Detecting and analyzing rowing motion in videos

Bence Tamás
tamasbence92@gmail.com
Budapest University of Technology and Economics
Department of Telecommunication and Media Informatics

Gábor Szûcs
supervisor

October 26, 2016

Abstract

This paper describes an image processing approach capable for estimating the pose of athletes exercising on indoor rowing machines in video sequences. The proposed algorithm finds and tracks the hand, elbow, shoulder, ankle, knee, hip and head, and the line of the back also. This work is distinguished by two key contribution. The first is a new background subtraction method, which can reliable separate the silhouette of athletes under our assumptions made to the videos. The paper also introduce – as the second contribution – a skeleton fitting method to find the joints of the athletes based on the results of the background subtraction. This algorithm is based on anthropometric data and the special movement patterns and pose of rowing. The overall solution works on a real time setting, reaching 600 fps int the test environment.

I. Introduction

The automatic analysis of different sports spreads increasingly, because such systems can help the athletes, especially in competitive sports. In the past decades vast amount of visual information is gathered about sport events, trainings and tournaments. However the utilization of this information hasn't started yet in an organized, consequent way except some professional clubs of popular ball games. Analyzing the image content may help improve the motion patterns, thus could have a great impact in many sport disciplines, such as in rowing.

The goal of this work is to design a system, which can identify the major body parts of the human body (head, back, hip, knees, ankles, elbows and hands) in video sequences taken from side view of athletes, practicing on indoor rowing machines. Further target is presenting and visualizing the captured data in a well understood format and deriving useful numerical parameters (e.g. stroke rate, length of segments), in order to support the coaches' decisions.

The first step in this information getting pipeline is to separate the rowers from the background. The related work section presents the *KNN* and *GMM* based algorithms available in

the *OpenCV software library*. In the methods section a new algorithm is presented, which takes advantage of the special setting of the video. Our assumption is that the motion is recorded from a side view of a rower, the rower is fully captured by the video all the time and the background behind the athlete and his or her body have different colors. We also assume, that the athlete's shadow is not projected into the background or to the floor and the athlete is doing the session with a stroke rate between 15 and 45 stroke per minute.

The second contribution is the identification the position of the body parts. The related works section reviews the existing methods in the literature. Then the methods section presents the applied algorithm. The methods section also gives some examples of useful data, that can be deducted from the video and presents some visualization techniques.

The results section compares the speed and accuracy of OpenCV's built-in background subtraction methods with our presented algorithm. It also covers the result of the skeleton fitting. The discussion section suggests some further improvements and usability considerations in every-day coaching decision processes.

II. Related work

II. i. Analyzing sport sequences on videos

Originally the whole topic of sport analytics had developed from the simple match statistics collected by coaches of various ball games. This simple statistics – which can be very handy today also – then evolved to a new area of science. The field of sports analytics is continuously growing in the past two decades. The number of published papers related to this topic shows clearly this process as suggested by [Coleman, 2012]. The term *sport analytics* has became an umbrella term, with a lot of branches, like sport marketing analysis, match analysis, player performance analysis or biomechanical analysis.

The biomechanical analysis deals with the motion patterns of the various sport, examining the mechanical properties of the motion and suggesting modifications in this motion patterns in order to improve athletics performance or prevent injuries. Although the field of sports analytics is quite young, basic biomechanical studies were carried out already at the 19th century [Baumann, 1989]. Some authors date back even earlier the birth of biomechanics and relate to the work of Leonardo da Vinci in the 15th century. This field uses various techniques and tools to precisely record the quantities, which are characterizing a particular sport motion. Dynamometers, odometers, speedometers, strain gauges and angular measurement devices are commonly used to describe the motion of human joints and sports equipments. Video sequences and kinograms[1] are also very popular. Researchers usually marks by hand the important points on every frame of a video sequence. Alternatively they can use marker based systems, with active or passive markers, like an infrared light camera system. The disadvantage of these systems is their high price, and that they can be used only in laboratory environments. Nevertheless standard video camera's are much cheaper, and they are often used in every day coaching work [Wilson, 2008]. For example *KinoVea*[2] is a free and open source software, which allows coaches to manually mark different points in videos.

[1] Kinograms are photo sequences taken from a motion.
[2] http://www.kinovea.org/, visited at Oct., 2016

The science of biomechanic is also heavily used in the sport of rowing, because the fact, that rowing is a highly technical sport. The restrictions on the motion patterns lead to simple, but powerful mathematical models of rowing as suggested by [Chris Pulman, 2003] and [Valery Kleshnev, 2006]. There are commercially available measurements systems, which can measure the angle of oars, the position of the athlete and also the force applied to the blades. But these are very expensive, so most coaches must rely only on simple video footage. Best of our knowledge, there are not any specialized video tools specially designed for rowing coaches available.

As the topic of sport analysis has evolved, more complex video analysis techniques were developed in ball games. The new systems heavily rely on machine learning methods and can identify the ball, the different players, and the actions of players on match recordings [Thomas W. Miller, 2015]. For example *SAS*[3] has a tool for analyzing soccer matches. Despite the huge interest in ball games, in other sports there was not large research regarding in complex video processing of videos [Thomas W. Miller, 2015]. This paper describes a new solution capable of automatically extract key body part positions from pure video data.

II. ii. Background subtraction

The separation of the background and foreground pixels in video sequences is a well established and heavily studied topic in video processing. This is a very important, but also a very hard step in the processing pipeline. When the camera is fixed to a location, it is a reasonable supposition, that the background pixels of the image exhibits some regular behavior, which can be described by statistical tools. The emerging foreground object moving through the scene will not fit into the learned model, therefore can be detected. After a parametric or nonparametric pixel based model is applied, further improvements can be made using image based algorithm like meanshift [Comaniciu and Meer, 2002], particle filters [Nummiaro et al., 2003] or Kalman filters. The *OpenCV 3.1 software library*[4] provides some well-established statistical model based background subtraction algorithm as a built-in method.

Pixel based means that these algorithms considers only the previous values of a pixel at given location to decide whether the current value should be labeled as background(BG) or foreground(FG). The pixel is more probably background if:

$$\frac{p(BG|\mathbf{x^{(t)}})}{p(FG|\mathbf{x^{(t)}})} = \frac{p(\mathbf{x^{(t)}}|BG)p(BG)}{p(\mathbf{x^{(t)}}|FG)p(FG)} > 1 \qquad (1)$$

We can then rearrange the inequality and introduce a threshold value c_{thr} for deciding if a pixel is background:

$$p(\mathbf{x^{(t)}}|BG) > \frac{p(\mathbf{x^{(t)}}|FG)p(FG)}{p(BG)} = c_thr \qquad (2)$$

Here $p(\mathbf{x^{(t)}}|BG)$ is referred as the underlying *background model* as suggested by [Zivkovic and van der Heijden, 2006] and can be estimated from a training set $D_t = \mathbf{x^{(0)}}, \mathbf{x^{(1)}}, ..., \mathbf{x^{(t-1)}}$ created from the previous pixel valued observed at this pixel location. The pixel values $\mathbf{x}^{(t)}$ are d dimensional vectors.

[3] http://www.sas.com/en_us/industry/sports.html, visited at Oct., 2016

[4] http://docs.opencv.org/3.1.0/, visited at Oct., 2016

There are two major technique for estimating an unknown statistical probability distributions: the first are the *parametric methods*, which assume that the data follows a given probability distribution, like normal distribution [Witte and Witte, 2010]. In this case the goal of the estimation to estimate the parameters of this distribution. In case of normal distribution this means estimating the mean and variance of the Gaussian function. The parameters can vary with the type of underlying distribution. However the *non-parametric* methods do not assume any type of distribution on the training data. The simplest non-parametric method is the histogram method, which classifies the data into different *bins* and estimates the probability as the ratio of the count of examples in one bin over the total count. The big difference between parametric and non-parametric techniques is that, while the parametric methods use fixed number of parameters, the parameter number of non-parametric methods can grow with the size of the training data set.

The OpenCV's built-in *BackgroundSubtractorKNN* method implements a simple non-parametric statistical estimation based background subtractor described in [Zivkovic and van der Heijden, 2006]. The terminology here can be a little bit confusing, because OpenCV calls it *k-NN* i.e. *k-nearest-neighbor*, while the paper refers to it as a *kernel density estimation*. In fact the algorithm can be considered as a special type of kernel density estimation, which is also a k-NN.

The density estimation of a given vector $\mathbf{x}$ would be:

$$\widehat{p}\left(\mathbf{x}|D_t, BG+FG\right) = \frac{1}{|D_t|}\sum_{D_t} K\left(\frac{\mathbf{x}-\mathbf{x_i}}{h}\right) \quad (3)$$

Where $\widehat{p}\left(\mathbf{x}|D_t, BG+FG\right)$ refers to the common model inferred from the training data set, unfolding the background and foreground behavior also. Instead of $p(\mathbf{x^{(t)}}|BG)$, this probability should be estimated, since the true background and foreground classification is not known. K refers to the *kernel* function and h to the so called *bandwidth*, which will control the smoothness of the estimation. The kernel function can be any type, the most commons are gaussian, uniform, triangular, biweight and triweight kernels, but the type of kernel has a minor impact on the performance of the overall background subtraction as stated by [Elgammal et al., 2000]. Figure 1 shows an example estimation of a density function using different kernels. Take care, that the parameters of the kernel are d dimensional vectors, so this should be handled either by *crossproduct separable kernel extension* or *(circle or sphere[5]) symmetric kernel extension*. The OpenCV implementation uses the symmetric kernel extension, thus our equation 3 will modify to:

$$\widehat{p}\left(\mathbf{x^{(t)}}|D_t, BG+FG\right) = \frac{1}{M}\sum_{m=M-1}^{t-1} K\left(\frac{||\mathbf{x^{(t)}}-\mathbf{x^{(m)}}||}{h}\right) \quad (4)$$

Where $||\mathbf{y}||$ means some distance metric, in this case the Euclidean distance (or L2 norm). The h now is the diameter of a sphere and K is a uniform kernel, meaning that $K(u) = 1$ if $u < 1/2$ and $K(u) = 0$ otherwise. The notation $x^{(k)}$ k means the pixel value of a given location at time k. Thus not all the data is used for predicting the probability, but only the last $M-1$. The type of the kernel function's has only a minor importance, however the value of the diameter h has a large impact on the estimation. [Elgammal et al., 2000] suggested that $m/(0.68 * \sqrt{2})$ is a good choice, where m denotes the median of the absolute

[5]Circle refers to the method in case of 2 dimensional data, sphere in case of 3. In higher dimensions the technique has not widely used name.

differences $||\mathbf{x^{(i)}} - \mathbf{x^{(i-1)}}||$. But using a fixed kernel size would be not the best choice. In the OpenCV implementation a simple *balloon estimator* is used, which adapts the kernel size so, that the diameter is increased until a fixed amount of k training data is covered. This can be considered as a type of k-NN (k-nearest-neighbor) method. This estimation is not a proper density estimation, since the integral of it is not equal to 1. The parameter k is configurable in the software implementation. [Zivkovic and van der Heijden, 2006] suggest that a value of $k = [0.1M]$, where $[*]$ means the round-to-integer operator, will be good for treating outliers. Thus in this case we do not „count" the data samples lying inside the diameter h, but instead we are defining the minimum of the diameter h, which will contain k samples. The threshold value for the decision in this case will be $c_{thr} = 1/h^d$.

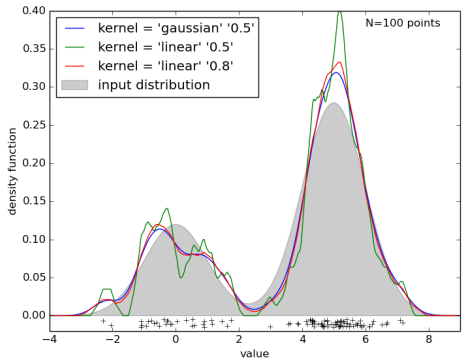

Figure 1: *A simple example probability distribution and it's estimations using different kernel density estimators. Generated with our Python script.*

In the practical implementation if D_t is large (if the video is long), keeping the samples and calculating the estimations would require too much memory and processor time. Thus only a random subset of the last M samples is used. Simple random sampling would lead to too sparse sampling thus a „short-term" and a „long-term" sample set is kept, and a denser and a sparser sampling is used respectively. The number of data samples can be controlled via a configurable parameter. If the total number of samples in the subsample is K, then the implemented algorithm will use $K/2$ sample from „short-term" and $K/2$ from „long-term". Also if a previous value was estimated as foreground, it will be removed from the subsampling.

OpenCV also offers various *Gaussian-mixture-model (GMM)* based algorithms. These have minor differences. This paper will describe in more detail the method used by the *BackgroundSubtractorMOG2* method. It was proposed by [Zivkovic, 2004]. GMMs are parametrized methods. As the name suggests, they use normal distribution. This is a good choice, because the fact, that due to the central limit theorem a lot of process in the nature follows a normal distribution. But instead using only one distribution, the method uses the *mixture* of N different Gaussian, with means $\mu_1, \mu_2, ..., \mu_N$ and standard deviations $\sigma_1, \sigma_2, ..., \sigma_N$. Gaussians are considered with a weights $\pi_1, \pi_2, ..., \pi_N$ respectively. These weights are non-negative and sum up to 1. Here the data is also d dimensional, so μ_i is a d dimensional vector, but for computational efficiency [Zivkovic, 2004] considers the covariance matrices as isotropic matrices, thus every dimension has the same σ_i^2 variance, and the dimensions are independent of each other. If we have estimations $\widehat{\mu}_i, \widehat{\sigma}_i$ and $\widehat{\pi}_i$ then the probability estimation will be:

$$\widehat{p}\left(\mathbf{x^{(t)}}|D_t, BG + FG\right) = \sum_{i=1}^{N} \widehat{\pi}_i \mathcal{N}\left(\mathbf{x^{(t)}}, \widehat{\mu}_i, \widehat{\sigma}_i{}^2 I\right) \tag{5}$$

The idea is illustrated in figure 2. Here a one dimensional distribution is estimated with

three component denoted by dashed lines. The continuous line represent the overall distribution.

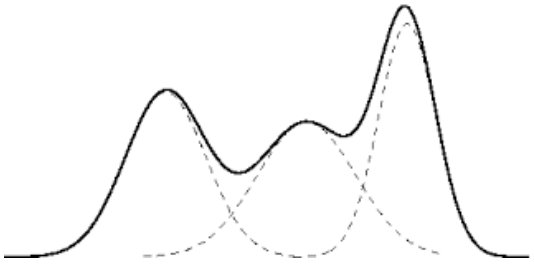

Figure 2: *An example GMM model with three component. Generated with Python script.*

As new data samples arrives, the parameters are updated according to the following equations:

$$\rho_i^{(t)} = x^{(t)} - \mu_i^{(t-1)} \tag{6}$$

$$\widehat{\pi}_i^{(t)} = \widehat{\pi}_i^{(t-1)} + \alpha \left(o_i^{(t)} - \widehat{\pi}_i^{(t-1)} \right) \tag{7}$$

$$\widehat{\mu}_i^{(t)} = \widehat{\mu}_i^{(t-1)} + o_i^{(t)} \frac{\alpha}{\widehat{\pi}_i^{(t)}} \left(\rho_i^{(t)} \right) \tag{8}$$

$$\widehat{\sigma}_i^{(t)} = \widehat{\sigma}_i^{(t-1)} + o_i^{(t)} \frac{\alpha}{\widehat{\pi}_i^{(t)}} \left(\rho_i^{(t)^T} \rho_i^{(t)} - \widehat{\sigma}_i^{(t-1)^2} \right) \tag{9}$$

Where α is the weighing parameter for an *exponential decay*. If we want to take into account approximately the last T pixel values, then we should choose $\alpha = 1/T$. $o_i^{(t)}$ is a binary variable, it is one for the *close mixture component* with largest π_i, and zero otherwise. A component is considered close to the pixel value, if the *Mahalabonis distance*([Jiwaei et al., 2012]) is less than a configurable threshold[6]. If there is no close component, then a new component is generated, with $\pi_{N+1} = \alpha$, $\mu_{N+1} = x^{(t)}$ and $\sigma_{N+1} = \sigma_0$, where σ_0 is a configurable parameter of the class. To keep the number of mixture components constant the component with the smallest π_i is dropped. As our assumption is that foreground objects will be represented by some clusters with small π_i weights, the first B large component can be considered as the background model:

$$\widehat{p}\left(\mathbf{x^{(t)}}|D_t, BG\right) \sim \sum_{i=1}^{B} \widehat{\pi}_i \mathcal{N}\left(\mathbf{x^{(t)}}, \widehat{\mu}_i, \widehat{\sigma}_i{}^2 I\right) \tag{10}$$

Here the parameter B means the number largest components which correspond to the $1 - c_f$ portion of the training data, thus can be calculated as:

$$B = \arg\min_b \left(\sum_{i=1}^{b} \pi_i > (1 - c_f) \right) \tag{11}$$

Here c_f is a configurable parameter and we assume that $\pi_k \leq \pi_j$ if $k < j$. Thus if a new object comes into the scene and remains static for a while, it will be added to the model as a new cluster. If enough time is elapsed, the corresponding weight become large and the new component will be considered to belong to the background.

The implementation of the GMM has one more trick. Instead of equation 7 the algorithm uses prior knowledge about the distributions, controllable with the *complexity reduction* parameter of the class. If we look the equation

$$\widehat{\pi}_i^{(t)} = \widehat{\pi}_i^{(t-1)} + \frac{1}{t} \left(o_i^{(t)} - \widehat{\pi}_i^{(t-1)} \right) \tag{12}$$

then with replacing the $1/t$ with $\alpha = 1/T$ we get back the original equation. Instead that we can introduce a prior knowledge with the so called *Dirichlet* prior and use a *maximum-likelihood* estimation. Ignoring the fine details, which are described in [Zivkovic, 2004] the final formula is:

[6] Which is 3 by default.

Table 1: *Parameters of the KNN background subtraction method*

KNN algorithm		
Parameter description		**Related methods**
sample size	M	getHistory setHistory
subsample number	K	getNSamples setNSamples
nearest neighbors	k	getkNNSamples setkNNSamples
decision threshold	c_{thr}	getDist2Threshold setDist2Threshold
shadow detection on/off	N/A	getDetectShadows setDetectSahdows
shadow detection threshold	τ_1	getShadowThreshold setShadowThreshold[8]

Table 2: *Parameters of the GMM background subtraction method*

GMM algorithm		
Parameter description		**Related methods**
background ratio	c_f	getBackgroundratio setBackgroundratio
complexity reduction	C_T	getComplexityReductionThreshold setComplexityReductionThreshold
sample size	T	getHistory setHistory
mixture components	N	getNMixtures setNMixtures
initial variance	σ_0	getVarInit setVarInit
max/min variance	N/A	getVarMin getVarMax setVarMin setVarMax
shadow detection on/off	N/A	getDetectShadows setDetectSahdows
shadow detection threshold	τ_1	getShadowThreshold setShadowThreshold[8]

$$\widehat{p}_i^{(t)} = \widehat{\pi}_i^{(t-1)} + \alpha \left(o_i^{(t)} - \widehat{\pi}_i^{(t-1)} \right) - \alpha C_T \quad (13)$$

where C_T is a small configurable class variable.

One of the drawbacks of the above described two method, the non-parametric KNN and the parametric GNN, is that they will mark the shadows projecting to the background as foreground. Therefore the OpenCV implementation has an additional mechanism for detecting shadows, which can be enabled or disabled in all two method[7]. The idea behind the implementation is suggested by [Cucchiara et al., 2002] and [Cucchiara et al., 2003]. The algorithm works in the *HSV* color space, because the shadows have not significant effect on hue values, but they will lower the value of the saturation. At a given pixel location the value $\mathbf{x}^{(t)}$ is considered shadow, if:

$$shadow = \tau_1 < \frac{\mathbf{x}^{(t)^V}}{B^V} < \tau_2 \wedge (\mathbf{x}^{(t)^S} - B^S) < \tau_s \wedge |\mathbf{x}^{(t)^H} - B^H| < \tau_h \quad (14)$$

Where τ_1, τ_2, τ_s and τ_h are parameters, but in the OpenCV implementation only τ_1 can be modified. The letters V, S and H in the upper indices denotes the components of the HSV pixel value, while B denotes the pixel value calculated from the background model. Either using the GMM or the KNN algorithm, there are more possible background pixel values, which should be considered.

Table 1 and 2 summarizes the different parameters of the presented background subtraction methods and shows it's corresponding methods in OpenCV.

[7] By default they are enabled.

[8] Note that there is a get/setShadowValue method in the class, but this has nothing to do with the algorithm settings, it affects only the format of output.

II. iii. Pose estimation, skeleton fitting

The term pose estimation may either refer to *2D pose estimation* or *3D pose estimation*, which means estimating the pose – thus calculating the position vectors of some given points in either a 2D or a 3D coordinate system – from an input sensor. Image or video based approaches are often used. The recording can be calibrated or uncalibrated. Previous case means that we know the exact position, angle and distortion coefficients of the camera respect to the world coordinate system. Human pose estimation refers to the process of recovering the human body part and joint positions from an image. *Skeleton fitting* is a related concept, it refers to calculate and putting some virtual bones to the human silhouette in the coordinate system of the image or a 3D world coordinate system [Szeliski, 2011]. Here virtual denotes that the bones shouldn't be real anatomical bones of the human body – instead some more simple model could be used, for example the spine can be modeled with one or two straight bone. Since pose estimation and especially human pose estimation is a very important task in computer vision, it has a huge literature [Perez-Sala et al., 2014].

Basically there are two main approach to this problem. The first one is the *model free* approach. In this case in the training phase a learning algorithm learns mapping between the *appearance space* (image data) and the body pose [Agarwal and Triggs, 2004]. This method could lead to fast performance and good accuracy for a certain type of actions (e.g. for walking). But these methods are limited by preprocessing or by poor generalization over different poses as suggested by [Perez-Sala et al., 2014].

The other approach is the *model based* approach. This means, that there are certain underlying assumptions about the human body, which will limit the searching space, thus lead to more robust, fast and accurate estimations. We can for example assume, that the arms are connected to the torso. This approach can be implemented in many different ways, as [Perez-Sala et al., 2014] suggests. The general detection process is shown by figure 3.

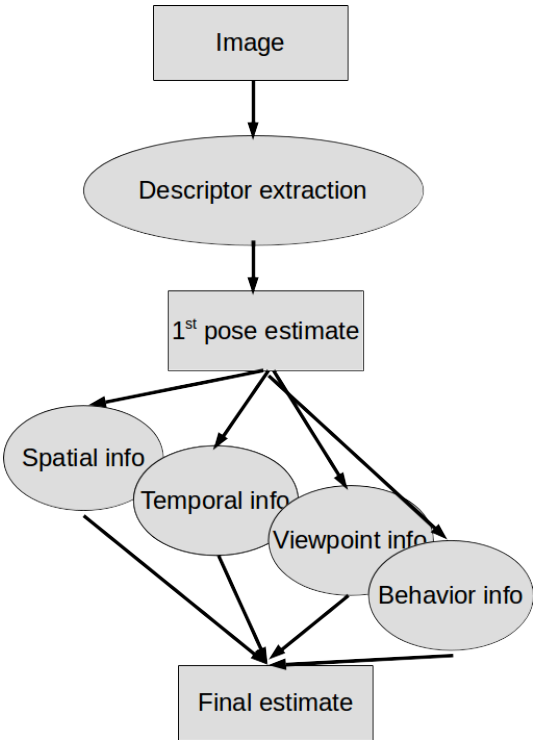

Figure 3: *Schematic idea of model-based pose estimation.*

The image is first processed by a *descriptor extractor*. In this context descriptor extractor is a very general term, it can be refer to any method, which will transform the raw pixel values of the input image – which are only numbers – into another numbers, which have a more concrete meaning regarding the human pose detection problem. A simple descriptor extractor can be a background subtraction or a segmentation algorithm, which outputs

the silhouette of detected human[8]. This could lead to a much simpler representation, still coding the most of the information about the human pose. Limitations of these methods are poor background subtraction or image segmentation quality. Better background subtraction can be done if depth information is available. Depth images can be acquired in multiple ways, for example with the *Microsoft Kinect* sensor [Microsoft, 2011]. From depth images the normal vectors of the surfaces could be also estimated [Shotton et al., 2013b]. But the depth imaging has its own limitations, for example Kinect can work only in an indoor setting, so *monocular* images are more common.

The most common descriptors used for pose estimations relies on color intensities and color intensity gradients. These descriptors in the first pass localize regions on images that seems to be important in some way. This is usually means finding corners, where the image gradient changes in two perpendicular direction. In the second pass, they compute a descriptor vector based on the pixel environment near the pixel in question [Perez-Sala et al., 2014]. Due to historical reasons the most common descriptors are the *histograms of oriented gradients (HOG)* and the *scale-invariant feature transform(SIFT)*. In the past years more advanced descriptors were proposed, like *BRISK*[Leutenegger et al., 2011] or *FREAK* [Ortiz, 2012], which are more robust and fast. Furthermore other descriptor extractors can be used like motion flows or logical operators [Perez-Sala et al., 2014].

After the descriptors are extracted (and usually a classifier is trained, either a classifier for detecting the full body or multiple classifier to detect various body parts), the result should be refined based on the assumptions of the model. This can mean, that the anthropometric properties of the human body are used to restrict the searching space. For example we can assume that one side of the leg is connected to the torso, and the angle between them should be between 90 and 110 degree. Based on anthropometric measurements we could also assume, that the forearm is approximately 9% longer than the upper arm [NASA, 1978]. Since legs do not appear on the visible frame in many TV shows and scenes in films, several works and datasets have been restricted to upper body estimation [Perez-Sala et al., 2014]. Besides *spatial information*, *temporal models* can also be used to enhance estimations. One can use the information about the *behavior*, specific motion, e.g. walking, to further restrict the searching space. Once the body or a body part is detected, tracking can be applied. One possible solution for tracking is the using of *Hidden Markov Models(HMM)*. Figure 4 represents a simple example HMM. The HMM consist N *hidden state*, denoted by $x_1, x_2, ..., y_N$ and M *observed state*, denoted by $y_1, y_2, ..., y_M$. We want to know, in which hidden state the system is, but we can only measure or estimate the observed states. However, based on the transition probabilities $a_{11}, a_{12}, ..., a_{1N}, ..., a_{NN}$ between the hidden states and the conditional probabilities $b_{11}, b_{12}, ...b_{1M}, ...b_{NM}$ we can infer the probabilities of being in the hidden states. The *Viterbi algorithm* is an efficient way to solve this problem. Additional considerations can be made, if we know the *viewpoint* of the camera. For example the problem of tracking in the cycling ergometer is highly simplified, if

[8]Technically it often means, that a result is a binary image, also called a *mask*, on which a pixel is set to zero if no human present at the given coordinate or set to one, if present. Another representation stores the contour line as an *chain* of vectors, but this is less common.

we know that the camera plane is parallel with the plane of the motion.

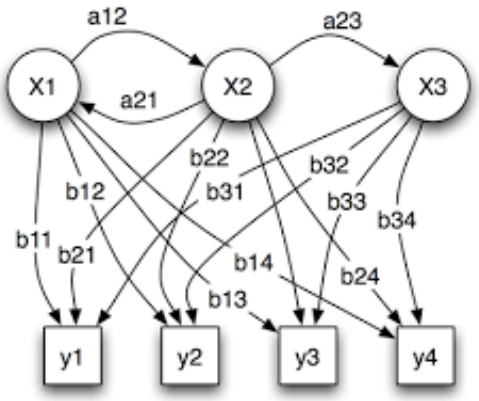

Figure 4: *A simple HMM with three hidden state and four observed state.*

For a more concrete example consider the *poselet* approach, suggested in [Bourdev and Malik, 2009] and [Bourdev et al., 2010], which got above on the generic pose estimators of that time[9] and nowadays can also achieve comparable results. In the first paper a 3D human pose estimation is done, while in the latter a 2D pose estimation, but the idea behind them is the same. The human body can be in different poses. A poselet describes a part of one's pose. In order to reliable estimate a pose, based on a large, annotated database about human poses *poselet activators* are built on training time, so that poselets are generated randomly and the best poselets are kept. The papers use the HOG feature descriptors to represent poselets. For estimating pose, the algorithm runs all the poselet activators on the image on different locations and scales. A poselet activators fires, if it finds a region, that is likely a poselet. It also produces a probability, which denotes how certain the activation is (these probabilities called *q-scores*). After this first phase the relationship of the different activations is taken into account, so if activations are consistent with each other, the newly computed *Q-scores* will be higher. In case of outliers Q-scores will be low, so they will be dropped. Then the consistent activations are clustered together to give the final estimation. The idea is presented on figure 5. The drawback of this approach is that it is not real time.

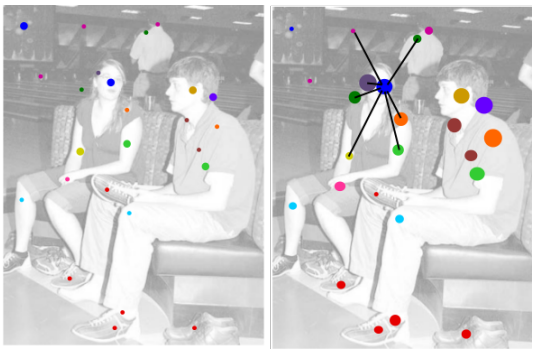

Figure 5: ***Left:*** *An example image with poselet activations. The size of the circles reflects the q-scores.* ***Right:*** *The same image with Q-scores. The consistent activations are connected. Figure is taken from [Bourdev et al., 2010].*

III. Methods

In this section a real time video processing framework is presented, which is capable of reliable detect an athlete performing a rowing workout on indoor rowing machines. The presented framework reports the position of the hand, elbow, shoulder, ankle, knee, hip and head and the line of the back. More accurately in case of joints, the estimated position could be the anatomic or the biomechanic axis. The previous means the exact location, where two bones are linked, while latter is a

[9] The method was published in 2009.

more practical and considers points close to the anatomic axis, which can be measured easier. The exact definitions of the axes among various body parts can be found in the related literature [Baumann, 1989]. Current work tries to estimate the biomechanic center of joints(ankle, knee, hip, elbow and shoulder). In case of hand and head, the body part's center of mass is the target.

The rowing motion is a very complex *cyclic* motion, which means, that more or less the same motion is repeated several times. Rowing utilizes all body parts and all important muscle groups, resulting in complex patterns, and sometimes – in case of beginners – a lot of technique errors. Rowers sit on a sliding seat in order to extend the movement to legs also. The rowing stroke begins at the *catch* position, where the legs are compressed, shins are vertical, the body leans slightly forward and the arms are extended, the shoulders are relaxed. Then the *driving phase* begins by pushing with the legs backward, while maintaining the same body position and keeping the arms straight. When the legs are almost extended, the body begins to swing to carry on accelerating the boat. Finally with arm pull the rower arrives at the *finish* position. Here the movement (compared to the boat) changes direction and the *recovery phase* begins, with doing all movements inversely: first the arms are straightened, then the body swings forward and finally the legs begin to bend and the rower arrives to the catch position again. Then this sequence is repeated several times. An example rowing sequence is presented in figure 6. In a rowing boat the hands are traveling along an arc, while in case of an indoor rowing machine the hands move along a straight line. Except that there is no difference between the two motion. Since analyzing an indoor scene seems to be much simpler (because the stable lighting conditions, constant background and fixed camera), this paper considers the case of indoor rowing[10]. As mentioned, rowing is a cyclic sport. This allows to define several important metrics, which can be related to rowing performance. The *stroke rate* is the most commonly used metric to describe the motion. In fact it is an another name for the frequency and in water sports its common unit is $1/min$. There are other important and intuitive metrics, like the length of the stroke (the path traversed by the hand compared to the boat), the contribution percentage of different body parts to the length, the vertical distance between the hands at drive and recovery phase[11] and so on. Length and angle metrics can be seen and at certain accuracy measured in video sequences. There are other important metrics, like force generated on the oars, which cannot be visually recognized. Section IV will provide further information about this.

The framework relies on a couple of assumptions regarding to the video and to the motion. These assumptions are summarized in table 3. The most important is related to the viewpoint: we assume, that the plane of motion [12] is parallel with the camera plane and the recorder is fixed. Also very important, that the rower and the background have different, distinguishable

[10]Keeping in mind the long-term goal: analyzing videos about on water training.

[11]The hands should be lowered at recovery phase, to avoid contact with the water.

[12]Since a rower moves back and forth along an axis, every point of its body is moving along a plane. For example if you consider a point of its hand. It can move vertically „up" and „down", and „forward" and „backward" from the body, but it always will stay in the same plane. Thus the important aspects of the motion can be captured from one viewpoint, from the „side-view".

Figure 6: *A whole cycle of the rowing motion. Taken from [Valery Kleshnev, 2006]*

colors, the background and the lighting conditions stay constant during the recording. All the time, the rower's full body is captured by the recording, and the rower is not too far, i.e. the bounding box around the image area, in which the rower is moving covers at least half of the whole image. The shadow's of the rower shouldn't appear on the image (or they should be very light) and shadow shouldn't cast to the rower's body. The athlete should row without any extraordinary big mistake in rowing technique or any incidental motion and should maintain stroke rate between 15 and 45spm[13]. Except to athlete now moving objects are captured by the camera.

III. i. Background subtraction

The simplest approach to background subtraction would be using OpenCV's built-in background subtraction algorithms, like the KNN and GMM methods described in section

[13]This is actually not a strict requirement, because rowers rarely do workouts below 20spm or exceeds the 40spm stroke rate.

Table 3: *Assumptions made to the input video.*

Criterion	Description/Status
lighting	lighting should stay constant
shadow by athlete	no
shadow on athlete	no
other shadow	allowed
viewpoint	side view (camera plane and motion plane is parallel)
viewpoint	fixed
field of view	athlete is fully captured
field of view	athletes bounding box area >= half of image area
rowing motion	smooth motion, no big rowing technique errors
stroke rate	between 15 and 45 spm
other objects	no moving object

II. But this leads to poor segmentation, because the athlete moves always in the same region of the image, thus the assumptions made to these algorithm do not hold. For example if at a given pixel location the athletes body pass at the finish position, that pixel can contain, let's say 30% of the time the athlete and 70% of the time the background. In this setting the GMM method would recognize the real background and also the athlete as two component of the background model and label both of them as background. This paper proposes a totally different approach, which is based on the assumptions of table 3 and capable of reliable differentiate the rower and the background. A deeper comparison of the new method against the KNN and GMM algorithms is given in section IV.

The core idea behind the proposed method is that as the athletes moves on the scene, some of its points can be captured by subtracting

the previous frame from the current frame and calculating the absolute value and then thresholding it. This will result in a binary image, where the one value will indicate a huge change between frames, and a zero value will indicate no or negligible change. For better illustration of the concept consider the HMM presented in figure 7. This shows the process on a pixel level. On the top of the figure, there are four states, which cannot be observed based on the thresholded different image. The names denotes if the pixel belongs to foreground or to background. For example $BG - FG$ is the state, where the pixel is now background pixel, but at the previous frame, the pixel belonged to the foreground. The arrows with $a_1, a_2, ..., a_8$ denotes the possible transition between states, and their probabilities. Note, that we don't know these probabilities and don't have any method to estimate them, so the Viterbi-algorithm will not work in this HMM. But the difference image encodes one of the two states from the bottom at every pixel. Again, we don't know the probabilities of $b_1, b_2, ..., b_8$, but based on the assumptions summarized in the previous section, we can deduce, that $b_1 \gg b_5$, $b_2 \gg b_6$, $b_3 \gg b_7$ and $b_4 \gg b_8$. The $BG - FG$ and $FG - BG$ changes will occur near the athletes contour, these states will cause the most of the contours. So now if we calculate the center of the observed contour points[14], it is *very likely*, that it will in the bounding box of the rower [15].

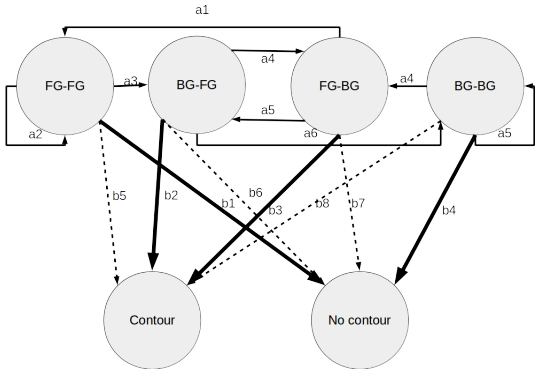

Figure 7: *The possible states after the subtraction of neighborer frames at pixel level. The top shows the possible states, which we cannot observe from the difference. The bottom show the two observable state.*

Consider figure 8 for an another aspect. If the rower is moving with speed v, it's height on the image plane is h, the time difference between to consecutive frame is Δt and we calculate the thresholded difference of frame i and $i + N$, where N is a small number, then the number of the contour points is roughly proportional to the height of the rower, and the distance he or she moved. So the number of contour points:

$$|contour| \approx hvN\Delta t \tag{15}$$

Based on these facts we suggest the following method[16]: on the first predefined number of frames [17] we determine the maximum number of contour pixels ($max_{|contour|}$) that can appear on a frame. Let us consider the frames with more than $c_{thr} max_{|contour|}$ contour point. Call them *good* frames, while the other frames *bad* frames. In the software implementation the $max_{|contour|}$ variable can be

[14] And if the athlete does not change much position between two frame. But this will hold, since we restricted the stroke rate to be less than 45 spm, and a video recording contains at least 24 frame per second.

[15] If we would apply more restrictions regarding the rower's size on the video and the stroke rate, we could estimate the b_i probabilities, and – very roughly the a_i probabilities also. Based on that it would be possible to give an exact number on the event, that the contour center is in the bounding box. But this isn't the goal of this paper.

[16] We will refer to it as *area based background subtraction*.

[17] $FPS * 4$ would be enough, where FPS means frame per second.

continuously updated with an *exponentially weighted moving average* scheme in order to reflect the newer observation with more weight. For every predefined number of frames[18] an independent $max^{new}_{|contour|}$ value is calculated. The cumulative $max^{cum}_{|contour|}$ value can be computed as: $max^{cum}_{|contour|} = (1-\alpha)max^{cum}_{|contour|} + \alpha max^{new}_{|contour|}$ [19]. For the good frames we calculate the center of the contours. A sample series is plotted in figure 9. To this series we apply a denoising. In this paper we applied a simple exponentially weighted moving average for this, with high „α" value. A more sophisticated solution would be to transform a slice of the series with the *FFT algorithm*[20], and apply a band pass filter with lower and upper frequencies of $15/60 = 0.25Hz$ and $45/60 = 0.75Hz$[21] In this manner searching for local maxima and minima, the exact time of the catch and finish can be acquired.

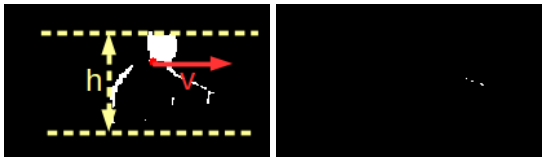

Figure 8: *The subtracted contours of the athlete.* ***Left:*** *Here the athlete is moving with v velocity to the right and is h pixel height on the image. the athlete is moving forward.* ***Right:*** *the athlete changes direction at the catch position. There are only few contour points.*

With this piece of information we can now construct the first version of the background image, which we will call *contour based* background image. Consider a *good* frame at the catch position. We apply to the corresponding

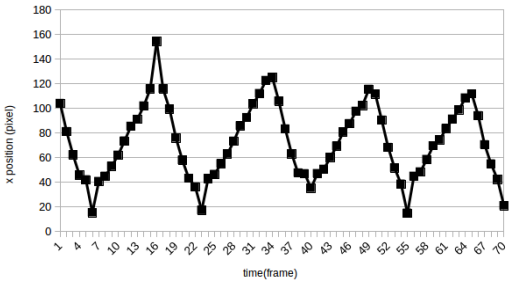

Figure 9: *The position of time contour center against time.*

contour image a *morphological eroding operation*, with a small rectangular *structuring element* to remove noise. An erode operation slides the structuring element through the vertical and horizontal axis of the input image and substitutes every pixel value, with the minimum of the pixel values belonging to the region under the structuring element. For a more detailed information see [Szeliski, 2011]. After locating the leftmost pixel (denote its x coordinate as x_{split}[22]) we can take the left part of the chosen good frame at catch position up to the coordinate x_{split} and the right part of a good frame at finish position beginning from the x_{split} coordinate, and compose them. Note that the shins still remain on the background image, with this method, they cannot be eliminated, so the result will be noisy around them. As we acquired the background image, we can now get the foreground mask with a simple subtraction and a thresholding operation. This process is shown on figure 10.

Based on our experiments, if the rower is facing left instead of right, the described method will give also fairly good segmentation, but not a perfect one. To solve this issue using this inac-

[18] $FPS * 4$ frames.

[19] For α 0.25 is sufficient.

[20] I.e. *Fast Fourier Transform* [Press et al., 2007] [Nise, 2011].

[21] Based on the assumptions made to the stroke rate.

[22] To be more precise we should extract half of the eroding structuring element's size from the leftmost pixel's x coordinate.

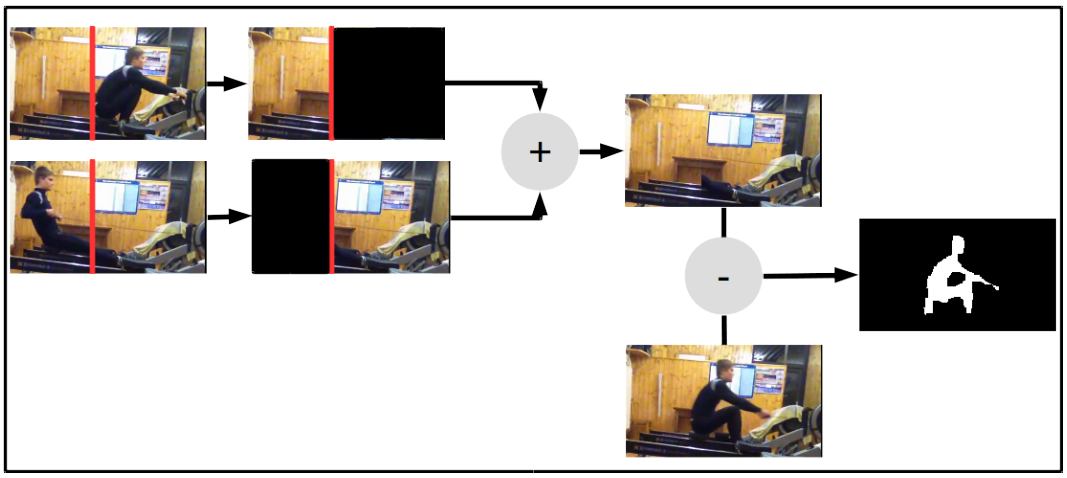

Figure 10: *The process of background subtraction. On the left good frames for catch and finish are acquired and the appropriate x_{split} x coordinate is calculated. Then the two images are merged to form the background. With this background and the current frame a foreground mask is calculated.*

curate background image, the software tracks the head position, and if the head is further at the (hypothesized) finish position from the left side of the rower's bounding box, than it is from the right side of the bounding box at (hypothesized) catch position, then the rower is facing left, thus the image is flipped and the whole background subtraction process is repeated, this time resulting a better background image.

If the background image is already available, when a new frame arrives only a subtraction and a thresholding operation is needed to compute the segmentation. This is very fast, much more faster than the methods described in section II. However, the framework continuously recalculates the background image, so it can adopt to slow changes of background. With the cost of more computation even more sophisticated methods can be used, like modeling the background pixels with a normal distribution, or even with GMM. Section IV presents the performance of the solution in more detail.

After the contour based background image is constructed, a second version could be constructed[23], using these results, with the same manner described above. The only difference is that this step instead of the difference of two frame, use the difference of a frame and the contour based background image. This will result a body silhouette instead of contours. The idea behind this second round, is that with using only one frame at time, it leads – in theory – to better localized estimation in the time domain[24]. But we found that this step does not improve the segmentation accuracy.

After the background subtraction if the recording is low quality some noise can appear on the segmentation mask. This can be reduced by considering only the largest connected region of the mask or using morphological operations. The software implementation uses the builtin *findContours* and *contourArea* methods for this step.

[23] Call it *silhouette based* background image.

[24] In contour based background creation method, where two consecutive frame are subtracted, the contour is more „scattered".

III. ii. Pose estimation

This paper demonstrates a simple approach for pose estimation. However this simple approach can produce good quality results due to the very specific body posture and accurate background subtraction. The whole estimation is based on the foreground silhouette and do not use any other visual information. It heavily relies on the anthropometric measurements published in [NASA, 1978] and on other temporal, spatial, behavioral and viewpoint informations also.

The algorithm outputs the 2D image coordinates of the biomechanic center of ankle, knee, hip, elbow and shoulder, the center of hand and head and also the line of the back, represented as series of straight lines.

For tracking the head, the implementation always use only the current frame. The mean proportion of the head and neck-torso length in adult population is 0.339. To make sure that the whole head is captured, we use a bit oversize, 0.38 proportion value. We cut out the top region of the silhouettes bounding box, then apply a *distance transform* operation. The center of the head then is at the maximum point. The distance transform is an image transformation, which calculates for every pixel location the distance of that pixel to the closest zero pixel[25], using some distance metric [Szeliski, 2011]. We use the builtin OpenCV method *distanceTransform*, with $L2$ distance metric and a mask size of 3.

The x coordinate of the hip is determined with similar calculations. However for y coordinates we scan the frame at finish position and take the center of thighs. Since thighs are horizontal this is equal to the hip's y coordinate. Based on our experiments this is a very robust estimation in this setting. This is demonstrated on figure 11.

Figure 11: *Determining the hip position.*

The point of the shoulder can be determined in the same manners as in case of head. The ankle position can be determined as the bottom right foreground point at catch position. Since the ankle is fixed to the rowing machine, it moves only a little. The program use this estimation for all video frames.

For localizing the hand, we crop the image above the thighs, then only the right side of the bounding box is kept. In this cropped image with a distance transform the center point is calculated.

Estimating the elbow and knee position is a bit more tricky. Because now the hand and shoulder or the ankle and hip positions are known and the elbow and knee moves along a plane[26], we can use a linear equation system. Let us assume that the hip is at $(hip_x^{(t)}, hip_y^{(t)})$ position, and the ankle is at $(ankle_x^{(t)}, ankle_y^{(t)})$. The length of the thigh is l_{thigh} and the length of the shins is l_{shin} and the total leg length is l_{leg} Now we can write:

[25] The closest zero pixel is the pixel, which value is zero (i.e. black), and closest to the current pixel from these zero value pixels.

[26] This is true for the knee, but the elbow usually moves a little bit in perpendicular direction to the image plane. This will affect estimation accuracy. But more on this in section IV

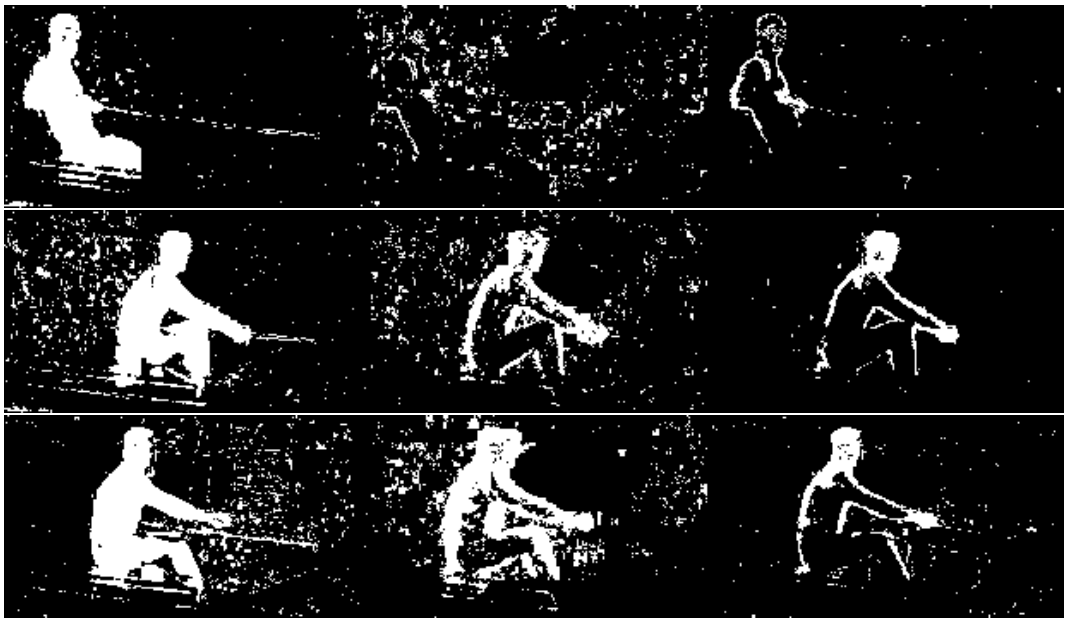

Figure 12: *The results of different background subtraction algorithm on a test video. From left to the right: masks acquired with the area based, GMM and KNN algorithms. Our method always gives fair results, while the others only capture contours or only a part of the contours. The top middle image shows a very bad, unrecognizable result made with GMM. The KNN also achieved poor result in this case, while the area based method performed well.*

$$l_{thigh}^2 = (x_{hip}^{(t)} - x_{knee}^{(t)})^2 + (y_{hip}^{(t)} - y_{knee}^{(t)})^2 \quad (16)$$

$$l_{shin}^2 = (x_{ankle}^{(t)} - x_{knee}^{(t)})^2 + (y_{ankle}^{(t)} - y_{knee}^{(t)})^2 \quad (17)$$

$$l_{leg} = l_{thigh} + l_{shin} \quad (18)$$

From anthropometric measurements we get 0.77 for the l_{shin}/l_{thigh} ratio. At the finish position the legs are straight, so the leg length can be calculated:

$$l_{leg} = (x_{hip}^{(finish)} - x_{ankle}^{(finish)})^2 + (y_{hip}^{(finish)} - y_{ankle}^{(finish)})^2 \quad (19)$$

Now we can solve the linear equation system and get the $(knee_x^{(t)}, knee_y^{(t)})$ location. The elbow location is calculated in a similar manner, using the catch position for calculating the arm length. To estimate the line of the back, the program simply takes the contour of the silhouette from the bottom of the head to the line of the hip. But at finish position the elbow cause trouble to this estimation, because it is behind the back. To solve the issue, a distance transform is applied to the torso, so the line of the spine is acquired. Then this line is shifted left to the position of the back[27]. This is shown on figure 13.

IV. Results

IV. i. Background subtraction

The following section compares the proposed background subtraction algorithm against the KNN and GMM background sub-

[27] But this solution won't work at catch, so based on the phase of motion one of these approaches is used.

Figure 13: *Determining the line of back. If the elbow passes the back, instead of the original estimation (cyan), a new estimation is used (green) with the help of the spine (red).*

tractors described in subsection II. ii. Because there were no videos available containing the annotations for the segmentation task, the section gives only a qualitative accuracy comparison based on manual inspection of some sample videos.

In the case of GMM and KNN for sample size parameter (T and M respectively) 40 frame was set. This relatively low size would achieve the best result[28], because the cyclic nature of rowing motion the foreground appears at a given pixel location always periodically. Because the low sample number, the subsample number was also set to 40 for the KNN method. The c_{thr} decision threshold was leaved on the default value, which seemed to be good. For the k parameter the usual value of 3 neighbor was set. In case of GMM 0.2 was set for the background ratio value (c_f), since the athlete occupies approximately one fifth of the test videos. For the number mixture components (N), we found that values greater than one did not affect significantly the accuracy, so $N = 1$ was used[29]. The shadow detection was switched off, both for KNN and GMM. The other parameters remained default.

Figure 14: *Result of the area based method after noise removal on an example image.*

Figure 12 presents example foreground mask captured with the different methods. The proposed area based method gives fair results with the test video sequences. The only problem is with the area near the shin. On the top left image, the shin is not captured, while on the bottom left image, „two shin" is detected, i. e. the white area below the knee is the shin, as it was approximately one second earlier in time. This due to the design of the algorithm, and it can cause problems also behind the buttocks of the athlete, which can negatively affect the back estimation accuracy. But this is the trade off for fast and in other regions good and robust segmentation. The area based method estimated the foreground fairly well, while GMM and KNN only captured the contours, if at all (on the top middle image, the GMM achieved very poor accuracy). Note, that these images are a bit noisy. In the software implementation a noise removing process is done, as described in section III. Figure 14 presents a mask image after the noise removal process. Based on the qualitative tests, we can say that the proposed method accuracy gets well above the builtin OpenCV algorithms.

But the area based method surpasses GMM and KNN algorithms not only in terms of

[28] At least at the 20 spm stroke rate, which the test video had.

[29] Indeed because the constant background also theoretically it is a good choice.

Table 4: *Average running time of background subtractors on different videos.*

(a) *Running time in milliseconds.*

Method Image resolution	**1** 206x168	**2** 320x240	**3** 212x117	**4** 270x152	**Avg.**
KNN	6.70ms	17.9ms	5.53ms	9.49ms	9.89ms
GMM	1.15ms	3.72ms	1.15ms	1.79ms	1.16ms
Own method	0.12ms	0.29ms	0.13ms	0.18ms	0.18ms
Own method (only subtraction)	0.06ms	0.17ms	0.07ms	0.09ms	0.09ms

(b) *Speedup of different algorithms compared to the KNN.*

Method Image resolution	**1** 206x168	**2** 320x240	**3** 212x117	**4** 270x152	**Avg.**
GMM	5.8	4.8	4.8	5.3	**5.1**
Own method	53	61	42	51	**52**
Own method	106	103	85	101	**99**

accuracy but also in terms of computational efficiency. Table 4a summarizes the average processing time of one video frame using the different methods and various videos. As described above the proposed algorithm after the initial phase can reset the background generation phase, or can use the calculated background image for a more faster segmentation. In the latter case the extra time spent in the background calculation on first few frames will be amortized amongst all video frames. The table shows running time for both case. The software was implemented in *cpp*. To access the source code, see appendix A. We get these results running on *Intel Core i3 CPU*[30] with an embedded GPU[31]. Since OpenCV only supports GPU computations with *NVIDIA* graphic cards [OpenCV, 2016], only the CPU was utilized. The test were running on *Ubuntu Linux* operating system[32], with OpenCV version 3.1.0. As the running time only the time spent with the actual algorithm was counted, for example the decoding from the *mp4* video files is not included.

Table 4b shows the speedup of the methods against the slowest one, the KNN algorithm. It shows the benefits of the proposed area based method very clear. While the GMM achieves approximately 5 times speedup against KNN, the area based method outperforms it with huge approximately 50 times speedup. If the continuous background recalculation is ignored, two order of magnitude speedup can be achieved. This makes the proposed algorithm capable for real-time processing. Potentially a fast mobile based application is feasible. In terms of FPS, which is a widely used metric for measuring performance, the method achieves 390 to 1050 FPS[33] on the test images, presented in table 4a.

IV. ii. Pose estimation

Figure 15 shows pose estimations made on same example videos. The estimated joint positions are marked with different colors on the frames. We conclude that the proposed method works well, even if some of the assumptions made in section III does not hold. In the top three video the background is not clearly distinguishable and also the image quality is very poor, thus the background subtraction leads to a noisy silhouette. At these videos various anomalies can be observed: in the topmost video the estimation of elbow and knee is not

[30] Verison *M 380.*

[31] *Intel Ironlake Mobile* graphic card.

[32] Version Ubuntu 14.04, with kernel version 3.19.0.

[33] At this time accounting the time spent with all the other stuff, like mp4 video stream decoding, etc. .

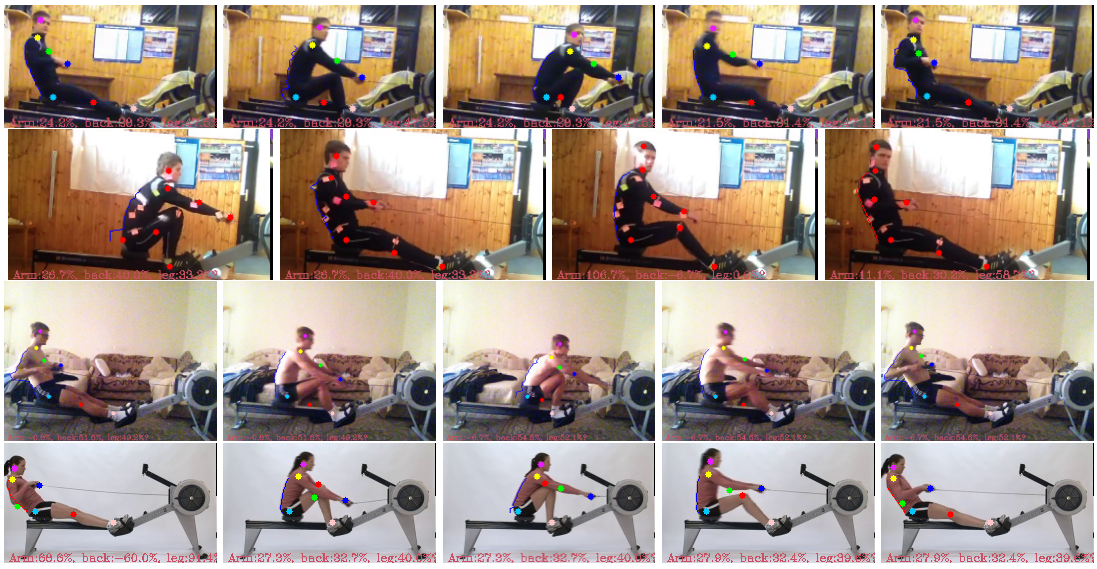

Figure 15: *Results of the pose estimations on different videos. The top three videos does not satisfy all the assumptions made in section III, but the estimation is still reasonably good. In these cases the estimation of the elbows, knees and sometimes back can get wrong. In the bottom video the proposed algorithm produces very good estimation. The head is marked with purple, the shoulder with yellow, the elbow with green, the hand with blue, the hip with cyan, the knee with red and the ankle with pink circle. (In the second row all points are marked with red, because the colorful sticky notes put on the clothing.) The line of the back is marked with blue – in case of silhouette based back estimation type– or red – in case of distance transform based estimation.*

working. This is because the algorithm cannot identify precisely the catch and finish phases, thus the estimation of leg and arm length is inaccurate. The wrong estimation of the elbow also causes trouble when estimating the line of the back (see section III). The errors of shoulder and hand or ankle and thigh estimations are amplified when estimating elbow or knee, because these estimations rely on the previous ones. The line of the back can be also very noisy, like in the second picture of the topmost row, because the low quality image. The videos in the second and third rows also have problems with knee and elbow for the same reason. The leftmost video in the second row also demonstrates an another problem: in some cases, if the athlete *slides on* the catch enough, i.e. bends the legs greatly and the hip get too close to the ankle, the back estimation near the height of the hip won't work. This is because the background subtraction cannot eliminate the legs totally from the created background image, a portion of the shin[34] remains on the background and the calculated mask would mark them as foreground.

Because there was no (indoor) rowing specific image database available, the following method was used to measure prediction accuracy: a small video sequence is collected with using colorful (red and green) markers on the body parts. Figure 16 shows an example video frame. Then a simply *Python* script was able to reliable separate the markers from the video.

[34] Exactly a portion of the shins as they are at the finish.

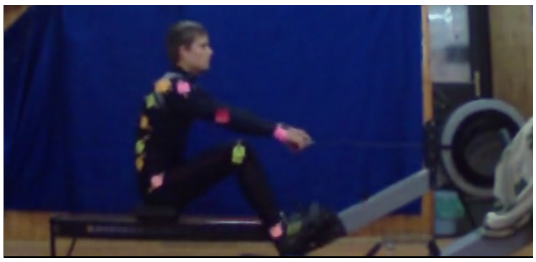

Figure 16: *Example frame with colorful markers (sticky notes).*

The center of the markers are considered as ground truth. Out of the 3360 video frames over all body part types only for 32 case could not estimate the script the positions. These cases were excluded from calculations. We found that these method estimates very reliable the marker's center. Note that because moving of the clothing very small differences can be observed between these „ground truth" positions and the real body part positions. Table 5 shows the *RMSE (root mean squared error)* of the predictions on different body parts. The error is defined as the Euclidean distance of the prediction and ground truth value. RMSE is an illustrative metric in this case. We choose it instead of simple average, because it penalizes greater error with greater weight. Note that the back estimation is not included. The values are presented in pixel values and also in „leg lengths", i.e. compared to the length of the leg[35].

The average RMSE is $6.3px$ or $0.03,,leglength''$. This means, if the leg is $1m$ long in the real world, then the average error is $3cm$. Based on the table, the ankle, elbow and shoulder prediction is the worst. Because the ankles don't move much, so a fixed point could estimate their position better than the actual estimation. The proposed method could be improved, if the user selects the point of ankle[36]. This is not a great request, since it must be done only once per video. The prediction accuracy of elbow relies on the accuracy of hand and shoulder. To improve it, we should try to improve shoulder estimation. But this is a very challenging task. In pose estimation literature the *average precision* and *mean average precision (MAP)* metrics are commonly used by researchers [Agarwal and Triggs, 2004] [Bourdev and Malik, 2009] [Bourdev et al., 2010]. However the proposed method does not allow to set the *recall* value – via a model parameter –, so these metrics couldn't be used.

Table 5: *Accuracy of the pose estimation. Video resolution is* $584x276$.

Body part	RMSQ (px)	RMSQ(leg)
Ankle	8.8	0.04
Knee	5.5	0.03
Hip	6.4	0.03
Hand	5.1	0.03
Elbow	8.0	0.04
Shoulder	7.7	0.04
Head	2.45	0.01
Average	6.3	0.03

V. Discussion

V. i. Possible use cases in daily coaching

Video cameras all widely used by coaches in everyday training among a variety of sport disciplines [Wilson, 2008]. With playing back videos coaches and athletes can notice

[35] Latter values make more sense, since they are independent from image size.

[36] With this improvement, the RMSE for ankle is 6.2 pixel or 0.03 leg length.

several things and technical issues, which is overlooked with the naked eye. Slowing down and stopping the video at certain points is also widely used. But other, more involved analysis is rarely done, either because the lack of software support or the nerve-racking and time consuming task of manual annotations. The proposed framework is able to support rowing coaches in a variety of way. The list possible use-cases is almost endless. This section gives some examples about how can the program help coaches to identify and illustrate technique errors.

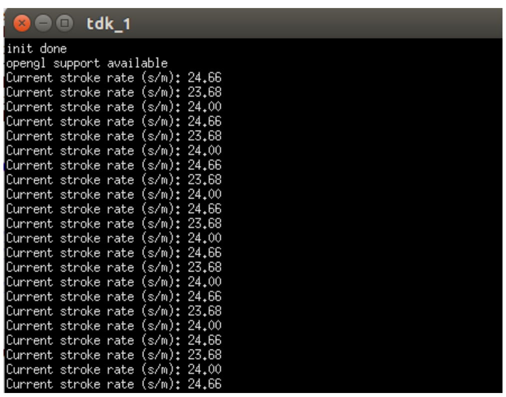

Figure 17: *Measuring stroke rate.*

The most obvious application of the program is to measure stroke rate, i. e. the number of strokes accomplished in one minute. An example stroke rate measurement is presented on figure 17.

As rowing involves the trunk and core muscles, proper back posture is very important in the indoor rowing machine and on water training also. Poor body posture can lead to lower back injuries, which is very common in rowers According to [Bahr et al., 2004] *lower back pain (LBP)* can affect 55% of rowers. Bad body posture can be captured with camera. Figure 18 gives example for body posture analysis. The curve of the back is highlighted on the images. Without bothering factors a coach can easily recognize bad posture. The images also reports the angle of the spine, as defined as the angle between vertical and the hip-shoulder line. The athlete in this case is doing the exercise correctly, the back is almost straight, slightly curved, which can be considered correct. However there could be some problem in the recovery phase with the body part sequencing: before the back swings forward fully, the legs begin to bend, and the hip begins to move. This is considered as technique error by most coaches[37].

Consider the top row in figure 18 for an another example. In this pictures the movement of the hand is tracked and the trajectory is plotted on the screen [38]. Also the actual speed is reported, in units of „leg-lengths". Based on the images a strange technique error appears: before the finish, the athlete pulls upward the handle and the hands. This is obviously a bad movement. In the indoor rowing machine it is only useless, as the rower cannot apply the necessary force to the handle and valuable milliseconds are passing away, while doing this movement. In a rowing boat it can cause even more problems, as the athlete will be harder to pull out the blades from the water.

Figure 19 shows an example of the reported body segment contributions, i. e. how much percentage of the full length is coming from using the arms, the back and the legs. The actual value are also reported below the video. This can be a very valuable information as it can vary a lot among rowers as suggested by [Kleshnev, 2016]. Very extreme values can in-

[37] Note that there is not an accepted an universal „best" technique in rowing, there are more popular rowing styles, which differ in some movements.

[38] This type of visualization can be even more useful in real-time, when the trajectory is continuously evolving.

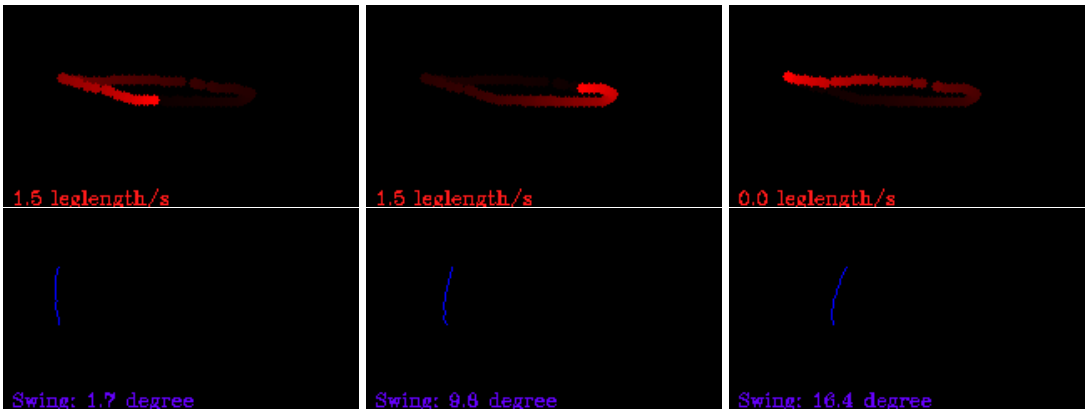

Figure 18: ***On the top:*** *trajectory of the hands. The image illustrates the technique error with great expression power.* ***On the bottom:*** *example images captured about the back posture of a rower. In the bottom of images the angle of spine is reported also.*

dicate technique errors. People with similar height and similar values can be put together to crew boats, because they are doing the same movement. Here the *total stroke length* is defined as the horizontal distance covered by the hands (s_{full}). If we subtract from it the horizontal path of the shoulder ($s_{shoulder}$), we get the length, which the hand traversed compared to the body, i. e. the *arm contribution length* ($s_{arm} = s_{full} - s_{shoudler}$). Similarly value for hip movement (s_{hip}) can be measured and the *leg* and *back contribution* can be calculated respectively ($s_{leg} = s_{hip}$ and $s_{back} = s_{full} - s_{leg} - s_{arm}$). There are even more use-case for our framework, but the goal of this paper is not to give a full survey, so only the above mentioned four application was highlighted.

V. ii. Future work

As mentioned above in section IV, in case of many real life videos the pose estimation is not the best. This is because the proposed framework has a couple of strict assumptions about the input, which do not always hold in real. To get better the results the silhouette based pose estimation might be improved.

But to achieve considerably improvement in we opinion a totally new approach is needed. The large annotated image corpus based *machine learning paradigm*may be a good candidate, there are promising results about general human pose estimation [Bourdev and Malik, 2009] [Bourdev et al., 2010] [Shotton et al., 2013a] and also about some activity specific pose estimation. The problem in this case is the lack of training data, i.e. the lack of indoor rowing training videos annotated with body part positions. To circumvent this problem, it is possible to build a rowing specific image database with automatic annotations using lively color stuff, like sticky notes on the clothing and with very basic image processing extracting they positions. Other possible solution would be using a non-rowing specific image database with joint annotations like the *Leeds Sports Pose Dataset* [Johnson and Everingham, 2010]. *Convolutional neural networks (CNNs)* are used with huge success in a wide variety of image and video processing problems [Goodfellow et al., 2016] [Wang, 2016]. A pos-

sible next step would be to build and train a *CNN* and evaluate its performance.

As the mobile platforms are very popular nowadays, and almost every young athlete and a lot of coaches have one, migrating the presented framework to the popular *Android*, *iOS* and *WindowsPhone* mobile platforms is also a possible improvement. All mobile phones has builtin cameras, the presented algorithm is very fast and Android and iOS supports the OpenCV software library [OpenCV, 2016]. These would all facilitate the migration.

V. iii. Final remarks

In this paper a new framework is presented, which is under the given assumptions capable of extract the head, shoulder, elbow, hand, hip, knee , ankle and back positions of a rowing athlete from video sequences. We proposed a problem specific, very fast and accurate background subtraction method, which surpasses the builtin methods implemented in the widely used OpenCV library. We also presented a pose estimation approach to extract the interested points.

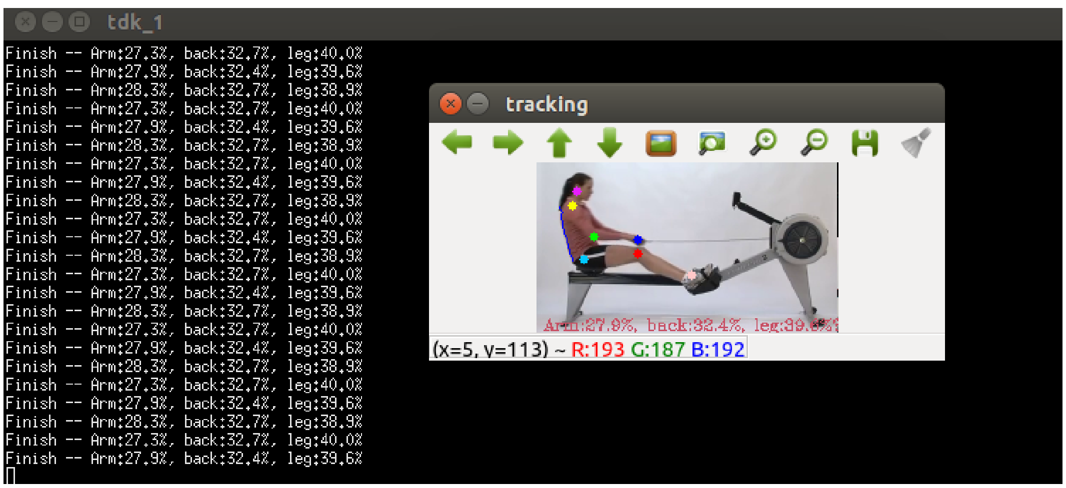

Figure 19: *Contribution of different body parts to the full stroke length at subsequent rowing strokes. The current value also presented on the video.*

References

[Agarwal and Triggs, 2004] Agarwal, A. and Triggs, B. (2004). 3d human pose from silhouettes by relevance vector regression. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–882–II–888 Vol.2.

[Bahr et al., 2004] Bahr, R., Andersen, S., and Loken, S. (2004). Low back pain among endurance athletes with and without specific back loading, a cross-sectional survey of cross-country skiers, rowers, orienteers and nonathletic controls. *Spine*, 29:449–454.

[Baumann, 1989] Baumann, W. (1989). *Grundlagen der Biomechanik*. Verlag Karl Hofman.

[Bourdev et al., 2010] Bourdev, L., Maji, S., Brox, T., and Malik, J. (2010). Detecting people using mutually consistent poselet activations. In *European Conference on Computer Vision (ECCV)*.

[Bourdev and Malik, 2009] Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3d human pose annotations. In *International Conference on Computer Vision (ICCV)*.

[Chris Pulman, 2003] Chris Pulman (2003). The physics of rowing. `www.atm.ox.ac.uk/rowing/physics/rowing.pdf` (visited at 22 Oct. 2015).

[Coleman, 2012] Coleman, B. J. (2012). Identifying the "players" in sports analytics research. *Interfaces*, 42(2):109–118.

[Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, page 603âĂŞ619.

[Cucchiara et al., 2002] Cucchiara, R., Grana, C., Neri, G., Piccardi, M., and Prati, A. (2002). *The Sakbot System for Moving Object Detection and Tracking*, pages 145–157. Springer US, Boston, MA.

[Cucchiara et al., 2003] Cucchiara, R., Grana, C., Piccardi, M., and Prati, A. (2003). Detecting moving objects, ghosts, and shadows in video streams. *IEEE transactions on pattern analysis and machine intelligence*, 25(10):1337–1342.

[Elgammal et al., 2000] Elgammal, A. M., Harwood, D., and Davis, L. S. (2000). Nonparametric model for background subtraction. In *Proceedings of the 6th European Conference on Computer Vision-Part II*, pages 751–767. Springer-Verlag.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

[Jiwaei et al., 2012] Jiwaei, H., Kamber, M., and Pei, J. (2012). *Data Mining: Concepts and Techniques*. The Morgan Kaufmann, 3th edition.

[Johnson and Everingham, 2010] Johnson, S. and Everingham, M. (2010). Clustered pose and nonlinear appearance models for human pose estimation. In *Proceedings of the British Machine Vision Conference*. doi:10.5244/C.24.12.

[Kleshnev, 2016] Kleshnev, V. (2016). *The biomechanics of rowing*. The crowood press.

[Leutenegger et al., 2011] Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). Brisk:

Binary robust invariant scalable keypoints. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2548–2555, Washington, DC, USA. IEEE Computer Society.

[Microsoft, 2011] Microsoft, C. (2011). Kinect for windows software development kit online documentation. http://research.microsoft.com/ en-us/um/ redmond/ projects/ kinectsdk/guides.aspx. Accessed: 10th October 2016.

[NASA, 1978] NASA (1978). *Anthropometric Source Book*, volume 2. Springfield VA, Johnson Space Center, Houston, TX.

[Nise, 2011] Nise, N. S. (2011). *Control Systems Engineering*. Wiley, 6th edition.

[Nummiaro et al., 2003] Nummiaro, K., Koller-Meier, E., and Gool, L. V. (2003). An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99âĂŞ110.

[OpenCV, 2016] OpenCV, C. (2016). Opencv 3.1.0 online documentation. http://docs.opencv.org/3.1.0/. Accessed: 7th October 2016.

[Ortiz, 2012] Ortiz, R. (2012). Freak: Fast retina keypoint. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 510–517, Washington, DC, USA. IEEE Computer Society.

[Perez-Sala et al., 2014] Perez-Sala, X., Escalera, S., Angulo, C., and Gonzalez, J. (2014). A survey on model based approaches for 2d and 3d visual human pose recovery. *Sensors*, 14(3):4189–4210.

[Press et al., 2007] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (2007). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition.

[Shotton et al., 2013a] Shotton, J., Girshick, R., Fitzgibbon, A., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., and Blake, A. (2013a). *Efficient Human Pose Estimation from Single Depth Images*, pages 175–192. Springer London, London.

[Shotton et al., 2013b] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. (2013b). Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124.

[Szeliski, 2011] Szeliski, R. (2011). *Computer Vision: Algorithms and Applications*. Springer, 2011th edition.

[Thomas W. Miller, 2015] Thomas W. Miller (2015). *Sports Analytics and Data Science: Winning the Game with Methods and Models*. FT Press Analytics.

[Valery Kleshnev, 2006] Valery Kleshnev (2006). Rowing biomechanics. `http:\www.biorow.com` (visited at 12th Nov. 2015).

[Wang, 2016] Wang, Y. (2016). Pose-rcnn: Joint object detection and pose estimation.

[Wilson, 2008] Wilson, B. D. (2008). Development in video technology for coaching. *Sports Technology*, 1(1):34–40.

[Witte and Witte, 2010] Witte, R. S. and Witte, J. S. (2010). *Statistics*. Wiley, 9 edition.

[Zivkovic, 2004] Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the Pattern Recognition, 17th International Conference*

on (ICPR'04) Volume 2 - Volume 02, ICPR '04, pages 28–31, Washington, DC, USA. IEEE Computer Society.

[Zivkovic and van der Heijden, 2006]
Zivkovic, Z. and van der Heijden, F. (2006). Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780.

Appendices

A. Accessing the implementation

The software was implemented in CPP. The source code is available in the git repository at the following URL: `https://bitbucket.org/bencetamas/tdk_1/`.

To run the software, you need the *Make*, *CMake*[39] and *GNU c++ compiler*[40] programs being installed. A properly installed version of OpenCV 3[41] software library is also required. Please note, that installing OpenCV could very challenging on some operating systems. The code should run on Windows, Mac and Linux operating systems also[42].

After cloning or downloading the git repository, open a terminal or command prompt, go to the downloading location and type the following commands:

```
cmake .
make
```

This should produce an executable in the root project directory called *main*. Type

```
./main
```

to run the program. Note that in the current version[43] the name of the processed video file is hard-wired to the code. To open another video file, edit the name in the *main.cpp* source file and recompile the program. The program can support various video file formats, but the list supported video formats depends on the settings of current OpenCV installation.

The syntax of program execution may change in the future, please always read the *README.md* for up-to date instructions.

[39] At least version 2.8.
[40] At least version 4.4.
[41] We tested with the version 3.1.0.
[42] The code was tested on Ubuntu 14.04.
[43] As of 21th October, 2016