Mean shift object Tracking
A computer vision technique to track objects

S. Korzec B. Sc.   0252700
skorzec@science.uva.nl
University of Amsterdam, December 2006

1. Introduction

In the field of computer vision scientists are looking for new methods to make objects in an image or a movie understandable to a computer. Humans are able to immediately understand and distinguish objects because they have knowledge about the objects in the real world and they understand the rules of the real world. These rules of the world are not understandable by a computer. An image is a mere approximation of the real world, given as a collection of pixels with a certain value. In computer vision we try to use methods which can approximate or even outperform the speed and accuracy at which a human can recognize objects. Many of these methods need extensive computing power, such as eigenvector analysis. This is due to the fact that images are large arrays of values. When we want to compare an object to a large database of objects we need extensive computation. These methods are especially not feasible when we want to use several images, like video. We are therefore looking for methods that do not hold a representation of the real world, but only focuses on the given image. Some of these techniques exist, the edge detection for instance. But they fail to properly track an object in real-time. Other techniques hold internal representations of the position of the camera in the world and of the positions of certain objects, but in most videos these coordinates are not known. The focus of our implementation is that it must be fast enough to process video in real-time.

In this report we are therefore going to use the colour distinctiveness of objects. We will create histogram models of the objects' colour space and compare these to the subsequent images in a video. This method, when properly implemented, is very fast and works in a variety of domains. However there are also several drawbacks to this algorithm. The main questions in this report are: What are the drawbacks of this method? Can this method be used in all domains? If not, in what domains will this method fail?

In section 2, we will discus the methods that we used. Section 3 briefly describes the implementation. Section 4 describes the results and in section 5 we will discuss drawbacks and improvements.

2. Method

In this section, we will discuss how the mean-shift tracker is constructed.

Selection method

Inner box
To track an object in a video, some user interaction is required. The user must give the initial object in the first frame, from there the tracker kicks in. The user needs to specify an inner box. The inner box is the histogram representation of the object and thus must be chosen carefully. The histogram is constructed in the following manner. The pixel values in between the four point of the inner box are subdivided in bins. The bins hold a value for a certain channel: either Red, Green or Blue. For each pixel that holds a unique R, G, B combination, the bin is increased by a certain value.

Outer box
Once the original histogram is created, we can use the next frame of the image to compare parts of the image to the histogram. We use the Euclidian distance between the original histogram and the histogram of the region of the next image. When we scan the entire image, two problems appear. The first is that scanning the entire image takes a lot of processing time. The second is that different objects that hold a similar histogram can be detected at a location the object could not have moved to in one frame. For these reasons, we will also ask user input for the outer box. The outer box is a region to which the object can maximally move to in one frame. The tracker will only search in this region, making it much faster and without crazy location jumps in between frames. Figure I shows an illustrative example of the inner and outer box.

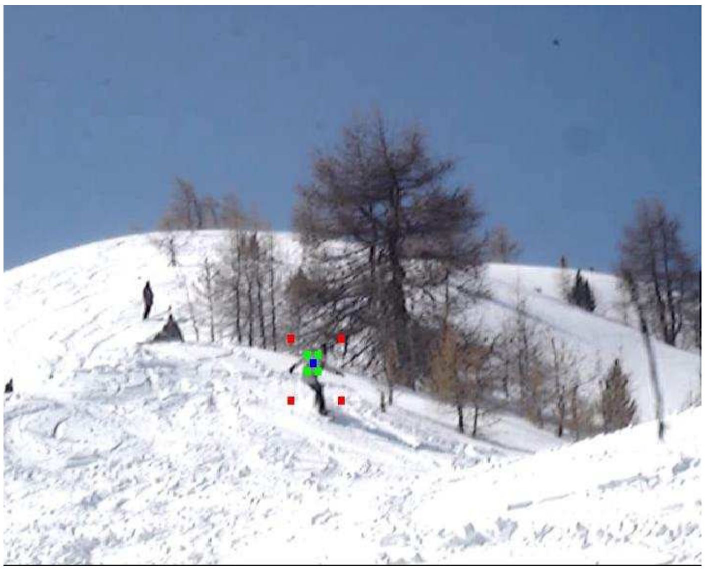

Figure I: Inner box in green, outer box in red, centre in blue.

Regular RGB versus normalised rgb

The way the original histogram of the object is constructed can influence certain domains. It is therefore important to decide upon whether we want to use regular RGB or normalised rgb. Normalised rgb works much better in most domains, because it is able to cope with shadows. However there are domains in which regular RGB outperforms normalised rgb. An example is given in the results. We will use the formulas in Figure II to calculate normalised rgb:

$$
\begin{gathered}
r = R + (R+G+B) \\
g = G + (R+G+B) \\
b = B + (R+G+B)
\end{gathered}
$$

Figure II: normalised rgb

Regular versus kernel based

We would also like to take into account the fact that an object is probably going to be close to its position from the previous frame. We will therefore introduce a scaling factor when incrementing an RGB position in the histogram. This scaling factor takes the distance to the centre into account, when the distance is small, the value will be larger than when the distance is large. The formula in figure III, gives the incremental value of a RGB position. x is the distance to the pixel from the centre, d=2 and c = pi.

$$
k(x) = \begin{cases} \frac{1}{2} c_d^{-1} (d+2)(1-x) & \text{if } x \leq 1 \\ 0 & \text{otherwise} \end{cases}
$$

Figure III: kernel based algorithm

3. Implementation

In this section I will discuss the implementation. The algorithm is written in Matlab and uses seven files. You can adjust the output directory in draw.m and the input directory in track.m. You will need the input files to reproduce the data. Due to the large result files and storage limits, I wasn't able to upload these files to my website.

Track.m
Track.m is the main method and can be invoked by track(m). Were m is a scenario number between 4 and 10. Track.m loops over all frames of the movie and handles them accordingly.

Set_original.m
In set_original.m the inner box is translated into a histogram matrix. This will be used as a reference further on.

Hist.m
Hist.m returns the histogram of an image or part of an image.

Distance.m
This method calculates the distance between two histograms.

Draw.m
This method draws the outer box, inner box and the centre in an image and then writes the new image to disk.

Kernel.m
Kernel takes as input the inner and outer box and the adjacent frames with their centres. Kernel then walks over the locations in the outer box and returns the centre of the best match.

Newcoordinates.m
Takes an outer box an inner box and the shift between the centres of two adjacent frames and then calculates the new coordinates.

Border checking

When the tracker starts moving around, we need to keep it away from the borders. A border checker is also implemented.

4. Results

In this section we will produce our results. We have tested the tracker in three different domains. The videos of the results can be viewed at: http://www.kortec.nl/onderzoek and http://home.student.uva.nl/sanne.korzec/. In some domains, the tracker was not able to perform correctly.

The failed videos can be viewed here: http://student.science.uva.nl/~skorzec/

Soccer domain – easy environment

We started out by testing the tracker in an easy environment. In this example, there is a clear difference between the foreground and the background. There are no other players with the same colour model in the vicinity. The player gets partially occluded by a player of another team, but there is still enough of him to be detected.

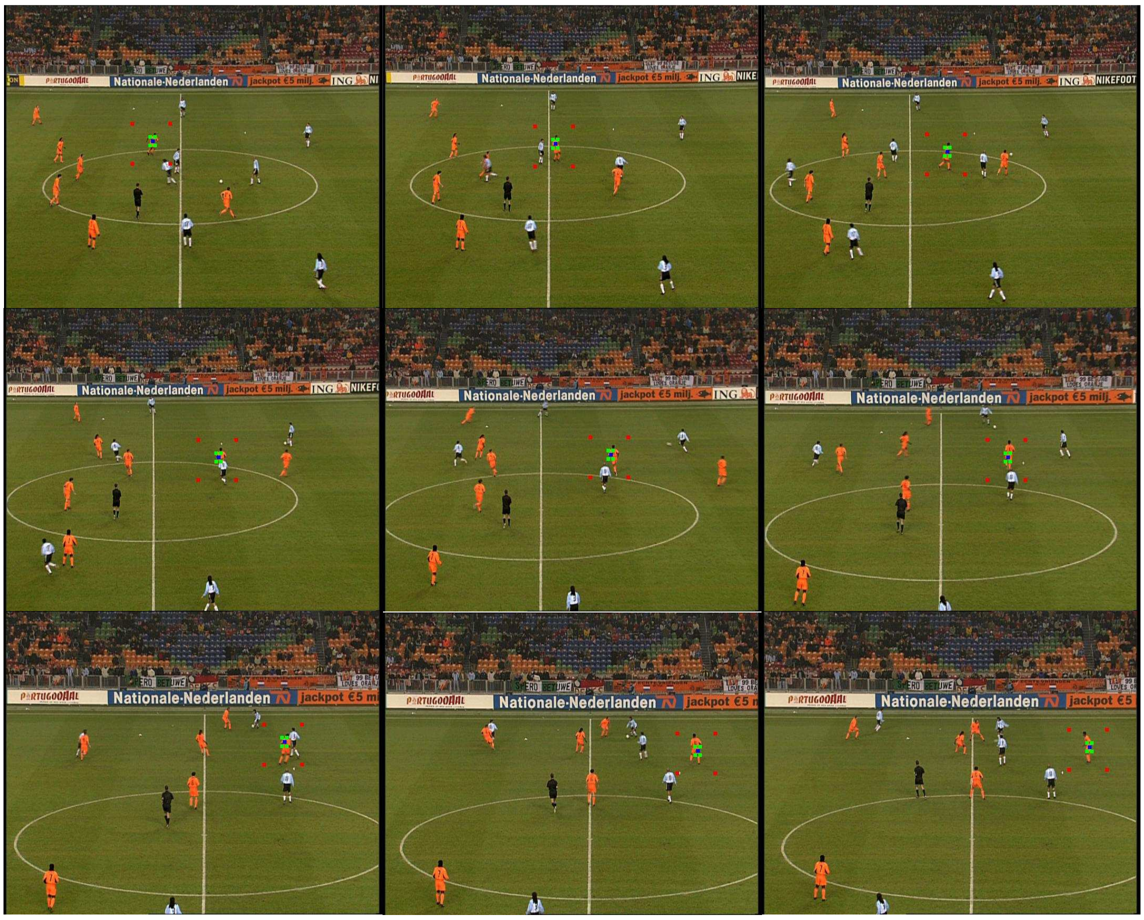
Figure IV: Soccer match, easy environment

Soccer domain – difficult environment

In our second example, we chose a player that is occluded by a player of the same team. Since the tracker is only based on colour models, we have a fifty percent chance of losing the correct object. Figure V shows an occlusion failure.

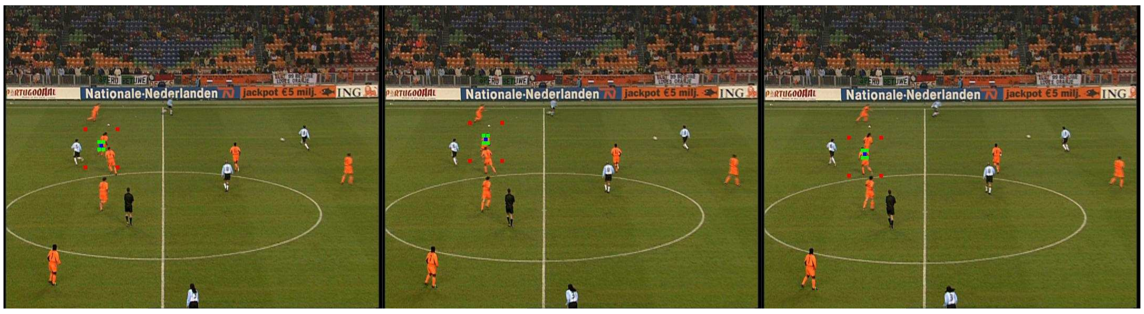
Figure V: Soccer match, difficult environment

Ramp domain – grey versus white

In this domain we tracked a grey object in a mostly white domain. The results are given in Figure VI.

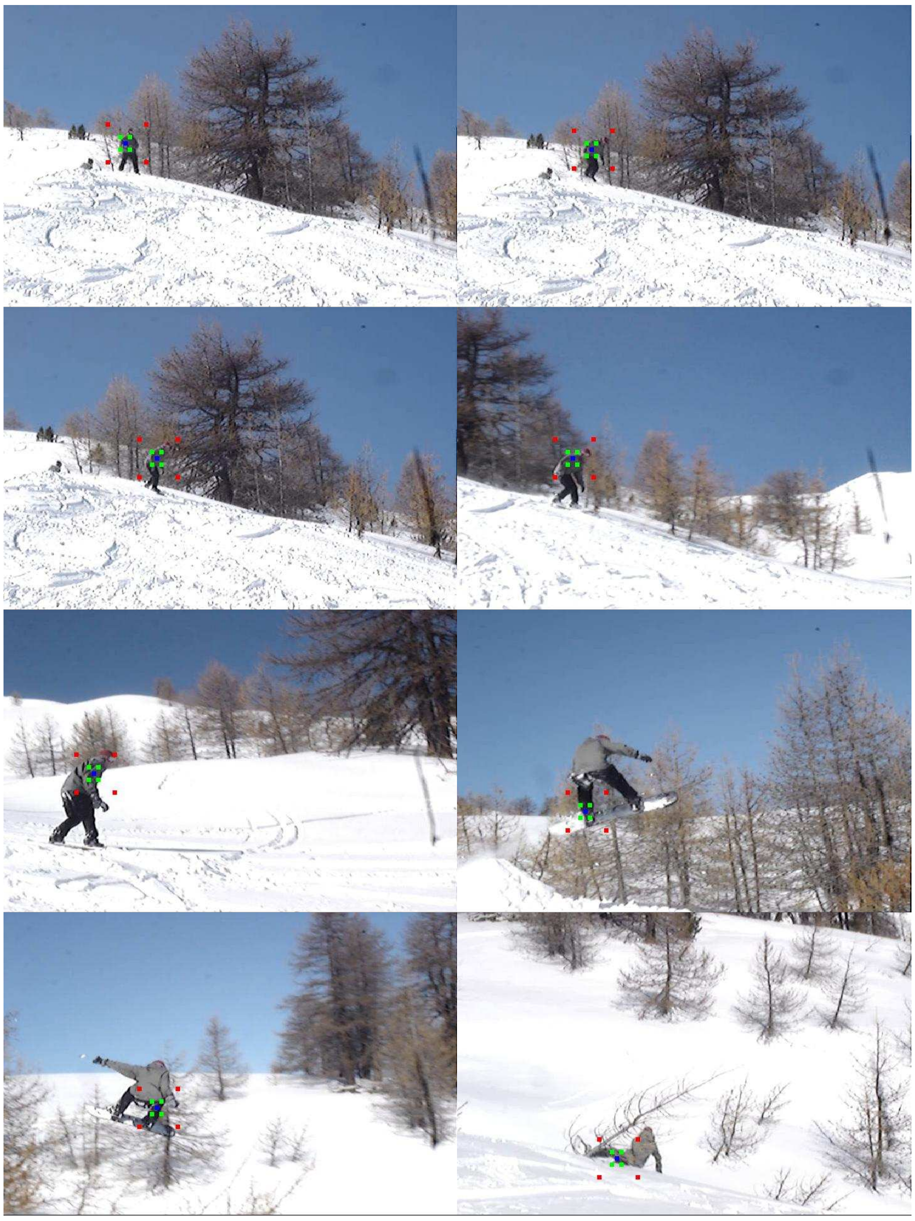

Figure VI: Snowboard Ramp, Grey on white

Ramp domain – beige versus white

When we introduce a colour that is very close to white, in respect to the colour models, the normalised rgb model, has great difficulties pinpointing the object. Changing to RGB improves the algorithm minimally. The results of normalised rgb are given in Figure VII and regular RGB in Figure VIII.

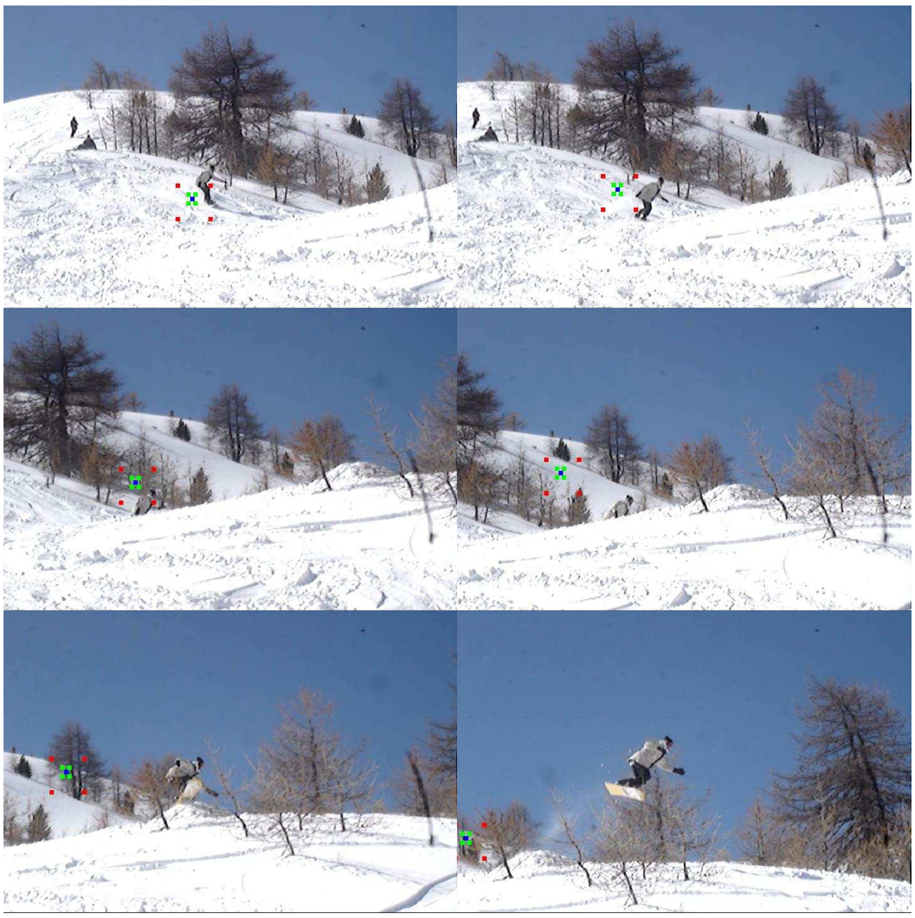

Figure VII: Snowboard Ramp. Beige on white, normalised rgb.

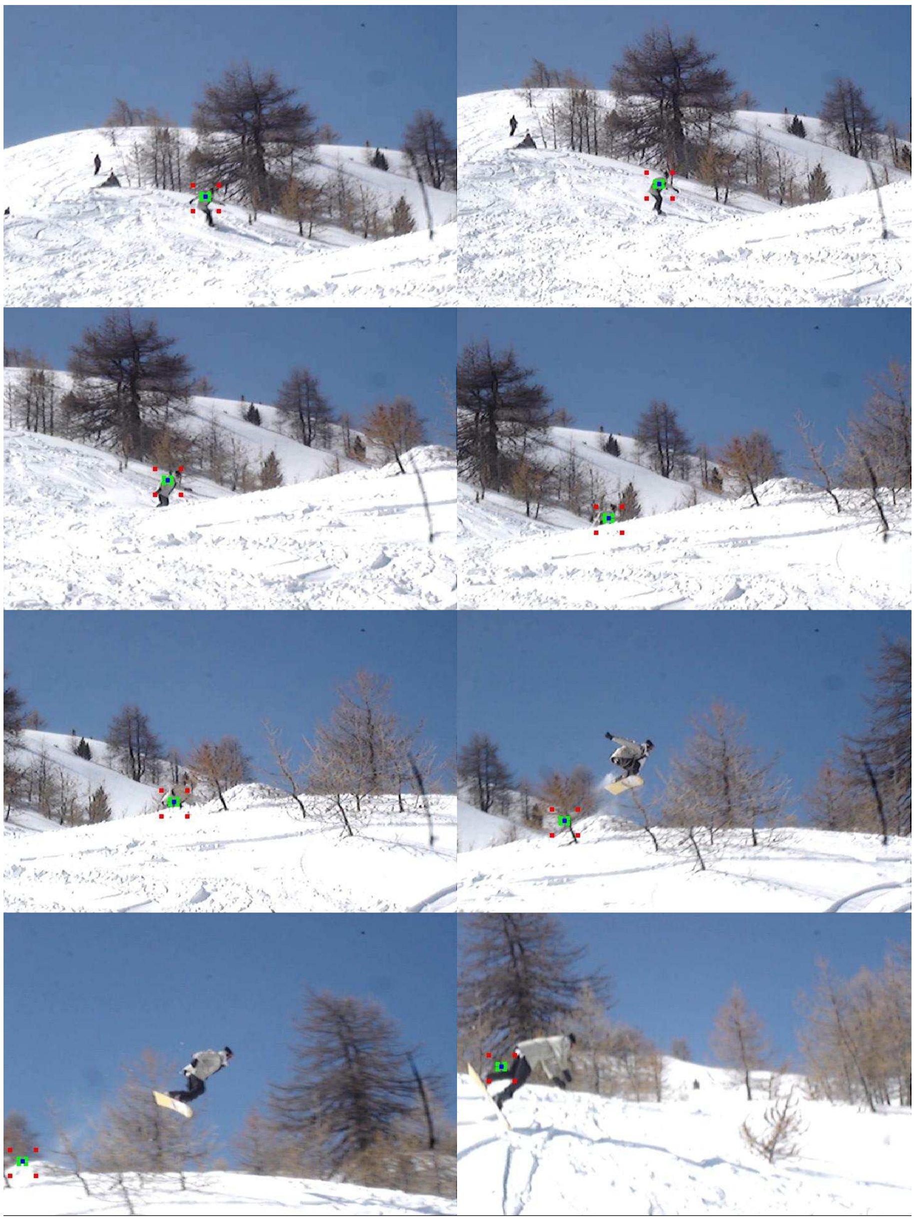

Figure VIII: Snowboard Ramp. Beige on white, regular RGB.

Woods domain – black on various colours

In the final domain, we track a snowboarder through the woods. This is a very difficult scenario, because we are tracking a black object that is constantly occluded by near black objects. The results are not garantied. Each run of the algorithm behaves differently. In Figure IX, we show a succeeded run.

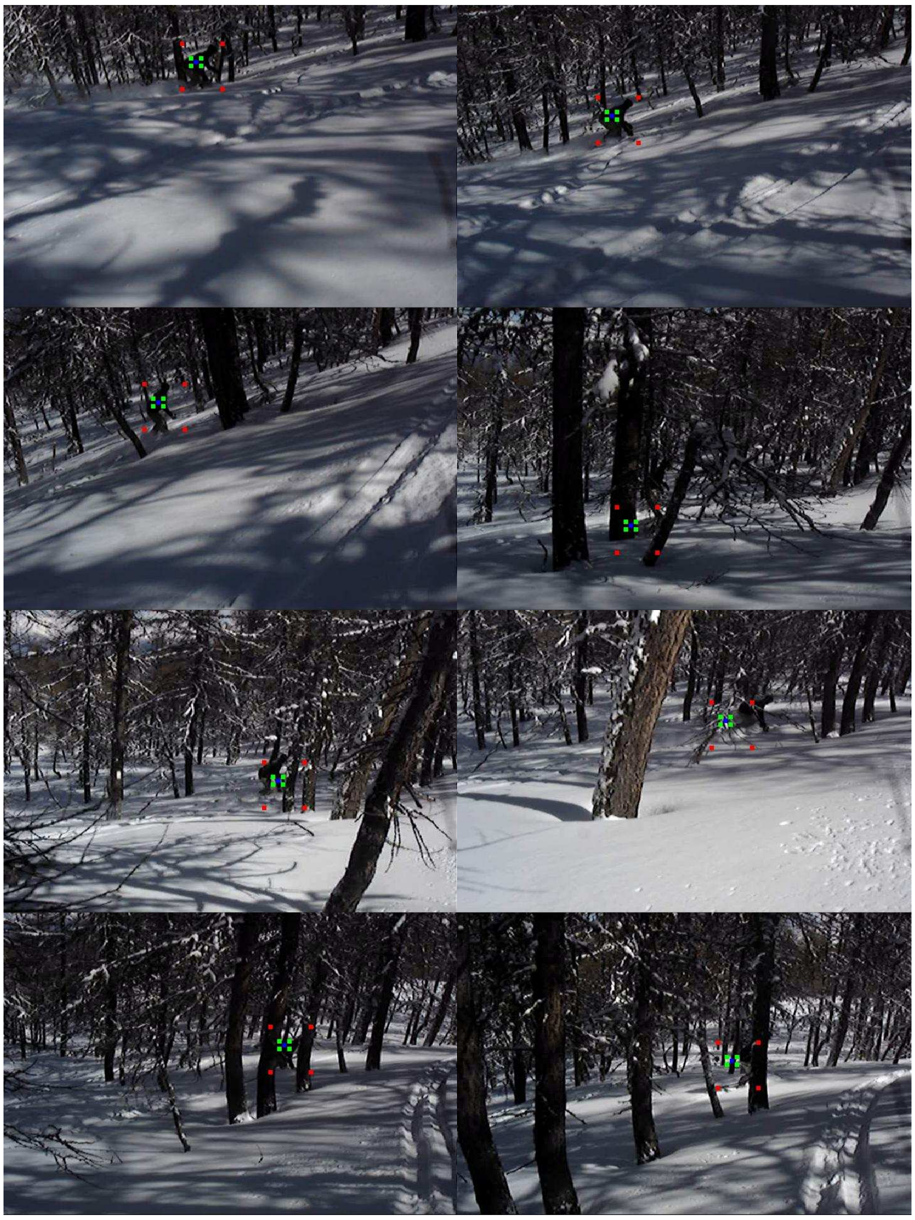

Figure IX: Through the woods, Black on various colours

5. Discussion

Common user errors

The tracker failed and succeeded on some domains when running the same example. A lot can be done to make sure the current algorithm doesn't fail. I will describe some common errors and tips that will improve the results of the current mean-shift tracker.

- When the user selects the initial histogram, it is of great importance that pixels of the inner box are more or less the same value. When parts of the background or other colours are also selected, the original histogram holds less uniqueness.

- When selected the outer box, it shouldn't be too big or too small. When it is too big, other objects have more chance of being selected. But when it is too small, the object of interest can move outside of the box, resulting in a loss.

In Figure V, we know the player is going to be occluded by his team-mate. It would improve the results if we would adjust the outer box. But then we are performing biased research. We are searching for methods that work under a broad variety of examples, without prior knowledge.

Conclusion

We have seen some interesting results. In some situations our tracker gives good results and accomplishes this in near real-time computation. But when we push the algorithm to its limits, we see that it can fail just as easily. One of the main drawbacks of the method is that the algorithm is only colour based. When we introduce a domain in which the background holds the same colour, the method will surely fail. Also when another object of the same colour model occludes the object of interest, we will have a big probability of failure. Therefore this algorithm can only be used in certain types of domains.

Improvements

Most of the errors we have encountered have to do with the colour model. If we want to create a model that is more robust in domains were the object colour is not unique, we need to introduce more features. Many features can be used, edge detection for instance. One could also take into account the direction an object is moving. This off course also has a downside, when the object changes direction.

References

- Multimedia information retrieval, T. Gevers, university of Amsterdam, 2006.
- Computer Vision, L. Dorst, university of Amsterdam, 2005.
- http://staff.science.uva.nl/~gijsenij/mir/kerneltracking.pdf

The results movies can be found here

- http://www.kortec.nl/onderzoek/
- http://home.student.uva.nl/sanne.korzec/
- http://student.science.uva.nl/~skorzec/

Some input files can be found here

- http://staff.science.uva.nl/~gijsenij/mir/data/

All sites were last accessed on 31-12-2006.