



APIDIS

Autonomous Production of Images based on Distributed and Intelligent Sensing

STREP Project, 1st FP7-216023

D4.1 Local spatio-temporal visual feature extraction

Due date of deliverable: 31-12-2008

Actual submission date: 09-01-2009

Start date of project: 1st January, 2008

Duration: 36 months

Lead contractor for this deliverable: QMUL

[Final v1]

| D4.1 | Local spatio-temporal visual feature extraction |
|-------------------------|---|
| Project Acronym : | APIDIS |
| Contract No : | FP7-216023 |
| Due Date : | 31-12-2008 |
| Reply To: | Fahad Daniyal Email fahad.daniyal@elec.qmul.ac.uk |
| Actual date of delivery | 09-01-2009 |

1. Executive Summary

The purpose of this document is to highlight and motivate the choice of local spatio-temporal features used in APIDIS: we describe the methods employed for the extraction of these features, and discuss their spatio-temporal distributions. These distributions are then used as input to WP5 and WP6 for event detection and autonomous content production. The features we consider are motion segmentation results, tracked features and Scale Invariant Feature Transform (SIFT) on spherical manifolds.

Deliverable Identification Sheet

| | |
|--------------------------------------|---|
| Project ref. no. | FP7-216023 |
| Project acronym | APIDIS |
| Project full title | FP7-216023 |
| Security (distribution level) | Public (PU) |
| Contractual date of delivery | Month 12, December 31, 2008 |
| Actual date of delivery | Month 12 |
| Deliverable number | D4.1 |
| Deliverable name | Local spatio-temporal visual feature extraction |
| Type | Report |
| Status & version | Final |
| Number of pages | 20 |
| WP / Task responsible | WP4 / QMUL |
| Other contributors | ACIC, EPFL |
| Author(s) | Fahad Daniyal, Christophe Parisot, Yannick Boursier |
| EC Project Officer | Albert Gauthier |
| Abstract | This document describes and motivates the choice of low-level features used in the APIDIS project. This document describes the methods employed for the extraction of these features. |
| Keywords | Low level feature analysis, change detection, background subtraction, feature tracking, SIFT on spherical manifolds |
| Sent to peer reviewer | UCL |
| Peer review completed | Yes |
| Circulated to partners | Yes |
| Read by partners | Yes |
| Mgt. Board approval | Pending |

Table of contents

| | |
|---|----|
| 1. Executive Summary | 2 |
| 2. Introduction | 5 |
| 3. Feature description | 6 |
| Change detection (QMUL) | 6 |
| Features trajectories (ACIC)..... | 6 |
| Local spatio-temporal features from Omni cams (EPFL)..... | 6 |
| 4. Proposed approach..... | 6 |
| Change detection | 6 |
| Results..... | 8 |
| Features trajectories..... | 11 |
| Objectives..... | 11 |
| Algorithm | 11 |
| Output formatting | 14 |
| Results..... | 15 |
| Conclusions | 17 |
| Omnivision local features | 17 |
| 5. Conclusions..... | 19 |
| References | 20 |

2. Introduction

This document describes the set of local spatio temporal features extracted in the framework of the APIDIS project. A visual feature at any given time represents interesting characteristics of local appearance. The purpose of this document is to describe the selected low-level features and motivate their choice. This document also describes the methodology involved in the extraction and the representation of these features.

The selected features (and responsible project partner) are:

- 1) Change detection (QMUL)
 - 2) Local motion trajectories (ACIC)
 - 3) Local spatio-temporal features from omni vision cameras (EPFL)
-

3. Feature description

Change detection (QMUL)

Change detection represents the amount of activity (motion) in the scene; it can be regarded as the change which occurs in each image with respect to a reference image. The aim of change detection is to identify regions of high activity in the camera view.

The segmentation of an image in two classes, namely background and foreground is performed using a colour change detector that extracts colour features from the perceptually uniform colour space CIELab.

Local feature trajectories (ACIC)

Features trajectories provide a description of local motion in each video stream; it is represented by a set of trajectories for moving parts of the pictures. The local features for which trajectories are computed correspond to square blocks of pixels free of any semantic description. While the change detection algorithm provides colour and texture information where some activity is detected, the features tracks provide information about the direction and the amplitude of motion in active parts of the video frames. Motion information can for example be used to zoom in on the regions with highest speed during the production step.

Local spatio-temporal features from Omni cams (EPFL)

After an expansive study of the literature concerning detection and description of features on images, we have chosen to implement and use the Scale Invariant Feature Transform (SIFT) on spherical manifold, well adapted to omnidirectional images. The SIFT features are local and based on the appearance of the object at particular interest points, and are invariant to image scale and rotation. The results provided by the SIFT on sphere can be easily compared to the ones produced by the classical SIFT on planar images, and then make possible a joint collaborative process between the two kinds of images. The main advantages of SIFT is their robustness to changes in illumination and noise. They are highly distinctive, and allow for correct object identification with low probability of mismatch. In addition, describing an object by a set of SIFT features is robust to partial occlusion

4. Proposed approach

Involved partners: QMUL

Change detection

Change in the foreground, due to the activity in the scene, is detected using a statistical colour change detector [1]. The steps involved are shown in Figure 1. Initially a reference image I_i^{ref} is generated for each camera C_i using adaptive background learning. This reference image depicts the state of the system at time

$t = 0$, with no observable features of interest. Figure 2, shows example frames for background learning.

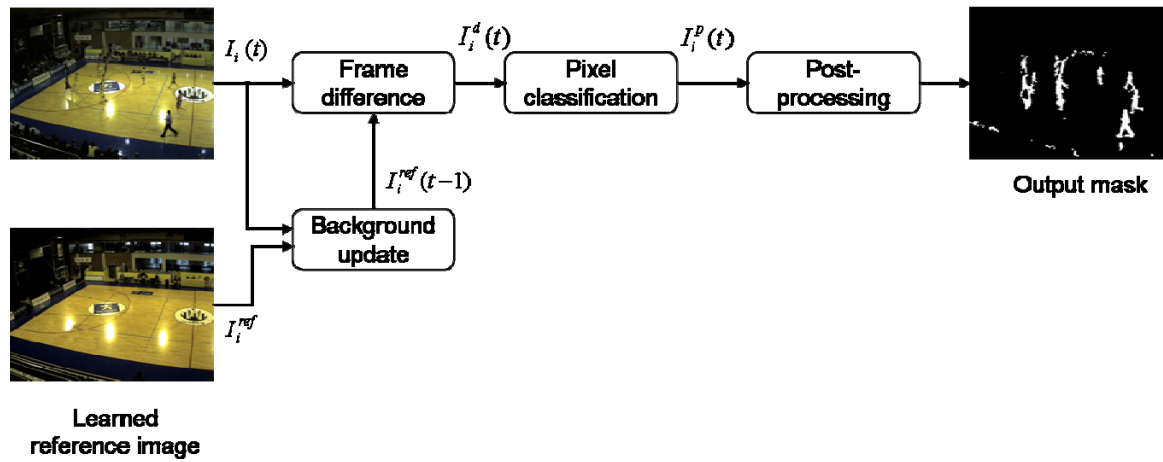


Figure 1 Block diagram showing the steps involved in the change detection algorithm

Let $I_i(t)$ be an input image from camera C_i at time t . Then the difference image $I_i^d(t)$ at time t is calculated as

$$I_i^d(t) = |I_i^{ref}(t-1) - I_i(t)|$$



Figure 2- Results for background learning. Starting from frame 0, the foreground objects are removed from the image.

We represent pixel at location (x, y) at time t in image $I_i^d(t)$ as $I_i^d(x, y, t)$. Pixels are classified as foreground or background based on dynamic thresholding. This threshold is learned through noise modelling. This model assumes an additive white Gaussian noise on each colour channel which is estimated separately. The noise removal is based on the hypothesis that the additive noise affecting each image of the sequence follows a Gaussian distribution with zero mean and standard deviation σ_i . The value of σ_i is computed by analyzing the image difference in areas without moving objects. This method verifies when $I_i^d(x, y, t) \neq 0$ because of the camera noise as opposed to other factors like moving object or illumination changes. Based on this hypothesis, H_o , the conditional probability density function for each pixel is defined as

$$f(I_i^d(x, y, t) | H_0) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{I_i^d(x, y, t)^2}{2\sigma_i^2}}$$

The noise model is applied on groups of pixels (blocks). Morphological processes are applied on the difference image after pixel classification $I_i^p(t)$ to reduce isolated classification noise.

To compensate for global illumination changes or small background changes we use on-the-fly background learning. The learning is controlled by the learning factor κ , which defines the update rate of the background. This update of background can be expressed as

$$I_i^{ref}(t) = \kappa I_i(t) + (1 - \kappa) I_i^{ref}(t - 1)$$

The choice of the value of κ is dependent on the dynamics of the scene. Figure 3 shows an example of the effect of learning factor on the learned background. If we choose a very low learning factor then the system becomes susceptible to the slightest changes in the illumination. If the value is too high then any object that stays still for a small amount of time is regarded as a part of background.



Figure 3-Effect of learning rate on the constructed background after 100 frames with (a) 10% (b) 50% learning rate

Results

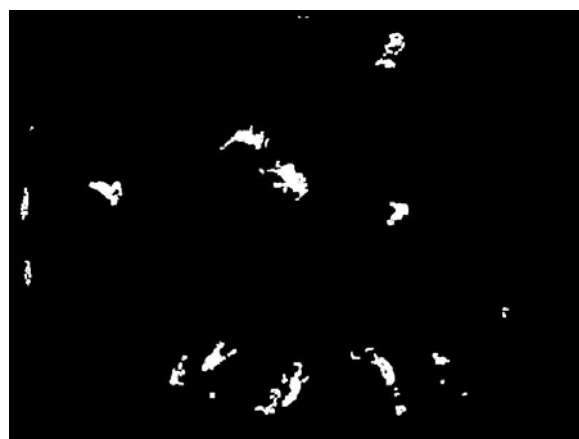
Sample results for the change detection in the seven cameras are given in Figure 4.



(a) Camera 1



(b) Camera 2



(c) Camera 3



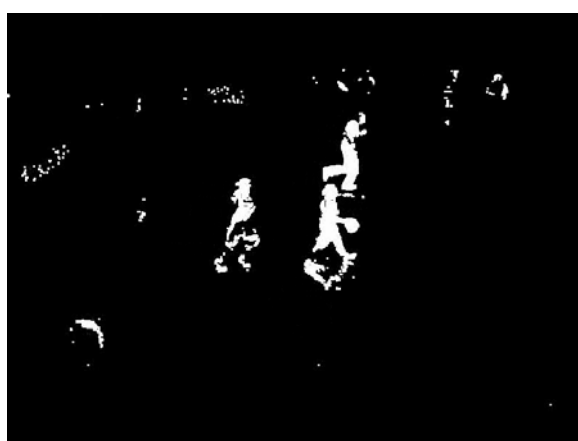
(d) Camera 4



(e) Camera 5



(a) Camera 6



(a) Camera 7

Figure 4-[a-g] Sample results for the change detection, where white represents the regions of the image with activity.

Local features trajectories

Involved partners: ACIC

This section deals with the extraction of low level motion information from the video streams.

Objectives

The goal is to extract a set of trajectories of pictures moving parts in each video stream. This will provide the distribution of motion in the frames in terms of location, direction and amplitude. This data will serve as input for the automatic event recognition (WP5) and autonomous production algorithms (WP6).

Algorithm

Definition of a feature track

The basic element that is detected and tracked with the algorithm is a square sub-image region of size eight by eight pixels. Although the size of this block could be easily changed, an eight by eight block size is generally considered as a good compromise by block matching video processing algorithms in terms of content information and block matching processing cost.

A feature track can be defined as its last eight by eight pixel values plus the history of its positions in previous frames.

General algorithm overview

For each new frame of a video stream, the steps of the algorithm are the following:

- Update of existing features trajectories
- Computation of possible initial positions for tracking new features
- Initialization of new tracks

The algorithm can be summarized by Figure 5.

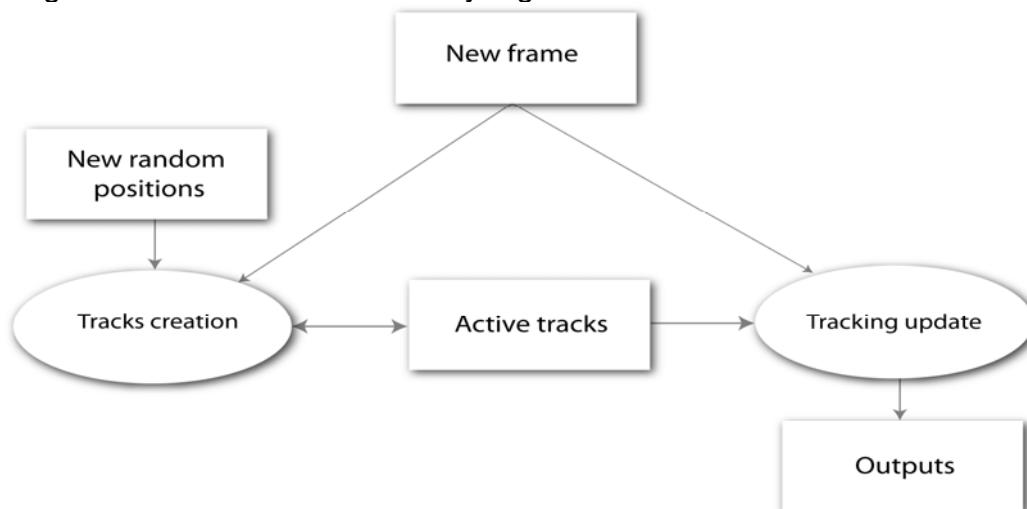


Figure 5: Overview of the scheme for the computation of local features trajectories

Inputs

The algorithm takes as inputs:

- A video stream
- A set of parameters relative to this video stream (e.g. region of interest, calibration, etc.)

The algorithm does not use other video processing outputs. In particular, it is not linked to the output of the background/foreground segmentation module.

Update of existing features trajectories

The current set of activated features is updated using a block-matching approach.

First, the position of the feature is predicted in the current frame thanks to the history of previous positions. A linear prediction of the position is used assuming that the acceleration is negligible from one frame to the next one.

Next, the best position in the new frame is searched around the predicted position using a diamond-search block matching algorithm. The distance for comparing two blocks is the sum of the absolute differences between the pixel values. If the distance computed for the best position is too high, then it is decided that the feature could not be tracked anymore and the feature track is terminated.

Finally, each of the features tracks can be kept active or rejected according to several considerations on their trajectories (see future section “Available parameters”).

Random distribution of initial features positions

While typical block matching algorithms split all frames in eight by eight blocks and compute a motion vector for each of these blocks, we use only a subset of candidate positions to initialize the tracking of new features.

Indeed, a set of candidate positions is randomly generated for each new frame. The goal is to have a high probability of detecting/tracking all local motions without the cost of an exhaustive block matching search over the entire frame.

Activation of the features

For each of the randomly defined positions in the current frame, a basic frame differencing approach is used to determine if a significant change occurred in the eight by eight sub-image region located at this position. The distance function which is used is the sum of the absolute differences between the pixels values. When the distance is over a pre-defined threshold, a new feature block is activated at this position for tracking in the next frames.

Available parameters

Here is a list of the algorithm parameters:

- Region of interest. This is a convex polygon that defines the pictures region in which the algorithm performs the features tracking algorithm.
-

- Number of features. This is the maximum number of features. This number must be high in order to detect most of the motion information. We have typically set it to 800.
- Number of tracks. This is the maximum number of active tracks. This number allows us to place a ceiling on the processing time of a new frame since most of the algorithm cost is due to the block matching search of each active feature track. We have typically set it to 600 since we are working offline and have therefore less time constraints than in real time processing.
- Size of the feature. Set to eight by eight pixels.
- Search space. This parameter sets the maximum number of tested positions in the diamond-search block matching algorithm. This implicitly defines the searching area. This parameter depends on the level of unpredictability of the observed motion. As an example, we use 40 positions for the two top cameras.
- Minimum duration of a track. All tracks that are terminated before this threshold are not published. This parameter allows removing some tracks that correspond to noise. A typical value is 0.5 second.
- Minimum speed. All tracks that never exceeded this speed are not published. This parameter allows removing tracks that may correspond to noise or at least to non discriminative motion.
- Minimum linearity. All tracks that have very erratic trajectories are not published. This parameter allows removing tracks that probably correspond to noise or to camera vibrations. In our context, this parameter is left somewhat permissive since the players motion may be chaotic.
- Feature activation threshold. This parameter is the maximum sum of absolute difference between pixel values for the activation of a new track. It is set to 32 for all cameras in our algorithm.

Figure 6 shows a snapshot of the feature tracking algorithm. The legend is as follows: Grey circles are candidates for initializing new tracks. Red and black crosses represent respectively too long breaks and too slow tracks. Magenta lines are for tracks which are too static (total distance was too small). Cyan lines are for tracks which were too short in duration. Blue lines are for tracks with unstructured or discontinuous motion. Green trajectories are considered as relevant for high level interpretation of the scene.

Difference with particle filter algorithms

In a sense, the algorithm behaves like particle filtering. The main difference with particle filter algorithms is that we don't try to group the particles into entities. The algorithm does not try to optimize the number of tracks around existing ones or to manage graphs of connected particles. The algorithm never kills a track due to the fact that it is too far from other particles. Indeed, the algorithm does no interpretation of the spatial distribution of active tracks. The goal here is to provide local motion information without any semantic description of the moving objects.

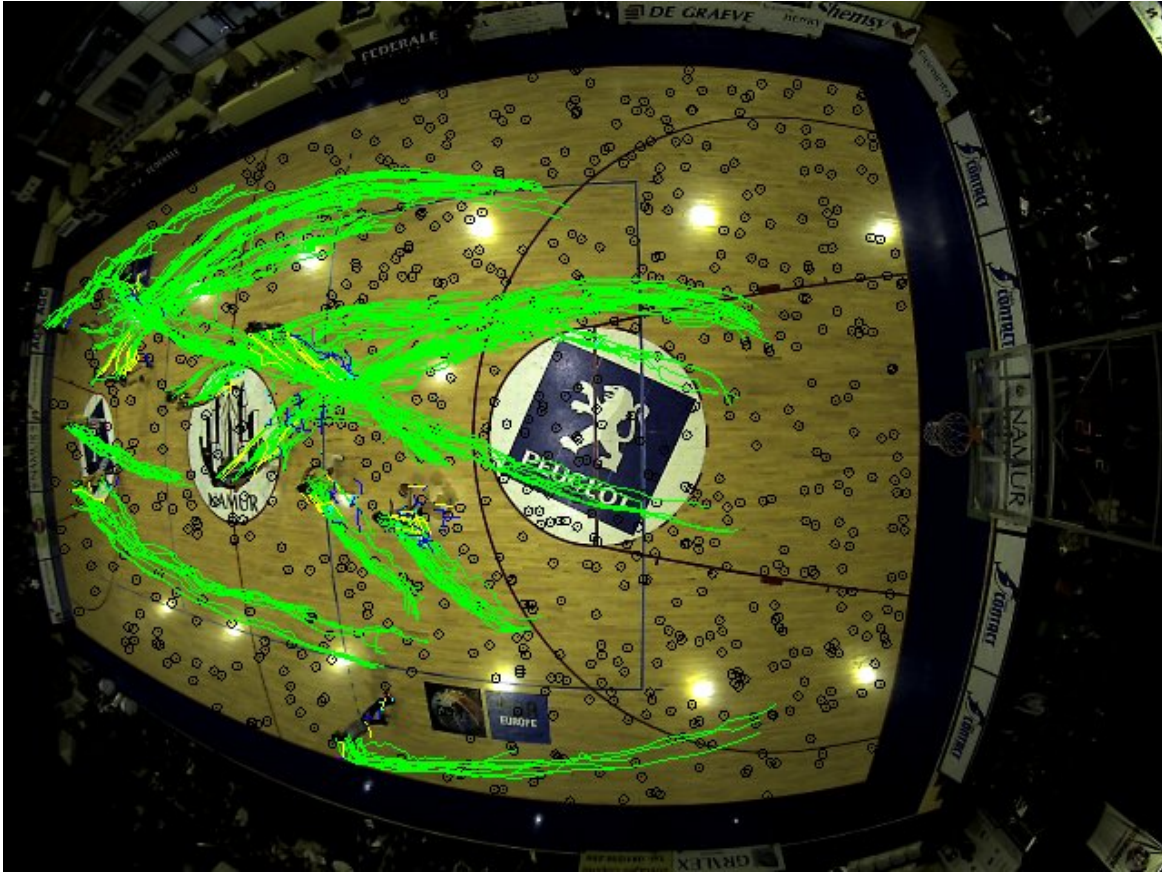


Figure 6: Snapshot of the feature tracking algorithm run on camera 3 stream.

Difference with block matching algorithms

The algorithm uses the same method as block matching algorithms for the computation of the displacement of eight by eight blocks from one frame to the next one. However, the displacement is not computed all over the image on a fixed grid of eight by eight image blocks. The displacement is only computed at the previous position of each track. This results in the tracking of eight by eight blocks over time rather than providing only frame to frame information. The computation cost is highly reduced since the block matching procedure is only computed in regions that are active.

Output formatting

Consolidated feature tracks are published in XML files. The filenames follow the structure proposed in deliverable D6.1 [3]. The files correspond to periods of one minute. For example:

```
...  
camera3_20080409T164700Z.tracks.xml  
camera3_20080409T164800Z.tracks.xml  
camera3_20080409T164900Z.tracks.xml  
...
```

The content of each file describes the stream ID it refers to, the size of the video frames in pixels as well as the time zone used for the timestamps in the XML file.

For each frame, the timestamp is provided and all active feature tracks with their ID and position in the current frame are listed. The files look like:

```
<tracks>
  <stream>camera3</stream>
  <framesize>
    <width>1600</width>
    <height>1200</height>
  </framesize>
  <timezone>Z</timezone>
  <frame>
    <time>
      <date>2008-04-09</date>
      <timestamp>16:47:00</timestamp>
      <fraction>000000</fraction>
    </time>
    <track>
      <id>133146</id>
      <position>
        <x>393</x>
        <y>262</y>
      </position>
    </track>
    <track>
      <id>133230</id>
      <position>
        <x>388</x>
        <y>264</y>
      </position>
    </track>
    ...
  </frame>
  <frame>
    ...
  </frame>
  ...
</tracks>
```

Results

The following table provides an overview of the performances obtained by the features tracking algorithm on each of the seven basket ball video cameras.

The following procedure was used for performance evaluation:

- Since each camera focuses on one of the two parts of the basket ball field, the performance is measured only in this region even if the feature tracking algorithm was run on the complete basket ball field, e.g. in case of the top cameras. The reason for reducing the region of interest is that a symmetrical camera is assumed to provide better results on the other part of the basket ball field.

- Since the manual annotations are not perfect, we have decided to automatically extend all the bounding boxes of the manually annotated objects by ten pixels in each direction. Note that there are still parts of players that are outside these extended bounding boxes.
- The performances are measured on the one minute period of manually annotated dataset.

The table columns are the following:

- Manually annotated bounding boxes: number of manually annotated bounding boxes that intersect the region of interest.
- Missed bounding boxes: ratio of those bounding boxes that contain no feature track.
- Average number of active feature trajectories per bounding box: for each bounding box that intersects the region of interest, we sum the number of feature tracks that are included in the bounding box. This number is divided by the first column to get an average number of tracks per bounding box.
- Feature trajectories positions out of manually annotated bounding boxes: this is the ratio of features tracks positions which fall inside the region of interest but outside all bounding boxes.

Table 1: Per camera performance of the feature tracking algorithm

| Camera ID | Manually annotated bounding boxes | Missed bounding boxes | Average number of active feature trajectories per bounding box | Features trajectories positions out of manually annotated bounding boxes |
|-----------|-----------------------------------|-----------------------|--|--|
| 1 | 7206 | 1.75% | 24.17 | 11.66% |
| 2 | 6749 | 1.13% | 48.73 | 6.39% |
| 3 | 6328 | 2.05% | 12.94 | 8.49% |
| 4 | 6926 | 1.67% | 57.63 | 9.49% |
| 5 | 8196 | 2.03% | 17.85 | 9.98% |
| 6 | 6625 | 1.75% | 30.20 | 7.50% |
| 7 | 6617 | 9.42% | 23.24 | 11.53% |
| Average | 6950 | 2.83% | 30.68 | 9.29% |

The table shows that the number of missed bounding boxes is very low. Since the static bounding boxes are taken into account in the performance measurement (when one player position does not change for a while), this means that the feature tracking algorithm behaves well. The average number of active feature tracks varies from one camera to another one since the size of the objects in pixels varies from one view to another one. For all cameras, there are at least ten active tracks per bounding box in average. The number of features positions falling outside bounding boxes is around 10%. We could observe that some of those erroneous feature tracks positions correspond to shadows. However, most of these bad positions are tracks that could not follow players' motion. This results in tracks that leave behind the players and stay on the basket ball field before being caught up by a new player or dying after the conditions for being kept active

are not valid any more. In any case, the tracks that have bad positions are located in regions where there has been motion recently.

Conclusions

An algorithm for computing local motion information has been proposed. It produces a list of partial features tracks. The results are in line with what could be expected from such a low level feature extraction module with a very low missed objects ratio

Omnivision local features

Involved partners: EPFL

We have adapted the Scale Invariant Feature Transform to the spherical manifold.

The algorithm is based on invariance in scale space and involves the following two main steps:

i) Detect of Scale-Space Feature: interest points are detected at this stage. The input image is blurred by a Gaussian filter bank generated for a range of variance values fixed by the user, and the Difference of Gaussian (DoG) spherical images are considered at multiple scales. Interest points are selected as those which are local extrema values of the DoG images across neighbouring scales.

ii) Discard low-contrast key points and remove edge points: some interest points previously selected may have a low contrast, and then be sensitive to noise. They are removed by comparing the value of the second derivative of the DoG image to a threshold fixed to 0.03. Moreover, this algorithm is designed such that points along edges provide a strong response. To ensure robustness to noise for edge points, information from the local principal curvature is extracted, and used to remove pixels that are too sensitive to noise.

Results obtained on two different images during the first acquisition campaign are displayed on Figure 7 and Figure 8. On these images, the different actors of the scene, e.g. the players and the referee, are clearly characterized by several key points. More precisely, some specific pattern like the number of a player on shirts can be identified, which may a start point for applications such that an automatic identification of the player.

Each key point returned by SIFT contain a lot of information and typically no more than 3 SIFT are needed to describe an object and to identify it without any ambiguity.

The final key points are provided along with a descriptor that statistically characterizes the feature, and allow matching with other features on different images for a tracking application.

The descriptors will be defined along with the other partners to efficiently perform tracking and to be consistent with descriptors needed as input of the tracking algorithms developed for standard cameras.



Figure 7-Result of the SIFT performed on fisheye images, where each red cross represents a final key point



Figure 8-Result of the SIFT performed on fisheye images, where each red cross represents a final key point.

5. Conclusions

In this document we described the local spatio-temporal features extracted for the APIDIS framework. These features represent the local appearance of the scene at hand and are directly calculated from the image itself. These features can be fused with high level features such as the output of a person (player) detector, feature tracks and contextual knowledge for semantic analysis of the scene in order to facilitate the automatic production of meaningful video content.

References

- [1]. M. Taj, E. Maggio, and A. Cavallaro, "Multi-feature graph-based object tracking," in CLEAR, Springer LNCS 4122, Southampton, UK, April 2006, pp. 190–199.
 - [2]. APIDIS deliverable D3.1 "Deployment of the acquisition and storage system, including calibration and annotation issues".
 - [3]. APIDIS deliverable D6.1 "Software environment definition through release of test-bed for flexible access to content".
-