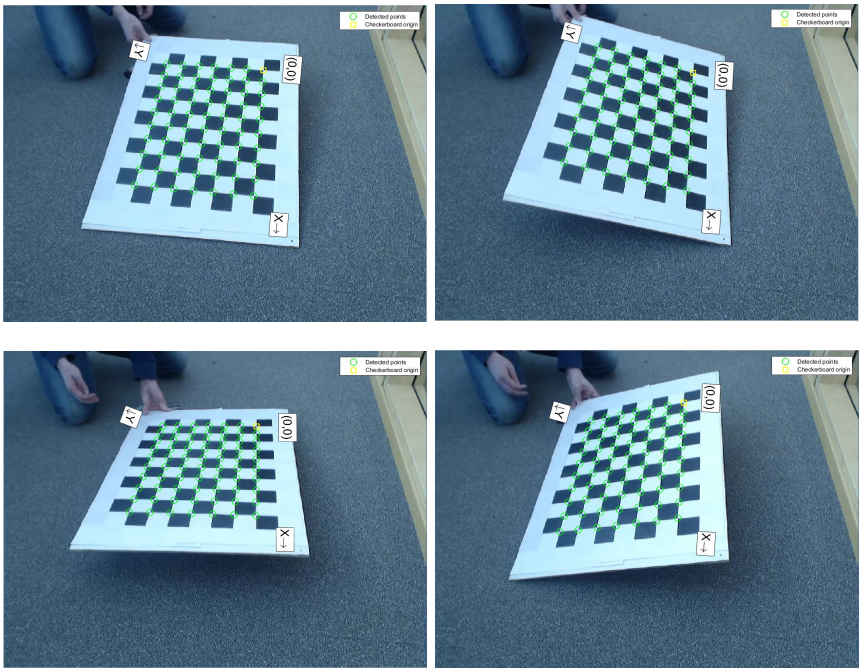


Figure 3.4: Calibration images with detected corners

and Camera Calibration Toolbox and compared. The algorithm detected corners very accurately in all cases as seen in Figure 3.4. The green circles indicate corners and the yellow square is the point of origin.

3.3.2 Obtaining extrinsic parameters

Extrinsic parameters had to be extracted for each video separately. They depend on relative position of cameras in the measurement system. Whole setup had to be done in different hospitals and rooms so each measurement video starts with putting checker board on a bed. If the system would be in everyday use in hospital, it would have a rigid setting and there would be no need for calibration every time it is used. Rotation and translation is calculated from one image only so any image from previous calibration can be chosen. It is assumed that cameras do not change positions during the time of measurement. Otherwise, calibration must be repeated. It is also critical that all the squares are detected and they have equal sizes. The square size is used again to make it possible to express the distance in real units. Simply put, extrinsic parameters describe the relative location of the camera with regard to the checker board. All the parameters are put together to form the camera matrix. It is a matrix specific to each camera (and its position) and it facilitates back-projection of objects and points of interest into 3-dimensional space.
Here, again, both Matlab and the external toolbox [12] had functions to calculate rotation

matrix and translation vector. In some cases, it was necessary to use Camera Calibration Toolbox as it provides the option to chose area of interest by mouse clicks. It facilitates the process when automatic detection fails or detects checker board somewhere else in the image.
Camera matrix $\mathbf{P}$ can be easily calculated using: intrinsic matrix $\mathbf{K}$, rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$:

$$\mathbf{P} = \mathbf{K} \cdot [\mathbf{R}|\mathbf{t}]; \tag{3.2}$$

where $\mathbf{R}$ is a 3 x 3 matrix, $\mathbf{t}$ is a 3 x 1 vector and $\mathbf{P}$ is a 3 x 4 matrix.

3.4 Position of markers in 3D

All the steps before lead to efficiently establishing 3D position of the markers. It is critical that all (or almost all) markers are detected and rare outliers cannot be found in more than 1 view. Calibrating the cameras needs to be also done accurately. The magnitude of position error depends on it and high error may make it impossible to discern between closely lying markers. Another important parameter is the frame rate of the videos. From the point of view of accuracy and algorithm development, the rate should be as high as possible. It corresponds to a small time interval between consecutive frames and position changes are not big. A high rate also reduces motion blur which is blurring of an image during a rapid motion. Such a phenomenon can drastically hinder the detection. On the other hand, high frame rate can pose a heavy load to the computer and data storage. Converting a video in, for example, .avi format to single frames is a time consuming process. Program detecting points, triangulating them and tracking is also greatly affected by increased size of data.

3.4.1 Matching objects in 2 views

Successful detection of markers is important but it is not enough. Objects detected in different views should be matched only if they correspond to the same one in 3D scene. Otherwise, performing triangulation will produce artificial 3-dimensional points which, in fact, did not exist in observed space. Proposed approach assumes that the marker must be visible in at least 2 views to be accepted. Triangulation cannot be done with just coordinates from just 1 view. This constraint also plays a filtering role. It may, although rarely, happen that detection program detects a point that is not really a marker (it just can have similar properties in the image). In such case, the spurious point will be rejected as it does not match with any point in other views.
As the first requirement is that the object is visible in at least 2 views, looking for correspondences is done pair-wise. All the operations are done for pairs: 1 and 2, 2 and 3, 1 and 3. Firstly, 6 fundamental matrices $\mathbf{F}$ are calculated from 3 camera matrices $\mathbf{P}$. There are 2 ways to obtain the fundamental matrices:

- from camera matrices,

- from point coordinate correspondences.

The first way is much easier and more reliable so camera matrices $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ could provide a set of matrices describing relations between all pairs of views: $\mathbf{F}_{12}$, $\mathbf{F}_{13}$, $\mathbf{F}_{21}$, $\mathbf{F}_{23}$, $\mathbf{F}_{31}$, $\mathbf{F}_{32}$. External toolbox was used here [15]

Preliminary matching

The idea behind pair-wise matching was that one can easily find candidates of a point in the other view when employing the right fundamental matrix. Assuming that $\mathbf{x}_1$ are homogeneous coordinates of a point in view 1 and $\mathbf{x}_2$ a point in view 2, they might be projections of the same object if this expression is true:

$$\mathbf{x}_2^T \mathbf{F}_{12} \mathbf{x}_1 = 0. \tag{3.3}$$

Nevertheless, it is still not enough to be sure that it is the right match. The fundamental matrix can only project the point from one view to the other as a line according to the rules of epipolar geometry. Therefore (3.3) shows only that the point $\mathbf{x}_1$ lies on this line. It is enough, though, to reject most of the points. In reality, the result of (3.3) for matching points is a relatively small number (higher than 0) describing distance from the epipolar line. One must set the right threshold as there is always some error. Figure 3.5 shows the relation between 2 views. The photo of chapel is taken from 2 different angles. Cross mark in Image 1 is mapped to a line in Image 2.

Figure 3.5: Point-to-line correspondence in 2 views [15].

After leaving only the points satisfying equation (below the threshold), the inverse problem is set. For example (3.3) it would be:

$$\mathbf{x}_1^T \mathbf{F}_{21} \mathbf{x}_2 = 0. \tag{3.4}$$

The algorithm for this process is shown in Algorithm 2. The same is done for 2 other pairs of views.

Algorithm 2: Preliminary matching

Data: Fundamental matrices $\mathbf{F}_{12}$ and $\mathbf{F}_{21}$, position of markers in 2 images $\mathbf{X}_1$ and $\mathbf{X}_2$
Result: Accepted matches $\mathbf{X}_{match}$.

```
for all points in X_1 do
    for all points in X_2 do
        Calculate distance_12 = |x_2^T F_12 x_1|
        if distance_12 < threshold then
            Calculate distance_12^inv = |x_1^T F_21 x_2|
            if distance_12^inv < threshold then
                X_match ← {x_1, x_2}
            end
        end
    end
end
```

Elimination of spurious points

It may happen that after preliminary matching some matches are not true. This happens because there can be more than 1 candidate point lying near the epipolar line as shown in Figure 3.5. It is necessary to employ the third view to discard these false positives. Unfortunately, it cannot be done as the same points is very rarely visible in all 3 camera views. However, the develped algorithm can discard many spurious points. Firstly, 3D position of the object is computed from preliminarily matched points in first and second view. It is possible with a use of 2 camera matrices and point coordinates in both images. After that, the 3D point is mapped on the third image using camera matrix $\mathbf{P}_3$:

$$\mathbf{x}_3 = \mathbf{P}_3\mathbf{x}. \tag{3.5}$$

Now, with the point in third image the procedure from (3.3) is repeated with matrices $\mathbf{F}_{13}$ and $\mathbf{F}_{23}$ to see if the point matches in the additional view. If these expressions are satisfied, the triangulation for 2 remaining pairs (in this example: 1,3 and 2,3) is done to clear all doubts. If the difference between 3D point obtained by all 3 pairs is small enough, the point is considered a true match. The idea is shown in Algorithm 3.

3.4.2 Generating trajectories

After matching the detected points in different views, triangulation is done from successful matches. Corresponding views of markers in different images are used to back-project the point into the 3D world. This process assigns a number of 3D points to every time unit in the processed video, which is a frame (or a set of 3 frames when considering all cameras collectively). It is a significant step forward but tracking cannot be done at this stage yet. Due to the number of objects higher than 1 and slightly changing throughout

Algorithm 3: Elimination of false positives

Data: Accepted matches $\mathbf{X}_{match}$ and fundamental matrices $\mathbf{F}_{13}$ and $\mathbf{F}_{23}$, position of markers in 2 images $\mathbf{X}_1$ and $\mathbf{X}_2$
Result: True matches $\mathbf{X}_{true}$

```
for all the pairs of points in X_match do
    Compute 3D point x_12
end
for all the 3D points do
    Map to the third view
    x_3 = P_3 x_12
end
for all mapped points x_3 do
    Calculate distance_13 = |x_3^T F_13 x_1|
    and distance_23 = |x_3^T F_23 x_2|
end
if distance_13 and distance_23<threshold then
    x_temp=x_match
end
for all the points in X_temp do
    Compute 3D point x_13 and x_23
    if |x_13 − x_12| and |x_23 − x_12|<threshold then
        x_true=x_temp
    end
end
```

the video, there should be a way to decide which point from frame $n+1$ belongs to particular point in frame n. Such an assignment is creating marker trajectories because they move in space and time.

It is not a simple issue and some assumptions need to be made. Also, the solution must take several events into consideration. Some of them are:

- marker occlusion,
- assigning multiple points from frame n to the same point in frame $n+1$,
- assigning one point from frame n to multiple points in frame $n+1$,
- detecting new targets.

Occlusion of an object happens when it is covered by something which is not an object of interest. In this application, markers can be occluded i.e. by an arm or a leg while the subject is moving. The use of 3 cameras minimizes the chance of such events

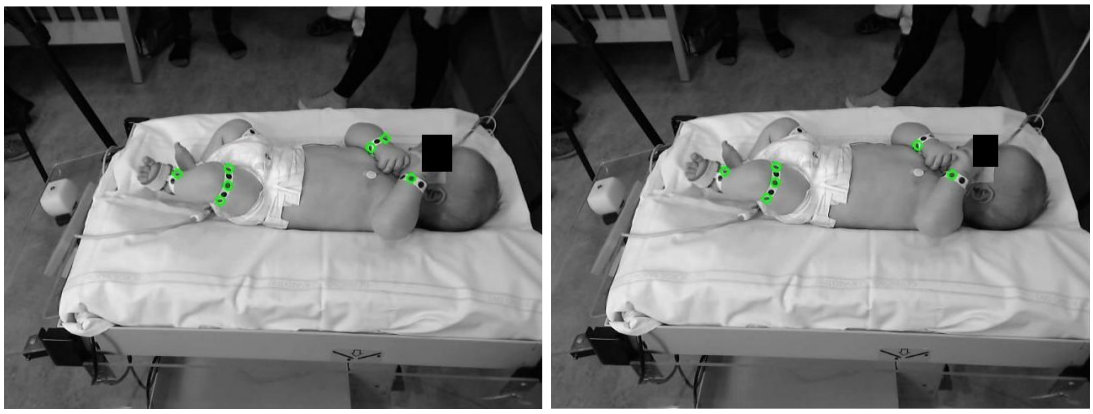


Figure 3.6: Disappearance of a target.

with comparison to 1 camera but the marker needs to be seen in at least 2 of 3 views. One may consider a duration of trajectory in frames, its lifetime. It refers to the time (or number of frames) for which the point exists. Therefore, appearance of a point is a target's birth and it ceases its existence with a target's death. It leads to 2 issues: when to decide that a target has been born and when to say that it is dead. The first parameter is especially important when detection does not work well and there might be frequent outliers appearing in single or small number of frames. Then, the program should wait some time before allocating the new point its trajectory. In this project birth of a target takes place immediately after its appearance. The latter is more vital in the case of occlusions. When the target is temporarily occluded, it disappears in a number of frames and may appear again after some time. Time to death should be set in such a way that it minimizes losing the point from trajectory. However, it is not a good idea to wait for its reappearance for too long as during this time nothing is known about the target. When the target dies, its trajectory is closed and the program will not search for new points for it. The 2D example of a disappearance is shown in Figure 3.6. Upper image shows detected markers in one frame and the lower presents the following frame. One target disappeared.

Another problem is deciding what to do with connections of type many-to-one or one-to-many. Choosing the right threshold for how far the marker can move between the frames minimizes occurrence of such phenomena. For a frame rate that is not exceptionally high, these situations often take place as markers may move a significant distance. When they are placed closed to each other, it creates a conflict as more markers are within the range of being possible predecessors. The inverse problem also exist, one marker sees multiple others as candidates for its potential trajectory. It is assumed that 1 point from frame n has only 1 correspondence in frame $n + 1$. Hence, there is no doubt that conflict points have to be at least discarded when there are too many candidates. When 2 points "fight" for the same pool of correspondences, they have to be distributed. One way to do this is

to assume that close lying points keep the same distance from each other in time. This approach has numerous disadvantages, though. Firstly, occlusions would jeopardize the success of this approach. Secondly, it is not easy to automatically decide which points are placed on the subject near each other and which are just temporarily in proximity because of the current position. Finally, realization of this approach is computationally exhaustive. The proposed solution to the general trajectory formation is presented in Algorithm 4.

3.5 Tracking markers

Having trajectories of different objects is very close to their tracking. However, measurements always incorporate some noise which distorts the true values of a signal. Such a signal should be filtered to get the best estimate as possible. In this project, positions of the markers include some noise too. The true position is not known but a right tracking filter can significantly de-noise the data.

The most known example of tracking filters is, without a doubt, a Kalman Filter. It is commonly used in various field of science and industry. The filter works on assumption that the process is linear and the noise is Gaussian. Particle Filters are more general and leave a place for more parameters to tune. One can specify i.e. the noise distribution or number of particles. The Particle Filter is one of Monte Carlo methods where a lot of samples are created according to given distribution function. After that, the samples are compared to observation and weighted so the ones with the highest likelihood are chosen. There are many types of Particle Filter which mainly concern sampling from a pool of created particles. The type chosen for this project is Sampling-Importance Resampling (SIR). The reason for choosing this method is how it handles the diminishing weights.

Successful filtering requires making good assumptions in the beginning. First thing to do is to write a model of behaviour of the object. A convenient representation is s state-space model. Such model describes a process as a set of algebraic equations. In this application, a constant velocity model was used as it is fairly simple and can well correspond to actual subject's motion. The model can be written as:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w}, \tag{3.6}$$

where $\mathbf{x}_t$ is a state vector at time t, $\mathbf{x}_{t-1}$ at time $t-1$, $\mathbf{A}$ is a state matrix and $\mathbf{w}$ is a system noise vector.

Transition of the states of a system in time is described by (3.6). It is a general equation. The difference between different models lies mainly in components of the state matrix. There is also an observation equation which models the observation assuming and observation noise:

$$\mathbf{z}_t = \mathbf{B}\mathbf{x}_t + \mathbf{v}, \tag{3.7}$$

Algorithm 4: Generating trajectories

Data: Matrix of 3D points in each frame $\mathbf{X}$, time to death t_d
Result: Trajectories $\mathbf{T}$

Initialize trajectories $\mathbf{T}$ and death counter $\mathbf{d}_c$
for *all points* $\boldsymbol{X}_{frame=1}$ *in first frame* **do**
    $\mathbf{T}_1$=$\mathbf{X}_{frame=1}$
    $\mathbf{d}_c$=0
end
for *each frame* n **do**
    for *each trajectory* $\boldsymbol{T}$ **do**
        for *each point in* $\boldsymbol{X}_n$ **do**
            Calculate distance *dist* between last point in $\mathbf{T}_i$ and current point in $\mathbf{X}_n$
            if *dist*<*threshold* **then**
                Add the pair to matches matrix $\mathbf{M}_n$
            end
        end
    end
    Filter $\mathbf{M}_n$ with internal function to resolve the conflicts in matches
    for *each* $\boldsymbol{T}$ **do**
        if *there is a match for* $\boldsymbol{T}_i$ **then**
            Add the point to trajectory and cancel the counter (for this $\mathbf{T}_i$)
            $\mathbf{T}_i$=$\mathbf{T}_i$,$\mathbf{X}_{match}$
            $\mathbf{d}_{ci}$=0
        end
        if *no match and* $\boldsymbol{d}_{ci}$<t_d **then**
            If there is no match, keep the last point and increase the counter
            $\mathbf{T}_i$=$\mathbf{T}_i$,last point($\mathbf{X}_i$) $\mathbf{d}_{ci}$=$\mathbf{d}_{ci}$+1
        end
        if *no match and* $\boldsymbol{d}_{ci}$==t_d **then**
            Close the trajectory
        end
    end
    for *points* $\boldsymbol{X}_n$ *that had no match* **do**
        Create new trajectories
    end
end

where $\mathbf{z}_t$ is an observation vector at time t, $\mathbf{B}$ is an observation matrix and $\mathbf{v}$ denotes observation noise.

The state model in this project is a constant velocity model and can be presented as:

$$\begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 0 & t & 0 & 0 \\ 0 & 1 & 0 & 0 & t & 0 \\ 0 & 0 & 1 & 0 & 0 & t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_{t-1} + \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix}, \tag{3.8}$$

where x, y and z are point coordinates in corresponding directions and $\dot{x}$, $\dot{y}$ and $\dot{z}$ are the velocity components.

Only the position of the object is of interest here so observation should be limited to the first 3 values of a state vector. Therefore, the observation equation has to look like this:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_t + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}. \tag{3.9}$$

Sampling-Importance Resampling has a big advantage over Systematic Importance Sampling filter. It deals much better with weight degeneracy. This event takes place over time where there are more and more particles of very low weights. SIR filter tackles this problem by calculating effective number of particles. If this number falls below some threshold, resampling occurs. Then, particles with higher weights have a higher chance to be selected and duplicated, substituting low-importance particles. All the probability distribution functions used in this project are Gaussian.

Number of parameters had to be set for the filter to operate effectively. Tracking in 3 dimensions required a relatively high number of particles which was set to 10000. Variance of system noise ($w_1 - w_6$) was found optimal at 0.1 while observation noise ($v_1 - v_3$) had values of 100. Resampling was performed when effective number of particles fell below half of the number of particles which is 5000.

4

Results

Measurements were done mostly at Östra Sjukhuset hospital in Gothenburg by a group of people including medical doctors. The first video was taken with 2 sets of cameras: regular ones with a cord and IP cameras. The advantage of the latter was that they come with a software which enables recording on all devices in the same time. They are, however, sensitive to weak internet connection. The other disadvantage is that their video quality and frame rate is too low for this application. Hence, webcameras (with cables) were chosen and they were controlled with a VirtualDub program. In all the cases, resolution was set to 640x480 pixels and frame rate to 30 frames/second. The first measurement setup with 2 sets of cameras is shown in Figure 4.1.

Figure 4.1: True path and detected path.

Figure 4.2: Subject with markers seen from 3 cameras: Camera 1 (left), Camera 2 (right), Camera 3 (down).

4.1 Recorded videos

The videos successfully captured the General Movements. The sample frames from each view are shown in Figure 4.2.
The cameras were placed around the bed to have in view as many markers as possible. It was important that they are not placed opposite to each other. In such a case, projection of points between the views using fundamental matrices would be dramatically hindered. There was also a practical reason to why angles between cameras with regard to the bed were not 120°, 120°and 120°. The tripods which held the cameras occupied considerable space and there was a need for clinician and parents to reach the subject. It was necessary for a number of reasons like:

- attachment and reattachment of markers,
- feeding the baby to calm it down,
- lifting the subject without decalibrating the system.

4.2 Marker detection evaluation

First part of method evaluation was analysis of marker detection. The SIFT parameters were set to guarantee very high sensitivity. However, some objects in the image had identical or very similar properties to the markers. Therefore, specificity was not always

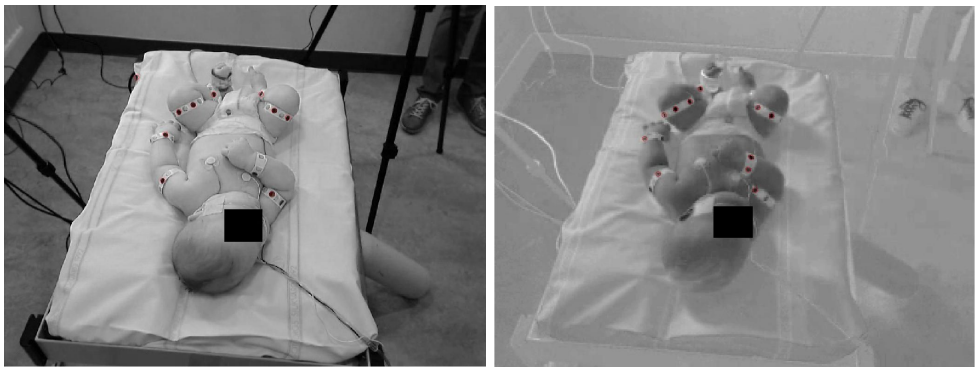


Figure 4.3: Images prepared for detection of black (left) and red markers (right).

	Black markers		Red markers	
	false/all found points	specificity	false/all found points	specificity
Camera 1	234/1059	77.90%	71/729	90.26%
Camera 2	2/804	99.75%	0/794	100%
Camera 3	0/804	100%	17/799	97.87%

Table 4.1: Sensitivity measured on 100 frames (770-869 in Set 1) in Camera 1, Camera 2 and Camera 3 views for black and red markers.

100%. The method was designed in a way that the rare outliers were not matched to any point in other views but there is still a chance of a false result. Black and red markers were detected separately as they required different pre-processing (see Chapter 3). Sample images fed to SIFT candidate point detection and enhancement program are shown in Figure 4.3. Left hand side picture was made for black markers detection and right hand side for red markers. The red circles are detected points. As these images do not carry a lot of useful information, they were used only to find positions of the markers.

Detection was evaluated on a length of 100 frames of randomly chosen part of video from Set 1. Set 1 consists of frames 13120-14120 from Video 2 from 22.04.2015. It was done for both black and red markers from every camera's view. Wrongly detected points were counted along with the total number of points to estimate specificity of detection. The result is shown in the Table 4.1. For each camera and type of marker, specificity was calculated as below:

$$(1 - \frac{false \quad points}{all \quad points}) \cdot 100\% \tag{4.1}$$

According to these results, red markers were not as much prone to produce outliers as the black ones. Black objects were abundant in the recorded scene which was the main reason for existence of false points. It is visible especially in Camera 1, where 2 points

belonging to bed were detected in almost every frame. These points are possible to discard with manual selection of region of interest. However, some of the false points belonged to the subject. In particular positions a false marker was found in the fold below the knee as it had same visual properties as black points. Some red points were also found on the mentioned bed in Camera 1 but there were not as many. False red markers were sometimes detected in Camera 3 on the subject's foot.

4.3 Projection of filtered trajectories

To estimate effectiveness of the developed method, it is crucial to do performance evaluation. Different kinds of qualitative evaluation were performed throughout the process of software development. For example, 3D points from each time step were plotted in Matlab to see their location. Later, a short movie was made to observe the markers' movement in time while following the calculated trajectories. Nevertheless, one cannot be sure if locations of these points are correct just by looking at such a plot. There was also a need to do quantitative evaluation to see the magnitude of error.

To do this, the points from trajectories were projected again onto all 3 views. Each set of points of consecutive time steps was presented on corresponding images from the chosen view. Only black markers were chosen as not to clutter the view. Projection from 3D to 2D is a simple operation when the camera matrix is known:

$$\mathbf{x}_3 = \mathbf{P}_3\mathbf{x}, \tag{4.2}$$

where $\mathbf{x}_3$ is the point projected onto view number 3.
This way, it was visible which point from trajectories corresponds to which marker on subject's body. However, there has to be ground truth provided to calculate the error between these points. In this case, ground truth was set by manually clicking on the centres of markers in consecutive frames. The ground truth was chosen only for view nr 3 as the points from longest trajectories were best visible there. However, 2 sets of videos with points projected onto every view were made to visually evaluate efficiency. Each set contains a recording from each camera which is 1000 frames long. On each frame there are red circles showing markers detected in that frame and blue points corresponding to mapped 3D points in that frame with 20 predecessors to visualize the movement. The green points is ground truth for chosen trajectories.

4.3.1 Marking of the ground truth

One of the methods of evaluation was calculation of difference between the marker visible on one of the views and the mapped points originating from filtered trajectories. Such an approach produces a quantitative result which can be easily used to evaluate efficiency of the algorithm. However, these mapped points have to be referred to true marker positions. These, due to the nature of the program, are not known. The solution proposed in this thesis is choosing the ground truth by clicking on the image manually.

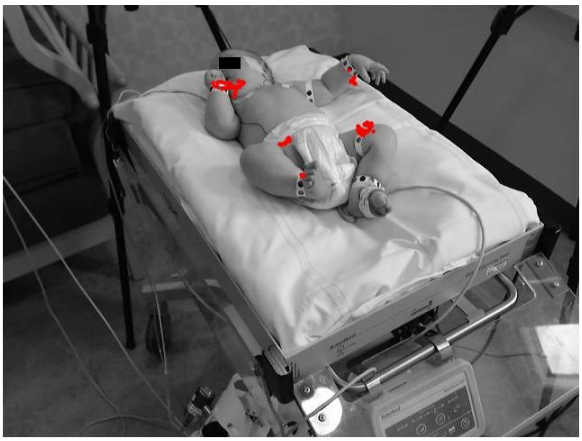
Matlab's function saves the coordinates of the selected point which can be later compared to any point in the image. This method is not perfect as selected points may not always lie in the very center of the marker but it is as close to "truth" as it can be. For example, if evaluated trajectory had a length of 433 frames, the ground truth was selected 433 times. The process was done on series of images from 1 camera view and the 3D points from filtered trajectories were mapped onto them. The choice of view was purely dictated by the visibility of the points of chosen trajectories.

4.3.2 Evaluation of Set 1 (Video 2 from 22.04.2015, frames 13120-14120)

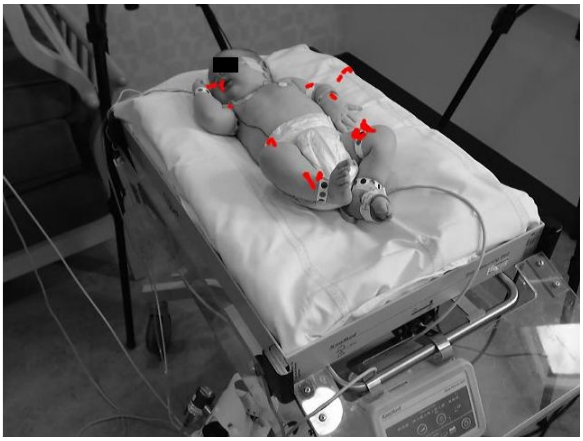
Set 1 is composed of 1000 frames (13120-14120 on Camera 1) from the second video recorded on 22.04.2015. Evaluation of the set is composed of 2 parts. Firstly, the trajectories were mapped onto each view to see if the points correspond to the movement of markers. It was done in 3 intervals where the subject's activity was high, each 50 frames long. The 3 intervals were chosen from the 1000-frame set where subject's activity was high. These were frames: 400-450, 450-500, 500-550. The same 3D points in the same frame intervals were mapped on view from Camera 1, Camera 2 and Camera 3. They are presented (as red dots) on Figure 4.4, 4.5 and 4.6.
Secondly, 4 trajectories were chosen for more precise evaluation. The ground truth was marked for them and can be seen as green dots. Figure 4.7 depicts these trajectories projected onto 3 views at the same time point for Set 1. These are the frames from the recorded video were each point was plotted with its 20 predecessors. This is why trajectories on images are no longer than 20. As mentioned earlier, the same point had to be detected in at least 2 views so not every marker has an assigned trajectory. Four long trajectories of different markers were selected for each set of videos and for the Set 1 they were located on (with their lengths of trajectory (in frames) in parentheses):

- left thigh (353),
- left wrist (101),
- right thigh (182),
- right wrist (212).

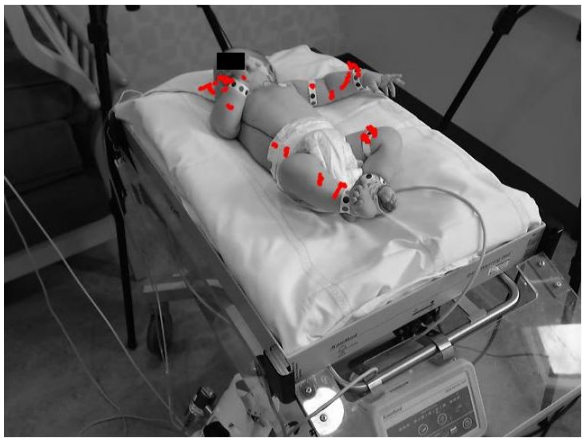
The exact positions of markers are shown in Figure 4.8 where green dots mark the mentioned objects.
Positions of points for the chosen trajectories were saved to calculate error which is a difference between these points and the ground truth. The ground truth was selected in Camera 3 view for all of these trajectories because of visibility, throughout the length of evaluated trajectory. Error statistics were put into Tables 4.2, 4.3, 4.4 and 4.5. Mean error was calculated over all frames of the trajectory. Minimal and maximal error are lowest and highest absolute differences respectively, found in the same time period.



(a)

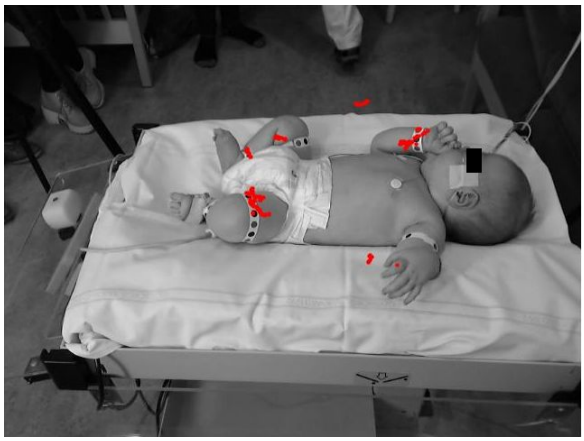


(b)

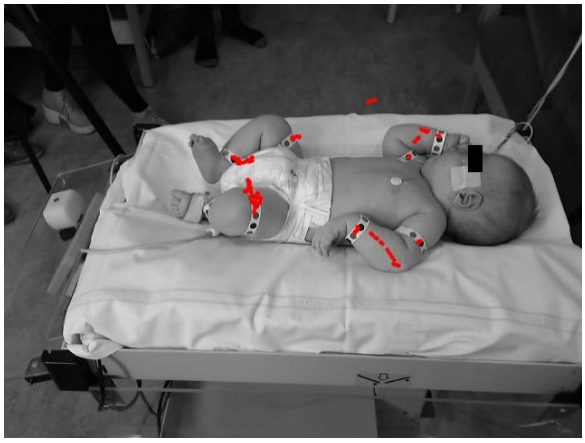


(c)

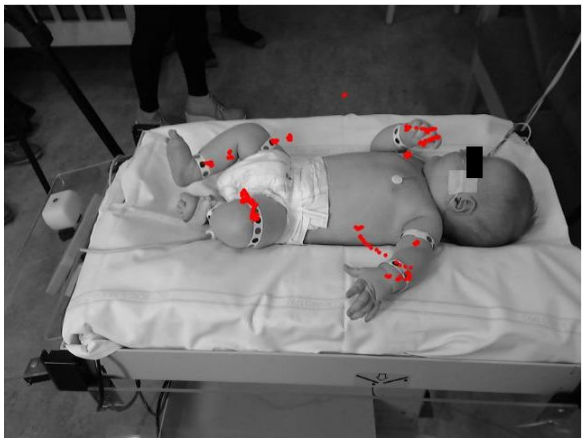
Figure 4.4: All trajectory points overlaid on the last frame from chosen frame interval in Set 1, Camera 1. Frames 400-450 on subfigure (a), frames 450-500 on subfigure (b) and frames 500-550 on subfigure (c).



(a)

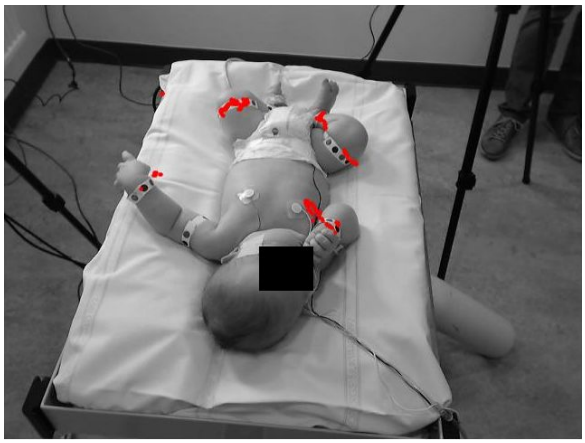


(b)

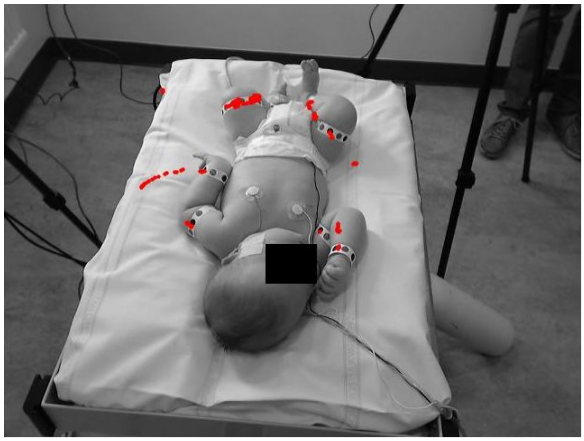


(c)

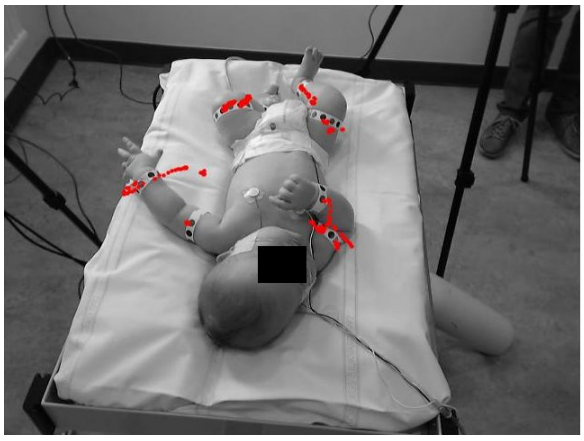
Figure 4.5: All trajectory points overlaid on the last frame from chosen frame interval in Set 1, Camera 2. Frames 400-450 on subfigure (a), frames 450-500 on subfigure (b) and frames 500-550 on subfigure (c).



(a)



(b)



(c)

Figure 4.6: All trajectory points overlaid on the last frame from chosen frame interval in Set 1, Camera 3. Frames 400-450 on subfigure (a), frames 450-500 on subfigure (b) and frames 500-550 on subfigure (c).

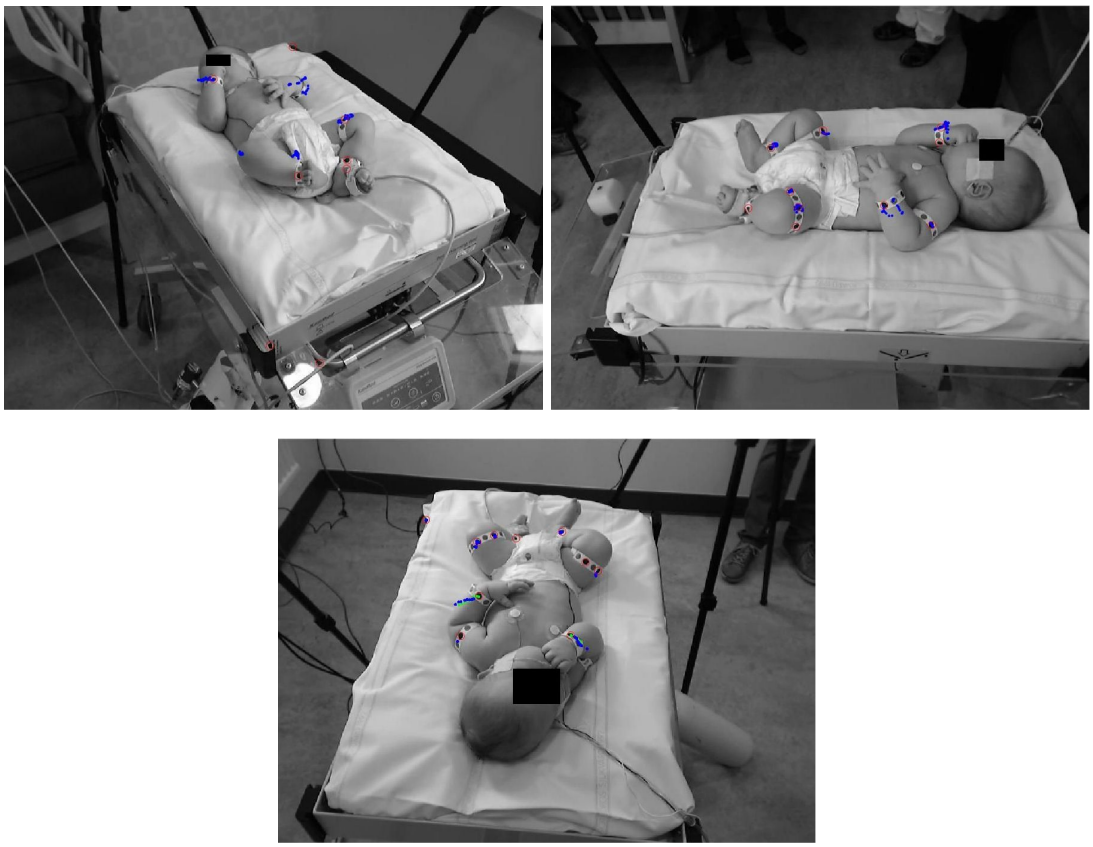


Figure 4.7: Frame with overlaid trajectories of markers for Set 1 seen from camera 1, 2 and 3.

Distance to the ground truth for marker on the right thigh	
Mean error	2.6506 pixels
Minimal error	0.1415 pixels
Maximal error	6.1438 pixels

Table 4.2: Error statistics for marker on the right thigh from Set 1 from Camera 3 view. Length of the trajectory: 182 frames (1-182 in Set 1).

Distance to the ground truth for marker on the left wrist	
Mean error	6.3421 pixels
Minimal error	1.0909 pixels
Maximal error	19.8277 pixels

Table 4.3: Error statistics for marker on the left wrist from Set 1 from Camera 3 view. Length of the trajectory: 101 frames (689-789 in Set 1).

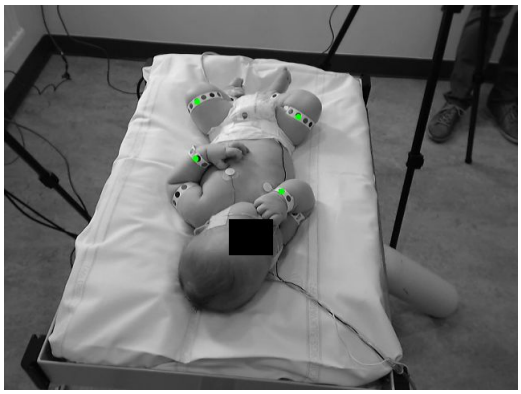


Figure 4.8: Placement of evaluated markers.

Distance to the ground truth for marker on the left thigh	
Mean error	2.8552 pixels
Minimal error	0.1193 pixels
Maximal error	10.9656 pixels

Table 4.4: Error statistics for marker on the left thigh from Set 1 from Camera 3 view. Length of the trajectory: 353 frames (438-790 in Set 1).

The errors are in acceptable range and they increase during sudden movements, as expected. The errors occur for numerous reasons: calibration, reprojection, camera resolution, filtering and projecting again onto the image plane. Additionally, images were enlarged to facilitate manual selection. This involves interpolation which might have introduced small errors too, despite being "the ground truth". These 4 trajectories were chosen because of their different locations on the body, their different movement and that they were continuous for a relatively long time.

One of evaluated trajectories was shown on a 3D plot in Figure 4.9. It represents the movement of a marker located on left wrist of the subject. The reason to choose this trajectory is its significant longitudinal movement which makes it easy to compare with

Distance to the ground truth for marker on the right wrist	
Mean error	3.5442 pixels
Minimal error	0.2168 pixels
Maximal error	17.0923 pixels

Table 4.5: Error statistics for marker on the right wrist from Set 1 from Camera 3 view. Length of the trajectory: 212 frames (737-948 in Set 1).

the ground truth in three dimensions. Other markers' movement was not as significant. The difference between the ground truth and the filtered position is mostly the result of chosen model and noise variances in particle filter.

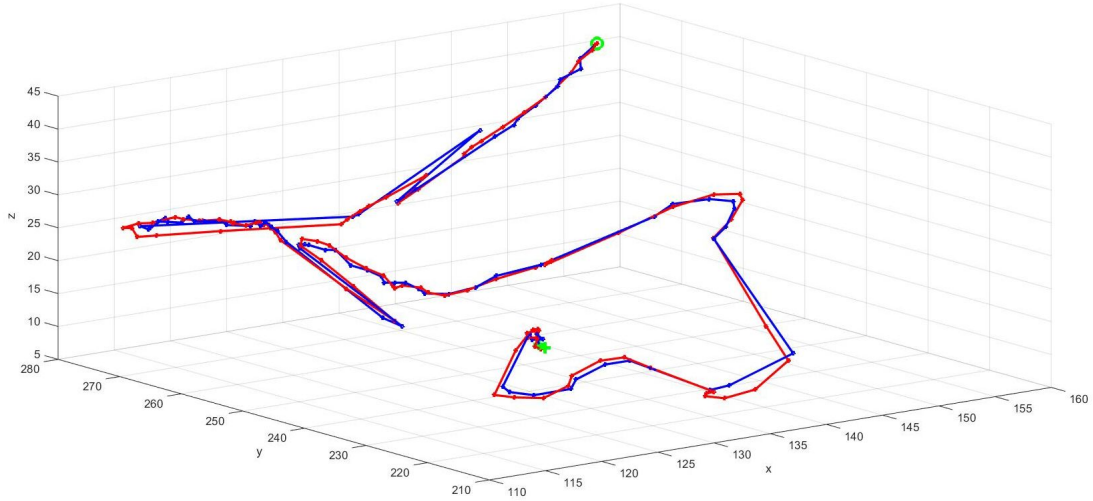
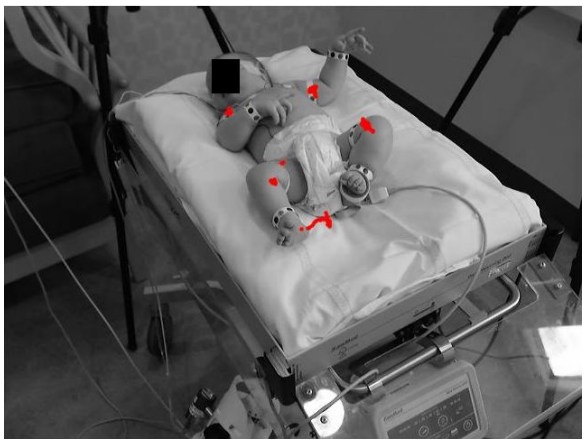


Figure 4.9: 3D trajectory of the marker located on left wrist (red line) and the ground truth (blue line) from Set 1 (frames 689-789). Green "O" indicates the starting position (for both) and green "+" is the end of trajectories.

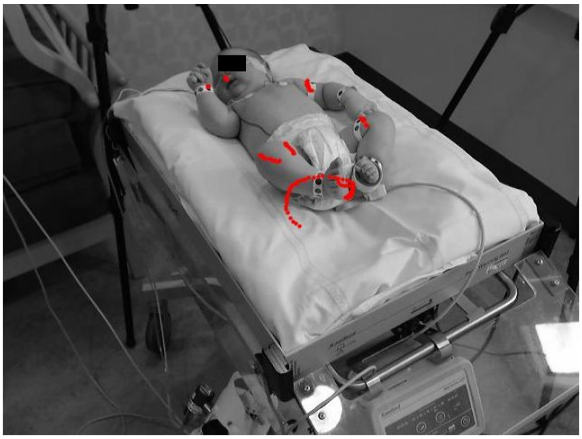
4.3.3 Evaluation of Set 2 (Video 2 from 22.04.2015, frames 17621-18621)

Set 2 is composed of 1000 frames (17621-18621 on Camera 1) from the second video recorded on 22.04.2015. The same process was done here as for the Set 1. Firstly, the trajectories were mapped onto each view to see if the points correspond to the movement of markers. It was done in 3 intervals where the subject's activity was high, each 50 frames long. The 3 intervals were chosen from the 1000-frame set where subject's activity was high. These were frames: 700-750, 750-800, 800-850. The same 3D points in the same frame intervals were mapped on view from Camera 1, Camera 2 and Camera 3. They are presented (as red dots) on Figure 4.10, 4.11 and 4.12.
The 4 chosen markers are shown in Figure 4.13. These are the frames from the recorded video were each point was plotted with its 20 predecessors. This is why trajectories on images are no longer than 20. For Set 2 the evaluated markers are on (with their lengths of trajectory (in frames) in parentheses):

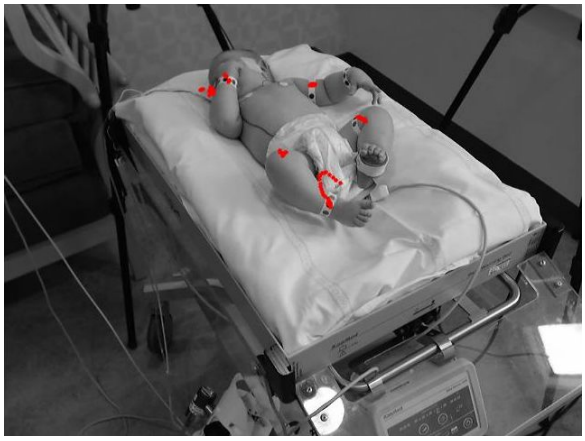
- left thigh (433),
- left wrist (459),
- right ankle (145),



(a)

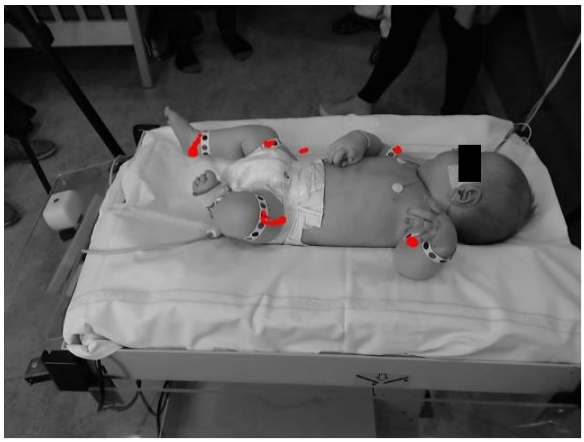


(b)

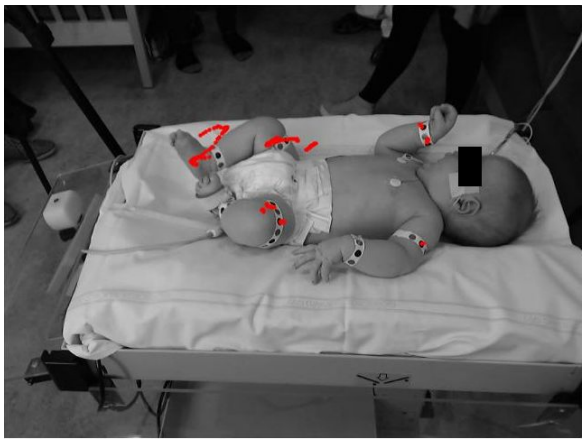


(c)

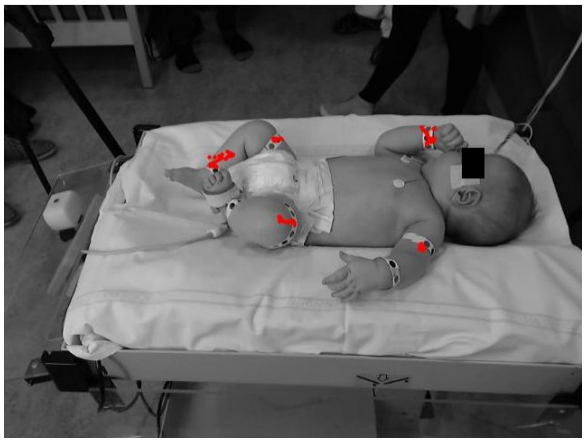
Figure 4.10: All trajectory points overlaid on the last frame from chosen frame interval in Set 2, Camera 1. Frames 700-750 on subfigure (a), frames 750-800 on subfigure (b) and frames 800-850 on subfigure (c).



(a)

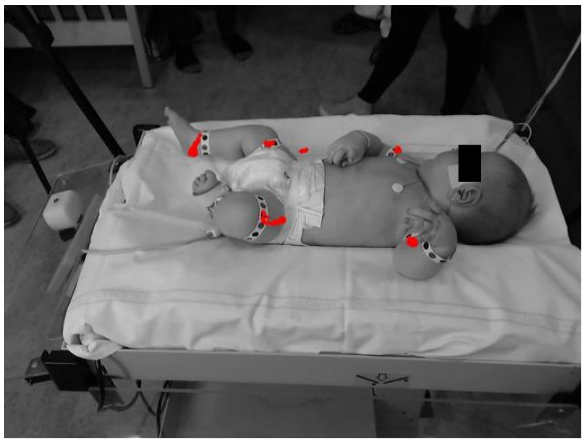


(b)

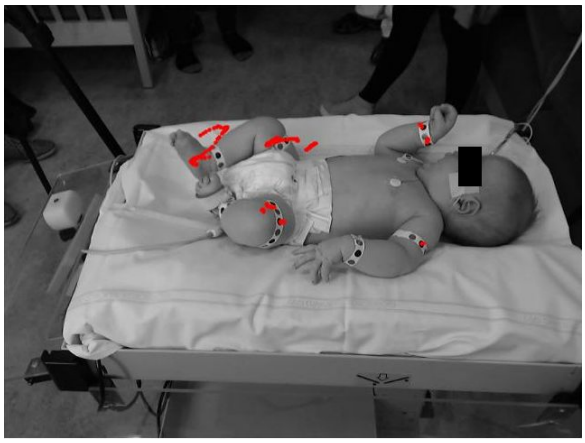


(c)

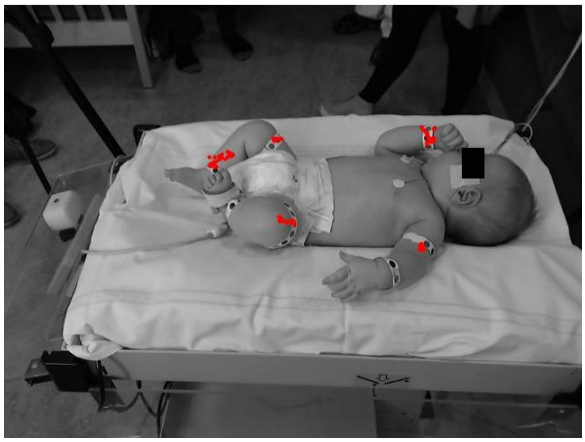
Figure 4.11: All trajectory points overlaid on the last frame from chosen frame interval in Set 2, Camera 2. Frames 700-750 on subfigure (a), frames 750-800 on subfigure (b) and frames 800-850 on subfigure (c).



(a)



(b)



(c)

Figure 4.12: All trajectory points overlaid on the last frame from chosen frame interval in Set 2, Camera 3. Frames 700-750 on subfigure (a), frames 750-800 on subfigure (b) and frames 800-850 on subfigure (c).

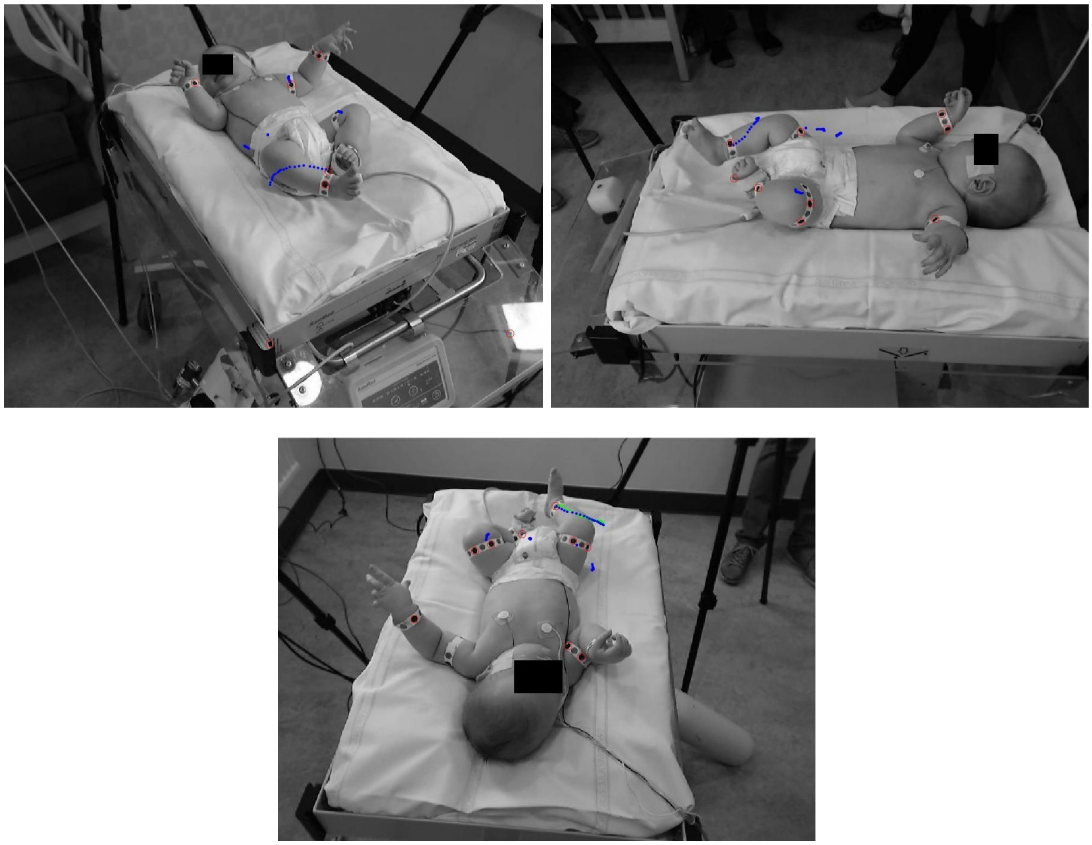


Figure 4.13: Frame with overlaid trajectories of markers for Set 2 seen from camera 1, 2 and 3.

- right wrist (287).

These trajectories were chosen for the same reasons as in Set 1. They were long and located on different parts of the subject's body. They were also subjected to different motion. Analysis of the error is shown in Tables 4.6, 4.7, 4.8 and 4.9. The ground truth was selected in Camera 3 view for all of these trajectories because of visibility, throughout the length of evaluated trajectory. The results were quite similar to Set 1. However, marker on the right ankle moved a lot but the error is not very big. The reason for this might be the fluency of the movement which allowed particle filter to adapt quickly.
One of evaluated trajectories was shown on a 3D plot in Figure 4.14. It represents the movement of a marker located on right ankle of the subject. As in the Set 1, the reason to choose this trajectory is its significant longitudinal movement which makes it easy to compare with the ground truth in three dimensions. Other markers' movement was not as significant. The difference between the ground truth and the filtered position is mostly the result of chosen model and noise variances in particle filter.

Distance to the ground truth for marker on the left thigh	
Mean error	2.6358 pixels
Minimal error	0.0700 pixels
Maximal error	8.0784 pixels

Table 4.6: Error statistics for marker on the left thigh from Set 2 from Camera 3 view. Length of the trajectory: 433 frames (10-442 in Set 2.

Distance to the ground truth for marker on the left wrist	
Mean error	6.6069 pixels
Minimal error	0.0854 pixels
Maximal error	19.6857 pixels

Table 4.7: Error statistics for marker on the left wrist from Set 2 from Camera 3 view. Length of the trajectory: 459 frames (1-459 in Set 2).

Distance to the ground truth for marker on the right wrist	
Mean error	3.9872 pixels
Minimal error	0.5295 pixels
Maximal error	11.2834 pixels

Table 4.8: Error statistics for marker on the right wrist from Set 2 from Camera 3 view. Length of the trajectory: 287 frames (181-467 in Set 2).

Distance to the ground truth for marker on the right ankle	
Mean error	2.8011 pixels
Minimal error	0.4142 pixels
Maximal error	7.4685 pixels

Table 4.9: Error statistics for marker on the right ankle from Set 2 from Camera 3 view. Length of the trajectory: 145 frames (684-828 in Set 2).

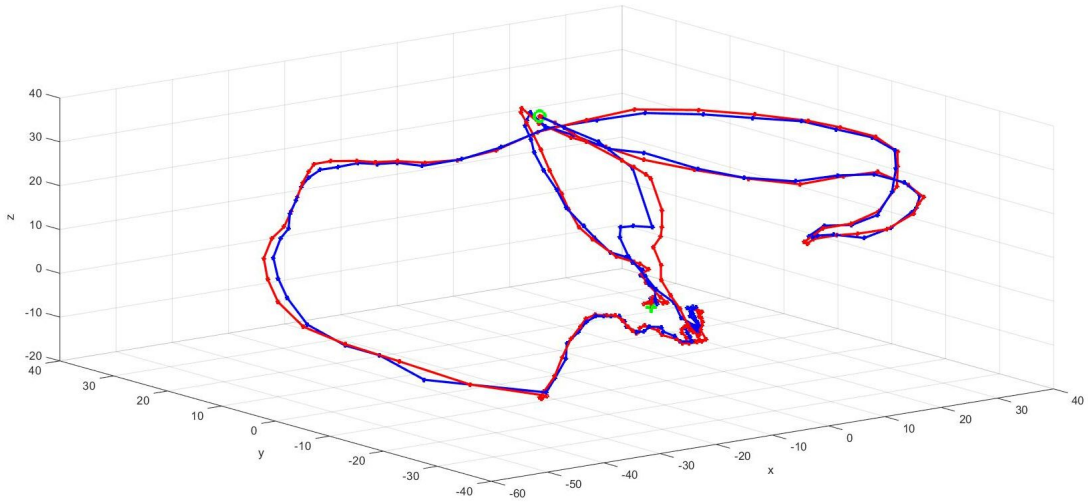


Figure 4.14: 3D trajectory of the marker located on right ankle (red line) and the ground truth (blue line) from Set 2 (frames 684-828). Green "O" indicates the starting position (for both) and green "+" is the end of trajectories.

4.4 Motion quantification

The final stage of the project was to quantify the movement. This procedure involves extracting a feature or set of features being descriptors of the motion. In this case, the sudden changes in trajectory needed to be derived as they indicate neurological problems. The method to quantify the movement is adopted from [6] with the same mathematical formulations (4.3) and (4.4). The only difference is that our tests are on 3D tracked trajectories instead of previously on the 2D trajectories. The mean residual power E_t for every time step was extracted using window averaging of prominent residual powers [6]:

$$E_t = \frac{1}{L} \sum_{i=t-L/2}^{t+L/2} \frac{1}{K} \sum_{k=1}^{K} ||\mathbf{e}_t^k||_2^2, \tag{4.3}$$

where

$$\mathbf{e}_t^k = \mathbf{z}_t^k - \mathbf{x}_t^k. \tag{4.4}$$

Both $\mathbf{z}_t^k$ and $\mathbf{x}_t^k$ are 3x1 vectors of 3D coordinates [x; y; z] and they represent unfiltered and filtered trajectories respectively. Therefore, $\mathbf{e}_t^k$ is a difference between the observed marker position and estimated location. E_t is averaged by window width L and the number of trajectories K which existed in this window calculated for time point t.
One measure of abrupt changes is their frequency. It can be done by counting these changes N_c in the time interval T. The changes are counted as E_t changes the direction between the chosen reference value:

$$N_c \leftarrow N_c + 1, if \qquad sign(E_t - \bar{E}) - sign(E_{t-1} - \bar{E}) \neq 0 \tag{4.5}$$

for t=2,...T; where $\bar{E}$ is mean value of E_t.

4.4.1 Quantification of Set 1

For a video of 1000 frames from Set 1, E_t is shown in Figure 4.15. First and last 50 frames in the plot need to be excluded as the window width is 100. Higher values between 350 and 600 frame are due to higher activity of the subject at that time.

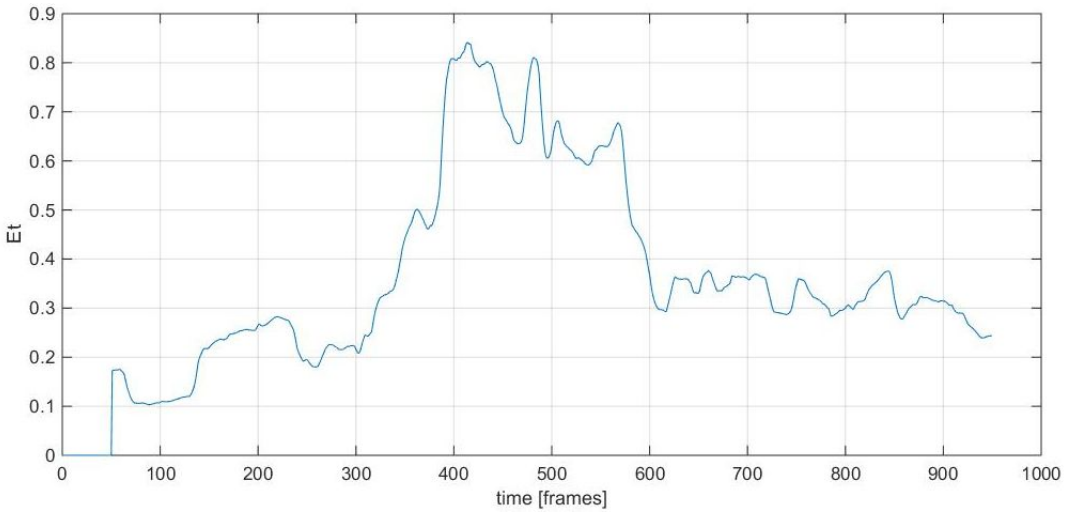


Figure 4.15: Prominent residual powers for Set 1.

For the curve in Figure 4.15, $N_c = 6$. It is expected that for subjects with neurological dysfunctions, this number will increase due to frequent abrupt movements. Unfortunately, such data is unavailable right now.

4.4.2 Quantification of Set 2

The same procedure was followed for Set 2. The prominent residual powers' distribution in time looks different as it is correlated with the movement. Here, subject was more active in the later stage of the video. It can be seen in Figure 4.16. $N_c = 2$, for this measurement.

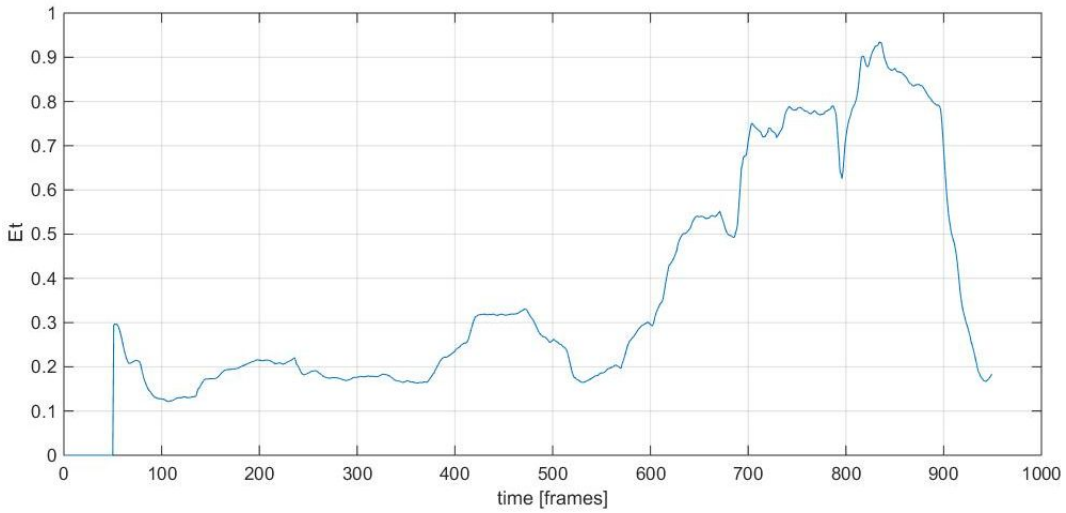


Figure 4.16: Prominent residual powers for Set 2.

5

Conclusion

The outcome of this thesis is a method to track and analyse limb movement of neonates using videos from 3 cameras. It is a new method of motion tracking in this field. It is hoped that the method will be successfully used in hospitals in the future to help newborns with neurological dysfunctions. The project did not aim to create a ready-to-use product. The goal was to develop a method which can be employed in the future. Therefore, there is still a lot that should be improved before it can be used in a practical application. Nevertheless, 3D movement tracking and quantification is a promising solution for helping newborns with neurological dysfunctions.

Most of the problems were strictly technical. For example, marker detection was very accurate but sometimes spurious point was detected. Rarely, markers were not detected due to change in illumination when marker reflected light at some angles. What is more, there were some small calibration errors which lead to adjusting tolerance thresholds in the marker matching program. In few cases, two different markers were seen as the same one because of correspondence ambiguity in epipolar geometry. The points lying on the epipolar line are seen as likely matches and it is impossible to choose the right one without additional information. Moreover, creating trajectories were done using the nearest neighbour algorithm. With markers placed close to each other during very fast movement, it may lead to ambiguities or false matches. Additionally, rotations along limbs' axes are hard to follow due to occlusions.

Bibliography

[1] L. Berthouze, M. Mayston, Design and validation of surface-marker clusters for the quantification of joint rotations in general movements in early infancy, Journal of biomechanics 44 (6) (2011) 1212–1215.

[2] H. Philippi, D. Karch, K.-S. Kang, K. Wochner, J. Pietz, H. Dickhaus, M. Hadders-Algra, Computer-based analysis of general movements reveals stereotypies predicting cerebral palsy, Developmental Medicine & Child Neurology 56 (10) (2014) 960–967.

[3] C. Einspieler, H. F. Prechtl, Prechtl's assessment of general movements: a diagnostic tool for the functional assessment of the young nervous system, Mental retardation and developmental disabilities research reviews 11 (1) (2005) 61–67.

[4] T. Lindeberg, Scale invariant feature transform, Scholarpedia 7 (5) (2012) 10491.

[5] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International journal of computer vision 60 (2) (2004) 91–110.

[6] L. Xu, I. Yu-Hua Gu, A. Flisberg, M. Thordstein, Video-based tracking and quantified assessment of spontaneous limb movements in neonates, in proc. of IEEE 17th int'l conference on E-health networking, application and services (HealthCom'15), 14-17 Oct 2015, Boston, USA (2015) 6 pages.

[7] M. S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking, Signal Processing, IEEE Transactions on 50 (2) (2002) 174–188.

[8] A. Doucet, A. M. Johansen, A tutorial on particle filtering and smoothing: Fifteen years later, Handbook of Nonlinear Filtering 12 (2009) 656–704.

[9] A. Doucet, S. Godsill, C. Andrieu, On sequential monte carlo sampling methods for bayesian filtering, Statistics and computing 10 (3) (2000) 197–208.

[10] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003.

[11] M. D. Seyed Masih Emami, Estimation of Visual Object Trajectory by using Video from Multiple Cameras, Chalmers University of Technology, 2012.

[12] J.-Y. Bouguet, Camera calibration toolbox for matlab.
URL `http://www.vision.caltech.edu/bouguetj/calib_doc/`

[13] A. Vedaldi, B. Fulkerson, VLFeat: An open and portable library of computer vision algorithms (2008).
URL `http://www.vlfeat.org/`

[14] M. S. Iraji, A. Tosinia, Skin color segmentation in ycbcr color space with adaptive fuzzy neural network (anfis), International Journal of Image, Graphics and Signal Processing (IJIGSP) 4 (4) (2012) 35.

[15] D. Capel, A. Fitzgibbon, P. Kovesi, T. Werner, Y. Wexler, A. Zisserman, Matlab functions for multiple view geometry.
URL `http://www.robots.ox.ac.uk/~vgg/hzbook/code/`