

## **AVIImage – video motion analysis software for tests of biomechanical movement characteristics**

ELŻBIETA ROSTKOWSKA<sup>\*\*\*</sup>, PRZEMYSŁAW BENZ<sup>\*</sup>, LECHOSŁAW B. DWORAK<sup>\*\*\*\*</sup>

<sup>\*</sup> University School of Physical Education in Poznań

<sup>\*\*</sup> University School of Physical Education in Warszawa – Biała Podlaska

<sup>\*\*\*</sup> Academy of Fine Arts in Poznań

The aim of this paper is to introduce our AVIImage software for the analysis of movement technique recorded by means of video registration with RGB 24 bits depth. AVIImage software allows manual or automatic recognition of markers characterising the object examined, analysis of basic parameters in 2D or 3D of kinematic structure of movement.

*Key words: biomechanics of movement techniques, software, video registration*

### **1. Introduction**

Professional software for the analysis of movement technique offered by companies such as Ariel, Peak Performance, Simi, Vicon is extremely expensive, hence unobtainable for many research teams engaged in movement analysis [2], [10], [11]. This situation made us inclined to produce our own software aimed at solving exact research projects. Works of ERDMANN et al. [3], [4] and KUZORA [6] are the examples which relate to the analysis of gait locomotion. On the other hand, BENZ [1] proposed the software that calculates basic kinematic characteristics relating in general to swimming techniques.

This paper constitutes both documentation and help for AVIImage, the AVI capture and processing utility. Video recording of sports makes good use of high-speed shutter. Many sportsmen have been using video for a number of years to analyse their style and technique when practising a certain sport. Golfers, in particular, are known to analyse their golf swings with the help of camcorders. The coach tapes the players as they practice, then a debriefing session is held afterwards where the players can

watch their golf swings frame by frame. Likewise, the same techniques are employed by tennis players to check that their serves are being performed as well as they think they are. As everybody knows, it is difficult to know exactly what you look like when you are practising a sport, and using a video camera to monitor your progress is a great form of training. You can see at a glance whether your body is balanced and adopts the right position, and also whether you look awkward or relaxed.

## **2. Program purpose**

The program allows one to analyse movements in video sequence. You can analyse video in two ways: manually, by mouse clicking, or automatically, by auto-recognizing markers on the object examined. During manual program operation you may watch the marked position in the status bar. If you make a mistake when marking the object, you have to press “CTRL+Z” to go back to the previous position to mark this point again. Before auto-analysing you have to define initial positions one, two or three to remember colour, shape and luminance. These parameters will be used for finding marker position in next frames.

After marking objects you may look at preliminary charts to check the general correctness of position. This is useful in long movie sequences where you can make some mistakes while point marking. The program consists of a number of tools to analyse the object metrics found, such as “calculate real angle”, “distance between points” or “change units from pixels to metric” or “searching the splitting point of body line and the object’s velocity”.

## **3. Technical aspects**

The program can work on two movies simultaneously. This is important if you want to analyse an object in three dimensions. Additionally you may study the same movement of two persons so that you can compare their parameters. Both movies must be recorded with the same settings as a full uncompressed frame with RGB 24 bits depth. Input movies have 25 Hz frame rate which means that they are 25 frames per second. To achieve 50 Hz each frame is divided into two half-images. The second image is shifted in time in relation to the first frame. When you record a movie without compression you always get the sequence of pictures with half-frames. The disadvantage of this solution is more serious in vertical resolution and the movie occupies more space on the hard disk. For example, two-second long sequence requires about forty Mbytes of free disk space. If one uses a coder with prediction to capture video, one cannot recover original frames and moreover one cannot divide them into half-frames.

#### 4. Creating markers

Before building markers, we would like to know how the light appears on a movie recorded in real-life conditions. Our first experience in using test video sequences provided was measuring the average level of each colour component. Here there are three sample pictures and the corresponding values of average Red, Green and Blue components (figure 1).




Picture	<i>R, G, B</i> component
	<i>R</i> = 161 <i>G</i> = 160 <i>B</i> = 150
	<i>R</i> = 138 <i>G</i> = 223 <i>B</i> = 212
	<i>R</i> = 167 <i>G</i> = 161 <i>B</i> = 113

Fig. 1. Sample pictures and corresponding values of the colour components

Grey colour dominates in the first picture which means that in this picture each component has a similar value. After they are evaluated one can see that it is true. In the second picture, which was taken underwater, the Green and the Blue components clearly dominate. This means that one should use Red colour to build a marker. The Red component is about twice as low in value as the Green and the Blue ones and when one uses Red light he can easily differentiate the marker from the background. In this way, one should choose the Blue marker in the third picture because this component has the lowest value. This is more difficult when analysing three markers simultaneously. If one wants to analyse the objects whose paths never cross he may use the same colour markers, because the program always selects the nearest position of the marker in the next frame. In future, the authors want to apply an intelligent

algorithm which will calculate the most likely position in the next step. The marker will first be sought in this calculated position. This algorithm reduces the ambiguity of the situation where the program finds two markers within the same distance.

## 5. Autodetection algorithm

There are two autodetecting algorithms in the AVIImage program. Their role is to find markers in a movie. Both algorithms are managed by a few parameters. The parameters are configured in the options window, so an advanced user may manipulate them. Before launching autodetect the user has to check markers in the first analysed frame of the video. During video analysis a lot of factors have to be taken into consideration, for example, the velocity or trajectory of movement, the colour of the markers, the background level of luminance and others. The objects observed may sometimes disappear behind other objects in motion. Such a situation occurs when sportsmen pass the hands behind their trunks. The user may choose the algorithm, depending on the properties of the video sequence analysed.

The first algorithm finds the segments whose colour is similar to that of the marker selected at the beginning of the movie. An average colour is estimated on the basis of the first marker for reference of future comparisons. The difference in the markers between the first and subsequent frames is calculated by Euclid's distance for each colour channel:

$$\Delta = \sqrt{(R_a - R_b)^2 + (G_a - G_b)^2 + (B_a - B_b)^2},$$

where:

- $R, G, B$  – Red, Green, Blue colour channel,
- index  $a$  – an average colour stored in program memory,
- index  $b$  – the colour of currently measured point.

The segments are created by applying similar colour to consecutive points. The marker selected by the user is the first segment. In the next frame, the algorithm looks for a similar segment only in a restricted area. Size of this area depends on the program setting and the calculated velocity and path of movement of markers (in the next frames). By doing this the program eliminates excess segments beyond a specific range and seriously shortens the time needed to calculate positions. In order to determine the position of the marker from several segments found, the program has to reject the worse results. First the segments to be rejected are those that are too large or too small in size and second the segments whose positions are too far from the path estimated.

The second algorithm uses a mask for detecting the marker position. The mask stores the marker shape and colour in two-dimensional array along with an

appropriate size. The mask should always be larger than the marker to make including all points of this one possible. The mask is created when the user selects a marker in the first frame of the video. The algorithm shifts this mask around the estimated marker position and calculates the resemblance between the stored factors and the currently detected shape. This shape should have the same size as the mask. In order to calculate this resemblance, Euclid's formula of distance for each colour channel between respective points in the mask and the currently detected area is used.

**Motion trajectory** describes the displacement of objects with time, an object being defined as any spatiotemporal region. The successive spatiotemporal positions of the object are represented by the position of one representative point of the object, such as the centre of mass. Such a simplicity is relevant, as humans perceive motion of objects at high level. The trajectory model is the first- or the second-order piecewise approximation of the spatial position of the representative point along time, for each spatial dimension:

- first-order approximation:

$$X(t) = x_i + V_i(t - t_i) \quad (\text{equation: } YX)$$

with:

$$V_i = \frac{x_{i+1} - x_i}{t_{i+1} - t_i},$$

- the second-order approximation:

$$X(t) = x_i + V_i(t - t_i) + \frac{1}{2}a_i(t - t_i)^2 \quad (\text{equation: } YY)$$

with:

$$V_i = \frac{x_{i+1} - x_i}{t_{i+1} - t_i} - \frac{1}{2}a_i(t_{i+1} - t_i),$$

and similarly in the case of other dimensions  $y$  and  $z$ . In a such model,  $V(i)$  and  $a(i)$  do correspond to the object's speed and acceleration, respectively, considered constants on  $t_i, t_{i+1}, x_i$  and  $x_{i+1}$  are the positions at the time  $t_i$  and  $t_{i+1}$ . On the basis of this model, the core of the description is a set key of key points, representing the successive spatiotemporal positions of the object described, between which trajectory is interpolated. Key points are defined by their coordinates in space (2D or 3D) and time:  $(x, y, t)$  or  $(x, y, z, t)$ . By default, linear interpolation is used (equation  $YX$ ). When non-linear interpolation is done between two key points in one spatial dimension, the corresponding interpolating parameter  $a(i)$  is added to specify the second-order function of time that should be used (equation  $YY$ ). Figure 2 illustrates the trajectory

representation. It shows a modelled trajectory, instantiated by the parameters in bold ( $x_i, t_i, a_0$ ).

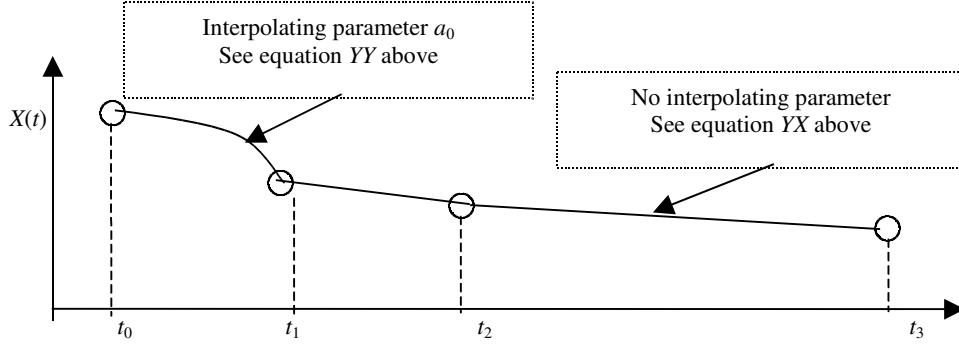


Fig. 2. Example of trajectory (one dimension)

Both algorithms use the linear interpolation while tracing object. Here, current object's position is estimated from an appropriate number of previous locations. The program calculates next location by analysing motion line based on linear regression theorem, where a line is represented by the following equation:

$$y = Ax + B.$$

The best approximation for coefficients  $A$  and  $B$  are:

$$A = \left[ n \left( \sum_{i=1}^n x_i y_i \right) - \left( \sum_{i=1}^n x_i \right) \left( \sum_{i=1}^n y_i \right) \right] \cdot \frac{1}{\Gamma},$$

$$B = \left[ \left( \sum_{i=1}^n x_i^2 \right) \left( \sum_{i=1}^n y_i \right) - \left( \sum_{i=1}^n x_i \right) \left( \sum_{i=1}^n x_i y_i \right) \right] \cdot \frac{1}{\Gamma},$$

where:

$$\Gamma = n \left( \sum_{i=1}^n x_i^2 \right) - \left( \sum_{i=1}^n x_i \right)^2.$$

In the above equations,  $X$  and  $Y$  are the positions in the first and the second dimensions, respectively, and  $n$  is the number of points. Calculating position provides one with certain advantages. The first lies in the fact that the algorithm always starts looking for a marker at an estimated position so the program works faster. The second advantage is a low complexity during autodetection which is important because this algorithm replaces manual checking in order to make working with the program comfortable and efficient. In a previous program version, an average velocity and the

line along the path of object motion were calculated to estimate the next object position. This algorithm did not work as correctly with bending motion as the one shown in figure 3.

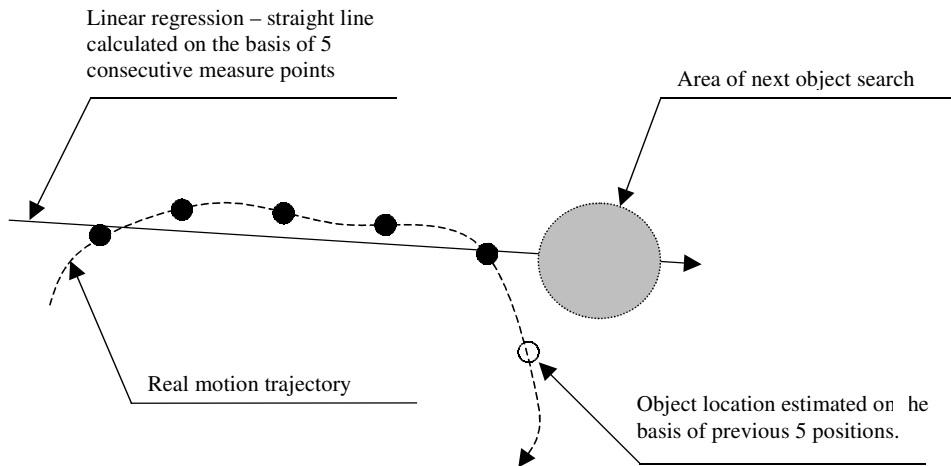


Fig. 3. Estimation of object location

## 6. Working with the program

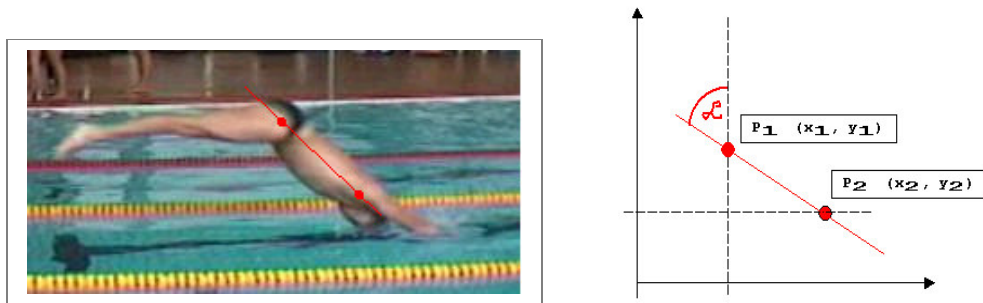


Fig. 4. Illustration of body angle calculation in swimming

Calculate the angle between vertical line and body.

$$\tan(\alpha) = \frac{x}{y} \Rightarrow \arctan(\alpha) = \frac{x}{y}$$

One may also measure the distance between the points  $P1$  and  $P2$  from the Pythagoras theorem:

$$h = \sqrt{x^2 + y^2} .$$

### 6.1. Three-point analysis

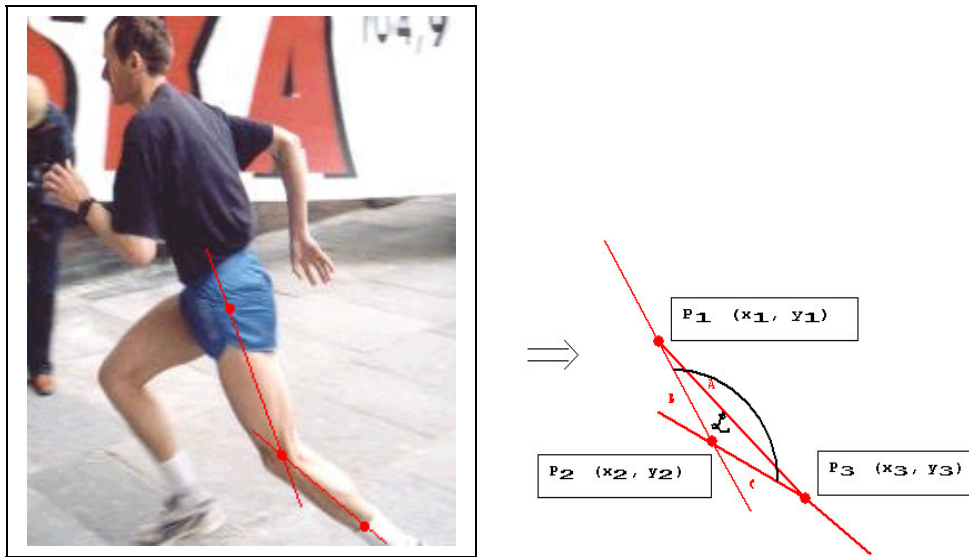


Fig. 5. Calculation of angle of knee joint

The distance between the points:

$$A = \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} ,$$

$$B = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} ,$$

$$C = \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2} .$$

The angle between the sides  $A$  and  $B$  from cosine theorem:

$$A^2 = B^2 + C^2 - 2BC \cos(\alpha)$$

after transformation our formula can be defined as follows:



$$\alpha = \arccos\left(\frac{B^2 + C^2 - A^2}{2BC}\right).$$

Check how far the swimmer moves his palm across the body line.

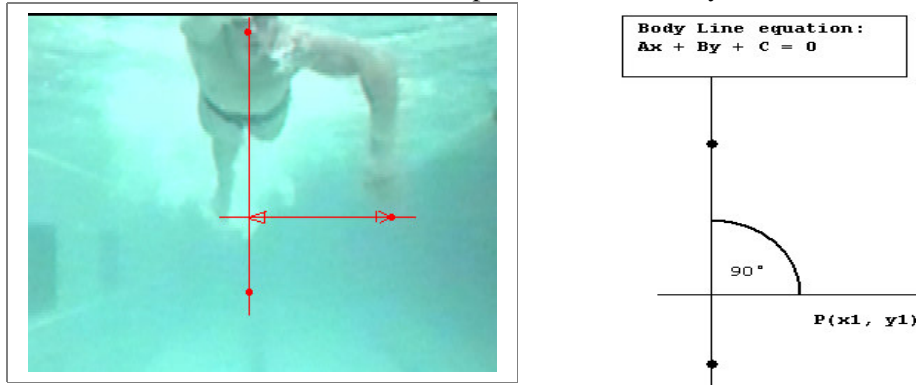


Fig. 6. Illustration of the sequence of swimmer's movements

So straight line perpendicular to the body line is described by the following equation:

$$Ax - By + C' = 0,$$

and the distance between the body line and the palm is equal to:

$$h = \frac{|Ax_1 + By_1 + C|}{\sqrt{A^2 + B^2}}.$$

## 6.2. Three-dimensional analysis

One of the advantages of the program is the possibility of carrying out a three-dimensional analysis with two cameras. During this program operation one can come across a few traps which should be taken into consideration both by the user and the designer of this program. We deal with the following traps:

- the properties of the camera such as the focal length,
- the distance between the object analysed and each camera,
- the velocity of the object movement,
- the object location in relation to the centre of point reference.

Figure 7 presents typical 3D situations. In this example, the authors try to measure the angle of the elbow bend, which is impossible with only one camera, because this camera should be always perpendicular to the bent elbow. To estimate this angle we

should set the first camera in the front of a swimmer and the second one on his left or right side. In a perfect situation, both cameras should move at the same velocity as a swimmer.

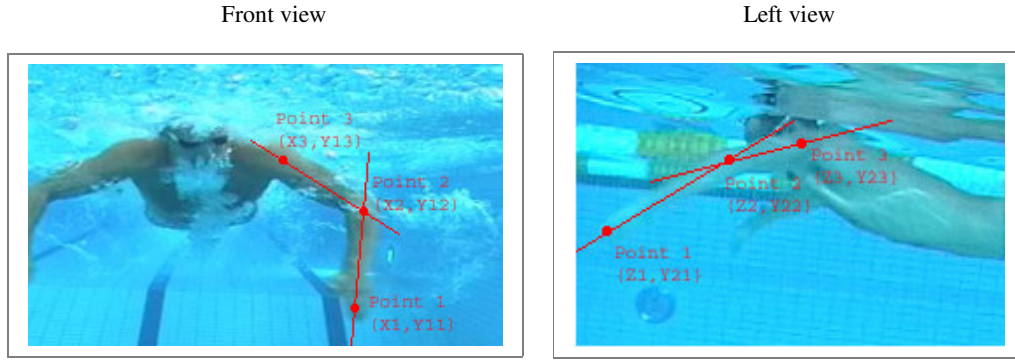


Fig. 7. Planes of filming

If both cameras are perpendicular to each other, we may use mathematical formulas such as the ones given further. Each camera gives 2 components for each of the 3 points. For example:

- $Y_{12}$  means the component  $Y$  in the second point in the first camera that is situated in the front of a swimmer,
- $Z_1$  means the component  $Z$  in the first point in the second camera that is situated on the left side of a swimmer.

Because it is impossible to get exactly the same size of the object in both views, an algorithm should compensate for this difference. It is possible to make use of the components  $Y$  in order to compare views and in this way to compensate for the component  $Z$  in the second view. This component  $Z$  depends on the rate of difference which is shown in this equation:

$$z_k = \frac{\max(Y_{11}, Y_{12}, Y_{13}) - \min(Y_{11}, Y_{12}, Y_{13})}{\max(Y_{21}, Y_{22}, Y_{23}) - \min(Y_{21}, Y_{22}, Y_{23})} \cdot Z_k^*$$

3D distance between points:

$$D_{13} = \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2 + (z_1 - z_3)^2},$$


$$D_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2},$$

$$D_{23} = \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2 + (z_2 - z_3)^2} .$$

Final angle may be calculated from this formula:

$$\alpha = \arccos \left( \frac{D_{13}^2 + D_{12}^2 - D_{23}^2}{2D_{12}D_{23}} \right) .$$

## 7. Access to result in output Excel file

To save position data and calculated values click on the brown disk icon in the toolbar “” and select a path and a file name. This file can be opened later on with Excel or notepad program for further analysis.

Structure of output file

Column number	Contents of the column
1	Number of frames in a movie
2	Number of points that describe a position
3	If this field is 0 then we have even half frame Else it's odd
4, 5	Position of the point in view
6	Angle in the first point and between the second and the third points
7	If we have a line through points one and two then the value in this column is the distance between the third point and the line
8, 9, 10	Distance between points

## 8. Implementation

The program is based on SDI application created in Visual C++, a member of the Visual Studio 6.0 family of development products [7]. The language used is object-oriented which means that the programming structure encapsulates both data and functionality (figure 8) that are defined and allocated as a single unit and to which the only public access is through the programming structure's interfaces. The AVIImage class represents the main program functionality such as:

- capturing and decoding video frames,
- drawing frames in the view,
- calculating output factors such as distance, angle, etc.,
- program security and DEMO version tools.

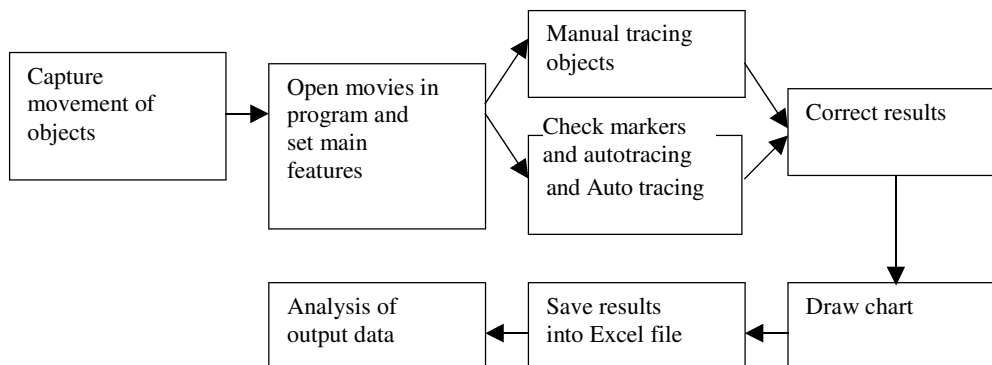


Fig. 8. Block diagram of the main program functionality

### Acknowledgement

The research project has been financed by a grant for the statute activity of the Department of Swimming and Water Rescue and the Chair of Biomechanics.

### References

- [1] BENZ P., *Automatic analysis of sportsman motion*, Poznań University of Technology, master's thesis, 2003, p. 42.
  - [2] DWORAK L.B., *Niektóre metody badawcze biomechaniki i ich zastosowania w sporcie, medycynie i ergonomii*, Skrypt nr 91, AWF w Poznaniu, 1991.
  - [3] ERDMANN W.S., DARGIEWICZ R., CIEŚLA P., KUZORA P., PRĘTKIEWICZ-ABACJEW E., *Investigations of walking by videocomputerized method*, Materiały XIII Szkoły Biomechaniki, Poznań, 1996, s. 131–135.
  - [4] ERDMANN W.S., KUZORA P., PRĘTKIEWICZ-ABACJEW E., *System videokomputerowy badania chodu dzieci*, *Mechanika w Medycynie*, 1998, 4, s. 89–93.
  - [5] JAHNE B., HAUSECKER H., *Computer Vision and Applications*, Academic Press, 2000.
  - [6] KUZORA P., *Automatyczne wyznaczanie charakterystyk chodu*, master's thesis, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska, 1995.
  - [7] LEINECKER R.C., *Visual C++5.0, Power Toolkit*, MIKOM, 1998.
  - [8] MSDN, *Library Help for Visual Studio 6.0*, 1995–1999, Microsoft Corporation, 2000.
  - [9] SALEMBIER P., SIKORA T., *Introduction to MGEG-7 Multimedia Content.*, Description Interface, John Wiley and Sons Ltd., 2002.
- [1] Simi, *SIMI Reality Motion System: 2D and 3D motion analysis*, SIMI Catalogue, 2002.

- [2] Vicon, *Vicon Motion System*, Catalogues, 2003.
- [3] [www.simi.com](http://www.simi.com).
- [4] [www.swim.ee](http://www.swim.ee).
- [5] [www.vicon.com](http://www.vicon.com).