A Robust Preprocessing Algorithm for Low-Resolution Soccer Videos

Martin F. Wrulich, Luca Superiori, Olivia Nemethova and Markus Rupp
Vienna University of Technology
Institute of Communications and RF Engineering
Gusshausstraße 25/389,1040 Vienna, Austria
mwrulich@nt.tuwien.ac.at, lsuper@nt.tuwien.ac.at, onemeth@nt.tuwien.ac.at,
mrupp@nt.tuwien.ac.at

ABSTRACT

Multimedia transmissions, especially of sport events, are an important share of the services offered by mobile network operators. Due to limited capacity, lossy compression codecs have to be used, inducing a significant quality degradation. The quality impairments can particularly be observed in soccer games, when the ball is blurred or even totally disappears on low-resolution user terminals. We present a suitable preprocessing of the original video by means of ball sharpening or enlarging in the original video sequence to avoid problems caused by resolution down-sampling and compression. The proposed system includes a novel initial ball-recognition algorithm and an improved ball-trajectory tracking. Our proposed algorithm runs robustly and is computational very economical.

Categories and Subject Descriptors

I.4 [**Image Processing and Computer Vision**]: Applications

General Terms

Algorithms, Experimentation

Keywords

Visual Content Analysis, Image Processing, Ball Tracking, Recognition, Segmentation

1. INTRODUCTION

Nowadays, soccer games represent very popular contents not only for analog and digital television, but also for streaming over mobile networks. Typical mobile terminals usually offer resolutions as small as 144×176 (QCIF). The Universal Mobile Telecommunication System (UMTS)[1] supports data rates up to 2 Mbit/s, shared by all users in a cell. Therefore, for unicast transmission of streaming video, data rates about up to 128 kbit/s are feasible. Video codecs supported by UMTS are currently H.263 and MPEG-4, with their basic profiles, H.264 is in discussion (see [1]). Lossy compression used by these codecs leads to visual quality degradation: frame reduction causes an overall jerkiness of the video; further compression achieved by the coarser quantization results in loss of spatial details accompanied with blockiness and blurriness.

A soccer match usually encompasses scenes with diverse character, separated by scene changes (i.e. cut, wipe, zoom in/out, transition, etc.). Most common are wide-angle moving camera shots. These are particularly critical for the compression, since the ball as well as the players are represented by a few pixels only, and thus very susceptible to any quality degradation. The ball in a soccer game carries the most important information; watching a game with a small ball blurred into the field becomes rather annoying. In some cases the ball even disappears from the image due to the compression or due to picture resolution down-sampling.

Figure 1 provides a schematic illustration of the problem: the ball in the already downsampled input sequence is represented by several pixels of not necessarily uniform color. The appearance of the ball differs even within the same video sequence from frame to frame. After video compression (performed before the transmission) the ball visibility worsens considerably. To improve the ball appearance in such cases, we already proposed a method [2] that searches the ball in each frame and replaces it by its enlarged or just sharpened version. The so replaced ball is then likely to be visible after the compression and promises to considerably enhance the user perceptual quality. The main advantage of such a solution is its wide applicability: it does not require any changes of the video codec or network protocol standards and the method can run at the streaming server of the mobile operator or the content provider.

In this paper, complementary to the findings in [2], we present a new initial ball-recognition algorithm, which is able to reliably detect the ball at the beginning of a scene without supervision. Furthermore, we introduce an MMSE (minimum mean square error) based ball-tracking to predict a suitable region of interest for the on-the-fly ball detection together with a simple but effective template update. Our proposed techniques are focused on affordable complexity,

[1]European standard of the 3rd generation of mobile systems, standardized by 3rd Generation Partnership Project (3GPP)

MV'07, September 28, 2007, Augsburg, Bavaria, Germany.

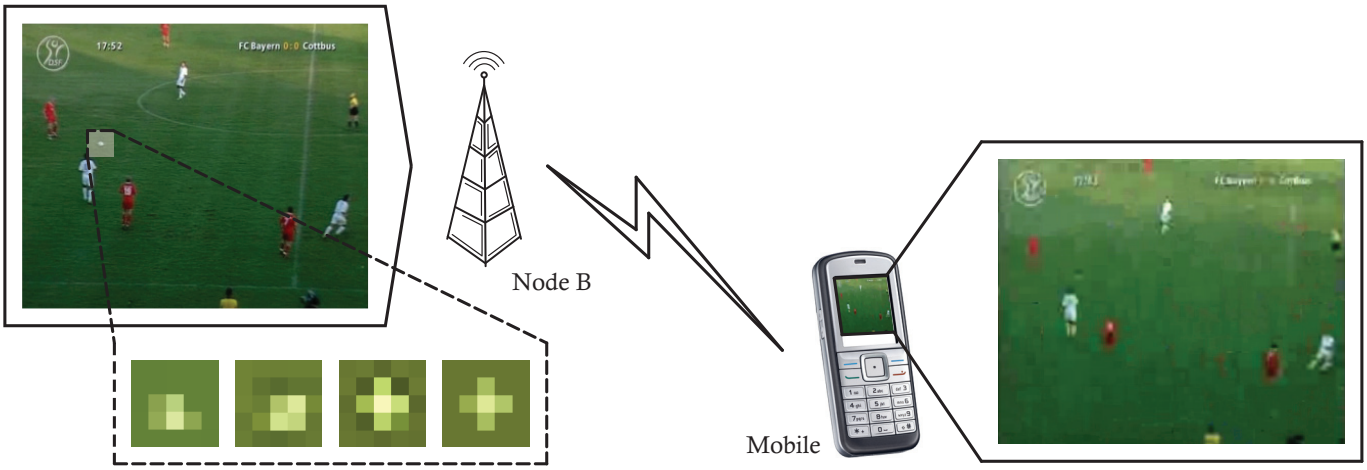


Figure 1: Depiction of the quality degradation as observed in wireless multimedia communication. The original video shows all relevant information (i.e. the ball), whereas after suitable compression for the capacity-limited wireless channel and subsequent transmission, the output video at the user-equipment is suffering from a significant quality reduction, i.e. the ball is nearly invisible. For the uncompressed original frame, a set of sample ball-appearances in case of QCIF resolution are shown.

allowing for a real-time implementation, in contrast to previous work (see e.g. [3–6]).

This paper is organised as follows. In Section 2 an overview of the ball appearance improvement system, together with its logical functions, is described. Some basic image processing techniques of our algorithm are recapitulated in Section 3. Afterwards, Section 4 introduces our novel initial ball recognition algorithm and Section 5 describes the on-the-fly search with details about trajectory tracking and template learning. In Section 6, we propose a low complexity method to generate a possible replacement ball. Experimental results can be found in Section 7, after which a conclusion finalises the paper.

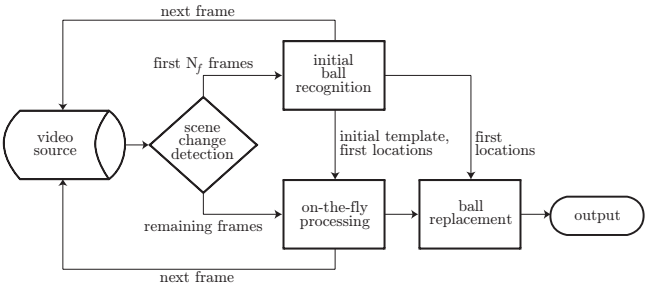


Figure 2: Overview of the basic algorithm interactions between the four separated main parts.

2. SYSTEM OVERVIEW

Our algorithm consists of four main parts: the *initial ball recognition* and the *on-the-fly search*, which are triggered by the *scene change detection*, as well as the *ball replacement*, which permits the desirable appearance improvement after the compression and completes the proposed algorithm. A schematic overview of the interaction between the four algorithm parts can be seen in Fig. 2. The particular parts: scene change detection, initial ball recognition and on-the-fly processing each uses a consequent image preprocessing to enhance their reliability (not depicted in Fig. 2).

The principle, the following steps are performed in our system: a video source provides a series of frames which are monitored by a scene change detector. This scene change detector chooses whether the initial ball recognition or the on-the-fly processing is performed. In case of a detected scene change, or at the beginning of the video, the frames are passed to the initial ball recognition. After its successful completion, the subsequent frames are then passed to the on-the-fly processing until the next scene change occurs.

To be more specific, the purpose of the initial ball recognition is to find an appropriate template for the ball and determine the first W ball-positions. These two tasks are mainly solved using template matching, as well as by forming so-called *minimum distance polygons*. The on-the-fly processing, in contrast, uses a consequent tracking of the ball-trajectory to predict the ball position in the present frame. Then, at the predicted position, an appropriate region of interest (ROI) $\mathcal{A}$, can be specified, where again template matching is performed to recognise the ball.

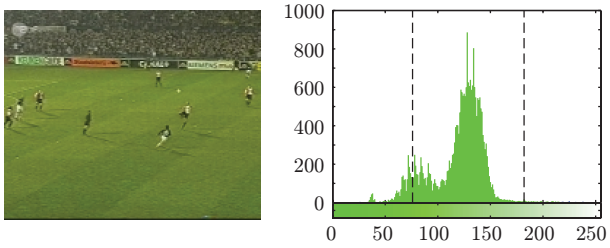
3. VIDEO PREPROCESSING

Within our system, we use some well-known image preprocessing techniques (cf. [7,8]), which we now briefly recapitulate in order to clarify our terminology.

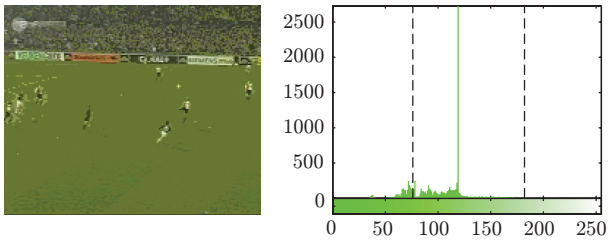
3.1 Image Preprocessing Techniques

3.1.1 Dominant Color Detection and Replacement

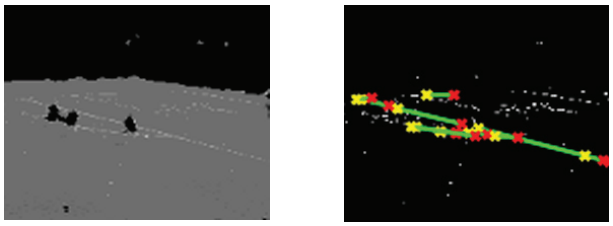
In order to enhance the reliability of the template matching (see Section 3.2), we found removing the variety of colors of the field to be a promising approach. Accordingly, a color histogram analysis is used (see [3, 9]) to specify the color range of the dominant color (the range of "greens" in the field). The resulting ranges are specified by thresholds, defined relative to the maximum value of the histogram in each RGB color channel. We empirically set these thresholds to



(a) A sample frame from a soccer video in QCIF resolution together with the histogram of the green color channel. The color boundaries to identify the dominant color range are marked by vertical dashed lines.



(b) The same frame as above with replaced field pixels and the resulting histogram of the green color channel.



(c) Binary map of the field and the result of the line detection for a wide angle shot.

Figure 3: Different examples for the image processing performed to increase the reliability of the proposed system.

25% in the red, 28% in the green and 32% in the blue color channel. Fig. 3(a) shows an example of the obtained color boundaries in the green color channel. Assuming that the frame shows mainly the field, the pixels with a color lying inside all color channel boundaries are replaced by pixels with a single color. Accordingly, most of the color variety of the field is eliminated. The uniform replacement color is found by a simple weighted average of the histograms of each color channel. Fig. 3(b) shows the result for the previous example. It has to be noted that the remaining value outside the color boundaries are caused by the pixels, where at least one color channel shows a value outside the specified boundary.

3.1.2 Line Detection

Since the ball and the vertical field-lines are of comparable size, the ball does not appear visible, if it lies on a vertical field-line. During the preprocessing, possible pixel lying on a field-line are detected and their color is replaced by the dominant one. Therefore, if the ball is overlapped by a line, a missed detection is generated. We found the Hough transformation [7] to deliver best results. The search for the lines takes place on a thresholded grayscale representation of the frame, where only the field (extracted by smoothing and dominant color detection) is considered. Fig. 3(c) shows an example for the grayscale representation of the field, and the recognised field lines.

3.2 Template Matching

One of our main goals is to locate the ball within each frame in quasi real-time and accordingly with low computational complexity. For this task we found similarity-measure based template-matching techniques to be very suitable, because they showed a sufficient reliability and a very low computational burden for the complete algorithm. We decided in favour of the sum of absolute differences (SAD) because of its sufficiently good performance in combination with the dominant color detection and replacement. The SAD value at a given point (x, y) in frame number f is defined by

$$\mathrm{SAD}(f, x, y) \triangleq \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \left| \mathbf{F}_f(x+i, y+j) - \mathbf{T}(i, j) \right|, \qquad (1)$$

where $\mathbf{F}_f$ denotes the fth frame of the video sequence; x, y are the SAD coordinates (within the searched region) and i, j are the coordinates within the $N_x \times N_y$ template $\mathbf{T}$. The most probable candidate position for the ball, $\mathbf{c}(f)$, can than be determined as the position (x, y), at which the SAD metric (1) yields the lowest value,

$$\mathbf{c}(f) = \arg \min_{x,y \in \mathcal{A}} \mathrm{SAD}(f, x, y), \qquad (2)$$

where $\mathcal{A}$ denotes the "search region" within the frame. Furthermore, the minimum SAD value, $\min_{x,y \in \mathcal{A}} \mathrm{SAD}(f, x, y)$, is used as a measure of the reliability of the ball-recognition.

3.3 Scene Change Detection

To facilitate the real-time operation of the algorithm, the scene change detection has to be performed using previous pictures only. There are several candidate methods suitable to accomplish this task [9]. We decided in favour of a technique based on the sum of pixel-wise differences [7]. The performance of a scene change detector based on such a difference metric depends essentially on the threshold. It is nearly impossible to find a fixed threshold that would suit well all the soccer videos, or even all the different scenes of one match. Therefore, we implemented a dynamic threshold (see [10]), using the instantaneous statistical properties of the sequence.

4. INITIAL BALL RECOGNITION

The initial ball recognition aims at identifying the ball at the beginning of a scene without any prior knowledge of an appropriate template. In contrast to the on-the-fly processing, the initial ball recognition has a higher computational demand and introduces a small delay at the beginning of each scene, which should not pose a big problem for the buffer. Basically, with the initial ball-recognition, we try to find possible ball-candidate motion-trajectories to determine an appropriate initial template for the successive recognitions in the on-the-fly processing. The method of motion trajectories is usually much more reliable and robust, because knowledge about the physical behavior of the ball can be used to identify favourable ball candidates. Similar approaches have already been investigates, e.g. [6], but those typically need a large number of processed frames and use

a more complex algorithm to find the most probable ball-candidate trajectory, which makes the usage in an online streaming system very questionable.

4.1 Minimum Distance Polygons

The initial ball recognition starts with a set of characteristic templates obtained from analysis of a large number of soccer sequences with different illumination, field color, resolution, size and contrast values, which we denote τ. The templates are rectangular, thus showing also non ball pixels. In order to keep the size of the template small, we color the non ball pixels of the templates with the dominant color of the actual video to make them suitable for the template matching process (comparable to the "blue-screen" technique).

First of all, we perform a dominant color detection and replacement to simplify the recognition process, and a line detection to identify possible clutter, caused by the field lines. After this image preprocessing, we conduct a template matching by using the described SAD similarity metric with all templates within the set $\tau = \{t_1, t_2, \dots t_N\}$. In contrast to choosing the most probable ball candidate (denoted by the lowest SAD value), we now extract the 15 most probable ball positions. Due to our substitution of the recognised field lines in the preprocessing, balls located on one of the field lines will be effectively discarded.

The above procedure of preprocessing and template matching is repeated for a number W of frames (typically chosen around 7), depending on the frame rate. The choice of W strongly influences the performance of the initial ball search, since a larger number allows for a better extraction of the motion trajectory. Nevertheless, as mentioned, the initial ball-recognition has a high computational demand that grows linearly with W. In addition, if W is very high, the probability that the motion trajectory is affected by players in a non-smooth way gets higher, thus making it less suitable for our algorithm. In case that the frame rate is about 25 fps, and the resolution is either CIF or QCIF, we identified a value between 5 and 10 as a good choice to ensure a high recognition probability and a reasonable computational complexity (processing delay).

After the preprocessing and template matching, as well as candidate list verification, $n_i(f)$ candidate positions are found for each template t_i $(i = 1, \dots, N)$ in each frame $f = 1, \dots, W$ respectively. The total number of candidate positions for a specific template t_{i_0} is thus given by $\sum_{f=1}^{W} n_{i_0}(f)$. If this total number is less or equal to two, no reliable motion trajectory can be estimated and thus, the corresponding template is discarded for further processing (removed from the set τ), such that the initial ball recognition only has to deal with a smaller set $\tau' = \{t_i\}, i = 1, \dots, N'$.

The remaining ball candidate positions and templates now are the basis for the extraction of the *minimum distance polygons*. Let us denote a specific candidate position corresponding to the template t_i in the frame f by $\mathbf{c}_{k,i}(f), k = 1, \dots, n_i(f)$. Then, starting from an arbitrary candidate position $\mathbf{c}_{k_0,i}(1)$ in the first frame of the scene, we calculate the Euclidean distances,

$$d(a, b, f, t_i) = \|\mathbf{c}_{a,i}(f) - \mathbf{c}_{b,i}(f+1)\|^2 , \qquad (3)$$

to all other candidate positions (of that template) in the subsequent frame, i.e. $d(k_0, l, 1, i)$. The ball-candidate position $\mathbf{c}_{l,i}(2)$ with the minimal distance to the starting point $\mathbf{c}_{k_0,i}(1)$, in the first frame, then forms the second point of the first minimum distance polygon corresponding to the specific starting point.

This procedure is repeated for all candidate positions of the first frame, thus forming $n_i(1)$ lines for the template t_i. Consequently, we use the candidate positions with minimum distance in frame two $\mathbf{c}_{l,i}(2)$, and calculate the Euclidean distances to the positions detected in frame three, i.e. $d(l, m, 2, t_i)$. And again, the points with minimum distance represent the next point of the polygons. This procedure is repeated until all $W = 7$ frames have been computed. Finally, we end up with $n_i(1)$ polygons, denoted $\mathbf{p}_{i,m}$ $(m = 1, \dots, n_i(1))$, for each template t_i.

For the practical implementation, we have to deal with two aggravating situations: 1) It is possible that the correct ball position is not recognised in the first frame (e.g. due to occlusion). Therefore, all candidate positions $\mathbf{c}_{k,i}(2)$ and $\mathbf{c}_{k,i}(3)$ are allowed to form starting points of new polygons. If a resulting polygon "merges" with an already known polygon (i.e. starting from frame one or two), it is discarded, because the merge suggests that the more reliable polygon has already been found. 2) It can happen that in a frame no candidate position is left over from the withdrawal of indeterminable ball-positions $(n_i(f) = 0)$. To track these circumstances, this frame is excluded from the computations to obtain the minimum distance polygon. If there are more than three frames in which no candidate position is left, the according template is excluded from the minimum distance polygon computation, because it is not guaranteed that reliable polygons will be found.

4.2 Determining the Optimal Polygon

The remaining minimum distance polygons represent possible trajectories of the candidate balls. As a metric to decide for the correct ball, we chose the sum of squared errors (SSE) between the minimum distance polygon $\mathbf{p}_{i,m}$ and its fitted polygon of order two, $\phi_{i,m}$,

$$\mathrm{SSE}(\mathbf{p}_{i,m}, \phi_{i,m}) = \sum_{f=1}^{W} \left(\mathbf{c}_{k,i}(f) - \tilde{\mathbf{c}}_{k,i}(f)\right)^2 , \qquad (4)$$

where $\mathbf{c}_{k,i}(f)$ and $\tilde{\mathbf{c}}_{k,i}(f)$ denote the candidate positions of the minimum-distance polygon $\mathbf{p}_{i,m}$ and its fitted version, $\phi_{i,m}$, respectively.

The identification of the "optimal" polygon is performed in two steps: 1) exclude all polygons with an SSE value above a fixed threshold (ensures that the motion trajectory is sufficiently smooth) and 2) within the remaining polygons, choose the one with the largest sum of Euclidean distances over the length of $W = 7$ frames. We chose this strategy, because we observed that in some cases, clutter in the field is interpreted as a possible ball candidate. These wrong candidates do not move much from frame to frame, leading to polygons which can be well fitted with low SSE values, but only show very short total polygon distances. Accordingly, we identify the trajectory with a very low SSE value but maximum sum of distances as the most probable candidate for the correct ball, see Fig. 4 for an example.

5. ON-THE-FLY PROCESSING

As shown in Section 4, the initial ball recognition provides an appropriate template and the first $W = 7$ ball positions of a scene. Despite the reliability of the method, it

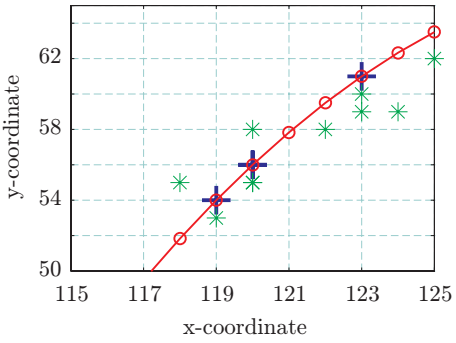


Figure 4: Example of an optimally fitted polygon. The optimal polygon is marked as a solid line with circles, the according positions underlying the fitting process are marked by crosses and other, wrong candidate positions are marked by stars.

is computationally very demanding. Depending on the chosen resolution of the uncompressed video, due to the whole frame SAD-based template matching, the optimal polygon search can be very time consuming. For a limited number of frames, such an accumulative delay can be tolerated, since the playout buffer at the user terminal is able to compensate it. Nevertheless, for the computation of the remaining frames of the scene, a different approach to detect the ball has to be developed.

A possibility to reduce the computation time of the template matching is to limit the search area to a certain region of interest (ROI) $\mathcal{A}$. Accordingly, for the on-the-fly processing, a consequent tracking of the motion trajectory of the ball is implemented, where the preceding ball positions are used to predict the position in the present frame. This estimated ball position is than used to determine the position and size of an appropriate ROI for the template matching.

5.1 Trajectory Tracking

We implemented an MMSE linear predictor on $\Delta\mathbf{c}(f) \triangleq \mathbf{c}(f) - \mathbf{c}(f-1)$ to reliably predict the preceding ball position and determine a suitable ROI, i.e.

$$\Delta\hat{\mathbf{c}}(f) = \begin{pmatrix} \Delta\hat{c}_x(f) \\ \Delta\hat{c}_y(f) \end{pmatrix} = \sum_{j=1}^{O} \begin{pmatrix} a_{x,j} \cdot \Delta\hat{c}_x(f-j) \\ a_{y,j} \cdot \Delta\hat{c}_y(f-j) \end{pmatrix}, \quad (5)$$

where $\Delta\hat{\mathbf{c}}(f)$ denotes the predicted "speed" value, and $a_{x,j}$, $a_{y,j}$ denote the MMSE filter coefficients for the x and y direction, respectively. The filter coefficients were determined by a minimisation of the Mean Square Error (MSE) $\mathbb{E}\left\{\|\Delta\mathbf{c}(f) - \Delta\hat{\mathbf{c}}(f)\|^2\right\}$ between the real ball movement $\Delta\mathbf{c}(f)$ and the predicted one $\Delta\hat{\mathbf{c}}(f)$.

Unfortunately, the statistics of the real ball motion are unknown a priori, thus a collection of 20 football sequences in CIF resolution digitally acquired at 25fps, each of them containing an average of 180 frames has been investigated. This collection has been used to compute the correlation matrix by averaging over the 20 correlation matrices for each sequence. It turned out that an MMSE order of three is sufficiently accurate for our problem.

Once the ball position has been predicted, the necessary steps to recognise the ball (i.e. image preprocessing and template matching) only take place in a ROI $\mathcal{A}$ centred at the predicted ball position. The size of the ROI is determined by the present speed of the ball, the frame rate and the template dimension. More precisely, in 25 fps videos, we calculate the size of the ROI to be five times the size of the template. Limiting the search region to the ROI offers some significant advantages over a full frame processing. The computational complexity is reduced significantly according to the usually small size of the ROI, and a lot of false hits that would occur outside of the ROI are avoided.

To identify the ball candidate, we only use a comparison of the minimum SAD value within the ROI with a fixed threshold. In case that the minimum SAD value position neither exceeds the given threshold, nor lies on one of the detected lines, the found candidate position is assumed to be valid. Otherwise, the algorithm reports a possible occlusion and rejects the ball replacement. Furthermore, in this case no reliable prediction of the next ball position is possible, so the center of the ROI is kept at the present position and the size of the ROI is slightly increased to ensure that the ball does not cross the border of the ROI. This procedure is repeated until the ball is found again, so that both the prediction and the original ROI size are applied again.

5.2 Template Learning

To build our algorithm as robust as possible against small ball appearance changes during a scene, we implemented an update of the template ball. The template t is updated by the newly recognised ball in an weighted manner, i.e. the new ball b is multiplied with an update factor α and the resulting new template t' is obtained according to $t' = (1-\alpha)t + \alpha b$.

6. BALL REPLACEMENT

The final part of the proposed system is the ball replacement by a suitable substitution-ball to enhance the viewing experience after compression. Before the replacement, the actual size of the ball is determined by a binary map analysis. In case that the recognised ball size exceeds a threshold, no replacement is necessary – and thus will not be conducted. In the opposite case, the substitution-ball has to be large enough and must have sufficient contrast to the surrounding pixels in the frame (i.e. the field) to guarantee the desired effect on the user terminal.

Due to the low resolution, the substitution-ball can be generated in a very simple manner. According to the recognised ball size, the substitution ball is chosen to be circular and filled with pure white color. This filled white circle is generated on a background that equals the actual dominant color (i.e. the average field color) and afterwards, simple Gaussian filtering ensures a smooth transition between the ball and the field as it occurs in practice. For small ball sizes, the so generated replacement is realistic enough to ensure a ball-like appearance in the resulting video and is satisfying our demand on contrast and size.

7. EXPERIMENTAL RESULTS

To determine the reliability of our algorithm to ground truth, we measured the "recall" and "precision" ratios, recall = $N_C/(N_C + N_M)$ and precision = $N_C/(N_C + N_F)$, with N_C denoting the number of correct ball recognitions, N_M specifying the number of missed recognitions and N_F denoting the number of false recognitions. In Table 1, we show the results of the on-the-fly processing of four QCIF test sequences. The computation time of the algorithm is strongly

Figure 5: Visual result of the ball detection and replacement. The substitution ball is chosen in a way to ensure that it is clearly visible at the user-terminal.

Table 1: Results of our on-the-fly algorithm for QCIF test video sequences.

	frames			detections				
video	total	with ball	without ball	correct	false	missed	recall	precision
"soccer 1"	141	123	18	113	0	10	0.85	1.00
"soccer 2"	141	137	4	130	1	6	0.91	0.99
"soccer 3"	275	266	9	257	6	3	0.96	0.98
"soccer 4"	370	306	64	269	4	33	0.79	0.99

dependent on the resolution of the video to process. Furthermore, if a lot of scene changes occur, the computationally more complex initial ball recognition has to be executed more often, thus resulting in a larger computation time. We tested the algorithm on a standard PC, in MATLAB, which resulted in an average frame-processing time of 115 ms (for QCIF resolution videos with moderate occlusions). Typical frame rates used in mobile networks are around 15 fps (i.e. 66 ms frame time), so that it seems that the speed enhancement necessary to achieve real-time precessing ability can likely be reached by a optimized implementation (i.e. coded in C++).

8. CONCLUSIONS

A complete algorithm to improve the ball appearance in low resolution soccer videos with low computational complexity has been developed. The difficulties in the ball recognition task are solved using image preprocessing, trajectory tracking and prediction techniques, together with a consequent scene change detection. The desired ball appearance improvement is achieved by replacing the detected original ball by a suitable substitution-ball, which guarantees an appearance improvement at the user-terminal side. We consider our proposed technique a good possibility for mobile network operators to enhance the subjective video quality of soccer videos, without major changes in existing hardware.

Acknowledgment

The authors would like to thank mobilkom Austria AG for supporting their research. The views expressed in this paper are those of the authors and do not necessarily reflect the views within mobilkom Austria AG.

9. REFERENCES

[1] *Transparent end-to-end Packet Switched Streaming Service (PSS); Protocols and Codecs*, 3GPP Std. TS 26.234, Rev. 7, Mar. 2007.

[2] O. Nemethova, M. Zahumensky, and M. Rupp, "Preprocessing of ball game video-sequences for robust transmission over mobile networks," in *Proceedings of the 9th CDMA International Conference (CIC 2004)*, Seoul, Korea, October 2004.

[3] A. Ekin, A. M. Tekalp, and R. Mehrotra, "Automatic soccer video analysis and summarization," *IEEE Transactions on Image Processing*, vol. 12, no. 7, pp. 796–807, July 2003.

[4] M.Leo, T. D'orazio, and A. Distante, "Independent component analysis for ball recognition in soccer images," in *Proceedings of International Conference on Intelligent Systems and Control*, Salzburg, June 2003, pp. 351–355.

[5] J. Ren, J. Orwell, and G. A. Jones, "Estimating the position of a football from multiple image sequences," BMVA One Day Symposium on Spatiotemporal Image Processing, London, UK, Mar. 2004.

[6] D. Liang, Y. Liu, Q. Huang, and W. Gao, "A scheme for ball detection and tracking in broadcast soccer video," in *Advances in Multimedia Information Processing - PCM 2005: 6th Pacific-Rim Conference an Multimedia*, Jeju Island, Korea, Nov. 2005, pp. 864–875.

[7] J.-R. Ohm, *Multimedia Communication Technology*. Springer, 2004, ISBN 3-540-01249-4.

[8] B. Jähne, *Practical Handbook on Image Processing for Scientific Applications (Hardcover)*, 1st ed. CRC-Press, May 1997.

[9] A. Hanjalic, *Content-based analysis of digital video*. Kluwer Academic Publishers, 2004, ISBN 1-4020-8114-6.

[10] A. Dimou, O. Nemethova, and M. Rupp, "Scene change detection for h.264 using dynamic threshold techniques," in *Proceedings of 5th EURASIP Conference on Speech and Image Processing, Multimedia Communications and Service*, Smolenice, Slovak Republic, June 2005.