Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: e0026641@student.tuwien.ac.at
e0026761@student.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-105 October 24, 2006

Motion Tracking with Normalized Cut and Minimum Spanning Tree

Stefan Fiel and Paul Guerrero

Abstract

We propose using the Normalized Cut method for motion tracking (J. Shi and J. Malik . "Motion Segmentation and Tracking Using Normalized Cuts". In IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8), pages 888 - 905, 2000) on regions in frames that were identified by a Minimum Spanning Tree (MST) method in a pre-proccessing step. The purpose of the pre-proccessing step is to reduce the spatial resolution without losing important image information. An energy function, based on some selected properties of regions, is then calculated for each pair of regions in a fixed number of consecutive frames. This energy function represents the similarity of two regions. Based on this similarity, the Normalized Cut method is used to identify salient groups of regions. Finally, corresponding salient groups of regions of neighbouring sets of consecutive frames are found. We show on different experiments, how pre-segmentation can help to reduce computation time by reducing the spatial resolution of the input frames. We tested different methods to reduce temporal resolution to gain an additional speedup.

Contents

1 Introduction

Universal methods of motion segmentation aim at partitioning a sequence of images into regions that match an intuitive partitioning of the sequence as done by a human observer as close as possible, even though only limited information is available. This problem is one of the central problems of computer vision and several methods have been proposed to tackle this difficulty. An additional difficulty is that motion segmentation usually needs to consider large amounts of information making runtime a major issue in applications that have to process many frames per second. Various methods have been proposed to cope with those difficulties, usually falling into two main categories:

- feature-based methods, and
- region-based methods

Feature based methods use a template object and try to match it to regions in the sequence (e.g. [1]). Region-based approaches usually use intensity, colour, texture and motion information or a combination of those. This information can either be used in an sequential or integrative process. Sequential approaches use multiple stages each using one type of information and iteratively refine the result [2] while integrative approaches combine all available information in one step [3]. In this document we propose a two step region-based approach. The first step uses only local information to reduce the complexity of the problem by removing redundant information, the second one being an integrative step combining all available information in an energy function (called similarity function).

Work related to our proposed method includes [4], [5] and [6]. In [4] motion information is used to compute a prediction template which is then corrected using static watershed segmentation. In [5] colour segmentation is used to determine stable segments which are then merged using motion information. In [6] Gelgon et.al. propose to use spatial segmentation to construct an initial region adjacency graph and to merge vertices using motion information. The resulting graph is then used to predict the partition of the next frame.

Another method for image segmentation proposed by Shi and Malik is a method based on the Normalized Cut (Ncut) criterion. Normalized Cuts was originally used for image segmentation [7] only, but in the paper [8] Shi and Malik proposed a method for using normalized cuts for video segmentation. Using a motion profile which shows the motion probability distribution of each pixel the regions with similar movement can be identified. The computation time and storage requirements of comparing the motion profiles of pairs of pixels make real-time application (multiple frames per second) of the method difficult. To reduce the number of vertices in the graph, Shi and Malik [8] propose subsampling every image before processing it. When subsampling to a very low resolution this gives great performance improvement but a lot of information is lost. Balancing performance versus information loss may be difficult.

We propose a combination of the original Ncut algorithm with a pre-processing step to reduce the spatial resolution without losing important information. This combination results in a speed-up of the entire algorithm. Usually in most images there are large regions of pixels that belong to the same salient region and have only small interior intensity

variations and are thus easily identified. To combine these pixels into one region a cautious segmentation algorithm is necessary. We will argue that pre-segmentation is a good method for achieving such a reduction of the spatial resolution. In our implementation, which we will call MSTNcut[1], we use Minimum Spanning Tree (MST) method proposed by Felzenszwalb et.al. in [9]. Also, in videos without rapid motion it is not necessary to segment each frame we therefore propose a reduction of temporal resolution. We will discuss the effects of several methods to reduce the temporal resolution.

The document is organised as follows. Section 2 briefly explains the original Ncut method. Section 3 describes our slightly modified version of the Ncut method. In Section 4 we detail our implementation. In section 5.1 we present the results of our experiments. We conclude in Section 6. Terms we introduced in this document are defined in Appendix A.

1.1 Breakdown

Section 1 to 5.1 is done jointly by Paul Guerrero and Stefan Fiel. Section 5.2.1 is done by Paul Guerrero and Section 5.2.2 by Stefan Fiel.

2 Normalized Cut

The following section is a short summary of the Normalized Cut criteria proposed by Shi and Malik in their papers [7] and [8].

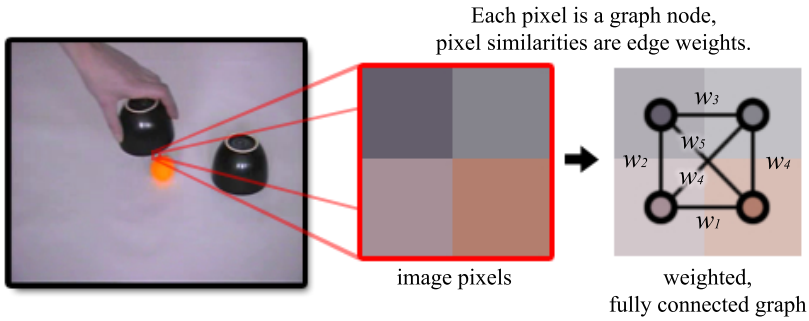


Figure 1: Constructing the graph from an image.

Images can be represented as graphs $G = (V, E)$ (see Figure 1) with each pixel being a vertex $v \in V$ and edges $e \in E$ connecting pairs of neighbouring vertices (e.g. using 4- or 8-neighbourhood). A weight function $w_{sim} : E \rightarrow \mathbb{R}^+$ is defined assigning each edge the similarity between the two corresponding pixels. For example function of the intensity difference of two pixels can be taken as edge weights.

$$w_{sim}(v_1, v_2) = exp^{\frac{-|I_{v_1} - I_{v_2}|_2}{\delta}}$$

[1]Ncut with MST pre-segmentation step.

where I_{v_1} and I_{v_2} are the intensities of the two pixels. For the δ parameter we usually use 0.1. We assign a weight of 0.0 to pairs of pixels that are not neighbours. of the pixels in an image using the similarity function, like the distance weighting function described by Shi and Malik in [7]. The vertices of this weighted graph can then be partitioned into two disjoint sets, such that $A, B, A \cup B = V$ and $A \cap B = \emptyset$, by cutting the edges connecting the two sets. The total weight of the cut edges is an indicator for the quality of the partition.

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

This criterion tends to result in small partitions since the amount of edges connecting the two resulting sets is usually minimal [7]. To make the partitions more independent of the size Shi and Malik proposed the Normalized Cut criterion which is defined as follows [7]:

$$Ncut(A, B) = \frac{cut(A, B)}{asso(A, V)} + \frac{cut(A, B)}{asso(B, V)}$$

Where $asso$ is the total weight of all connection of all vertices from A to all vertices in V.

$$asso(A, V) = \sum_{u \in A, t \in V} w(u, t)$$

$Ncut(A, B)$ is less biased with respect to the region size than $cut(A, B)$ [7].

To segment video sequences with this method, the naive approach would be to add the pixels of all frames to one graph but this is clearly not practicable. A video sequence consisting of 100 frames with a size of 320×240 pixel would consume approximately 54 terabyte of memory (using a fully connected graph with 1 byte per edgeweight). The other extreme, to process each frame separately lacks important information of how frames connect to each other. Instead Shi and Malik propose in [8] to only consider a frame window in every step which is moved over the whole sequence to incorporate information across several frames without overly stressing resources.

3 Dendogram of Normalized Cuts with Pre-segmentation

A disadvantage of the method described by Shi and Malik in [8], when using a fully connected graph, is the prohibitive space and computation time requirement. They stem from the fact that the incidence matrix of the fully connected graph has to be constructed and evaluated. To reduce the number of vertices in the graph, vertices can be used to represent disjoint regions covering the image instead of single pixels. These regions are constructed in a pre-segmentation step. The several methods for finding these regions will be discussed in section 5.2.1.

Figure 2 illustrates the method we propose for region-based tracking. Since we have three segmentation steps we use a specific name for each type of segment resulting from one of the segmentation steps. The segments resulting from the pre-segmentation will be called elements, the segments resulting from the Ncut dendogram will be called initial

segments and the segments resulting from the merge step will be called final segments. In the following, we describe each step in detail. Pseudo code for dendogram construction and the merge step is available in Appendix B.

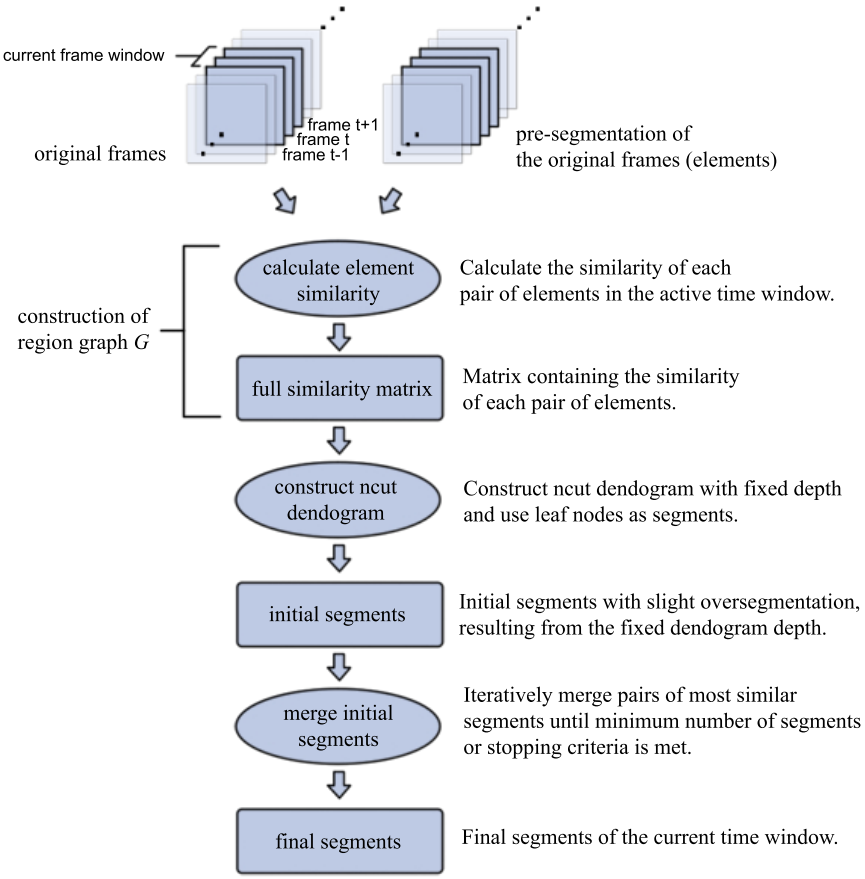


Figure 2: Flowchart illustrating each step of the method.

All following steps consider only the information of one frame window. A frame window consists of three consecutive frames of the sequence.

As result of the pre-segmentation we get the segmentation of each of the frames of the current frame window into regions (called elements). All elements of the current frame window form the vertices $v \in V$ of an undirected, fully connected graph without self-loops $G = (V, E)$. The weights of the edges $e \in E$ between two vertices represent the similarity of the two elements. This graph will later be used to calculate bipartitions using the Ncut criterion as described above. To determine the weight of an edge connecting two vertices we define a similarity function $w_{sim} : E \rightarrow \mathbb{R}^+$ that assigns a value to each edge depending on information collected from the element, such as intensity, colour, intensity variation, etc. represented by the vertex. No special calculations are done for regions belonging to different frames. It should be noted that one problem of using pre-segmentation is that the elements in oversegmented regions will vary widely with small image variations. Therefore properties like shape, size, roundness, convexity, etc. can not be used as a measure of similarity of two elements. The similarity of each pair of vertices is stored in

a weighted incidence matrix, the similarity matrix.

The Ncut is calculated on the graph G to find a bipartition of the image. For finding the Ncut one can use the method described by Shi and Malik which involves calculating the eigenvalues of the slightly transformed weighted incidence matrix (for details of the eigenvalue method for calculating Ncut we refer to [7]). The eigenvector with the smallest eigenvalue has eigenvalue 0 and cannot be used for partitioning the graph. The remaining eigenvectors are indicator vectors for separating the vertices of the graph G into two groups. Figure 3 gives an example of an eigenvector with dimension 85. Each component of this 85-dimensional Vector is plotted along the x-axis, the value of each component is shown on the y-axis. Since the eigenvectors usually take on continuous values, a splitting point, that separates vector component values indicating one half from values indicating the other half, has to be found. Zero is usually a good value for the splitting point, but to achieve better results, the splitting point with the best Ncut value of a range of possible splitting points is used. The eigenvectors are approximation for the optimal Ncut solution, the second smallest one being the closest approximation. With second smallest eigenvector one can divide vertices of the graph G into two groups to form two new subgraphs G_1 and G_2. The edges from vertices in G_1 to vertices in G_2 are removed.

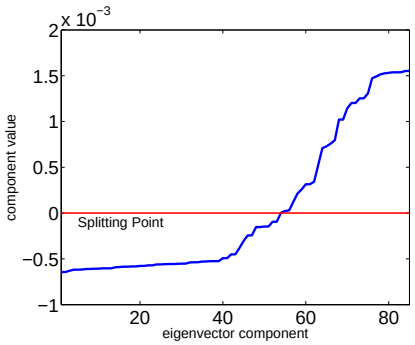


Figure 3: An example of one of the eigenvectors used to bipartition a region.

The third, forth, etc. smallest eigenvectors represent alternative partitions and can be used to further partition both halves. Another possibility is to calculate a new Ncut for each of the subgraphs. The first method requires less computation time since most eigenvector solvers obtain the n smallest eigenvectors in the process of finding the smallest one. The second method is more precise because the optimal partition of the remaining graph is approximated as close as possible.

The recursive partition of the graph G can be represented as a dendogram with G being the root vertex and G_1 and G_2 being the child vertices, see Figure 4. The root of dendogram is the original image of a video used for testing. This image was partitioned into two halves using the eigenvector with the second smallest eigenvalue. The left branch contains one half of the original image (background, cup and ball), the right branch the other one (cup and hand). These halves are further partitioned resulting in four leaves.

The depth of the dendogram depends on the number of partitions required. Some eigenvectors may lead to partitions that are inadequate but these should not be excluded as their subgraphs may still contain valid partitions. The final dendogram is a 'K-way'

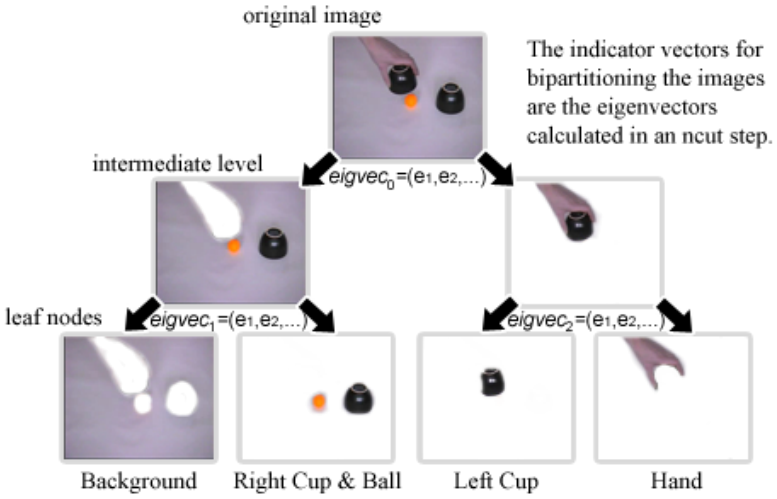


Figure 4: Construction of the Ncut dendogram.

cut of the graph G, representing an oversegmentation of the original image. The leaf vertices represent the initial segments.

It should be noted that the initial segments may encompass elements from every frame of one frame window. The dendogram does not take the frame number of each element into account.

To reduce the oversegmentation of the image an additional post-processing step to merge the affected initial segments into final segments is needed:

1. Simple Merge: In this naive approach (based on the "greedy pruning" of Shi and Malik [8]) the similarity of every segment to every other segment is calculated and pairs are iteratively merged until a minimum number of segments is reached or until the maximum similarity of each pair is smaller than a specified threshold.

2. Subtree Merge: In this approach, we merge only segments of one half of an inadequate bi-partition with segments of the other half. The criterion for determining whether a partition is inadequate depends on the application. This method prevents segments from being merged if a valid Ncut bi-partition in the dendogram identified them as belonging to different segments, therefore strengthening the influence of the Ncut step on the final segmentation.

It should be noted that the larger the oversegmentation, the larger the influence of the merge step on the final segmentation (and the smaller the influence of the Ncut step). To achieve the best results, one should try to keep the oversegmentation as low as possible while still segmenting all relevant image regions.

4 Implementation of Dendogram of Normalized Cuts using Pre-segmentation

In our implementation we use a MST segmentation method propsed by Felzenszwalb et.al. [9] in the pre-segmentation step (Figure 5 shows an image and its MST segmentation). We chose this method because only few frames showed undersegmentation and the low computation time needed. Any method could be used in this step, as long as the frames will be oversegmented rather than undersegmented. This is important, since the following steps are not able to split up segments resulting from this pre-segmentation.

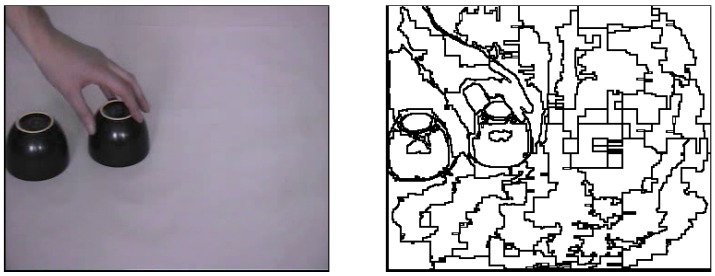


Figure 5: An image and the result of segmenting it with the MST algorithm [9].

To segment the complete video sequence using Ncut, only $\pm$ 1 frames centred on the current time step t are considered for computing the graph partitions. For the next time step $t+1$, the first frame $t-1$ is dropped and the frame $t+2$ is incorporated in the frame window as shown in Figure 2.

As described in the last section, we use the segments (as mentioned, we call these elements) found by the MST method in all the frames of the frame window, as vertices for a fully connected, undirected graph without self-loops since the similarity of an element to itself is alway the maximum value. To determine the edge weights, we define a similarity function that assigns a positive value to each pair of elements. While developing our MSTNcut algorithm , we experimented with different similarity functions. First we tried using the RGB colour space distance which showed poor performance on the videos we worked with. The colour space distance between objects and their background was in most cases larger than the distance between two regions belonging to the same object (e.g. shadowed regions), resulting in the MSTNcut algorithm not being able to find the correct partition. Next, we experimented with comparing intensity variation and colour histograms of the elements. This approach showed better results, but shadowed regions were still split from the object in many frames.

Since shadowed regions have soft transitions to brighter object regions and objects usually have sharper transitions to the background, we tried to use an edge detector to solve the problem with the shadows. The edges of the elements from pre-segmentation do not coincide with the real object borders, usually there is a small element covering the real object border. Figure 6 shows a pre-segmentation of an image. The white lines are the element borders. The small elements covering the real object borders are encircled. Additionally this method was sensible to noise and video compression artifacts and since the edges in the video were blurry the results were unreliable.

Figure 6: Elements covering object borders make edge detection difficult.

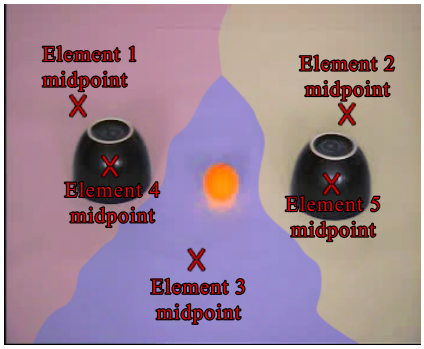


Figure 7: A problem when using midpoint distances.

A criterion, originally used by Shi and Malik [7], used in conjunction with colour space distance for calculating the similarity of two elements is the distance between the midpoints of these two elements. This criterion acts on the assumption that elements that are close to each other are more likely to belong to the same object. Figure 7 illustrates the problem that occurs when using midpoint distance of the elements resulting from pre-segmentation. As one can see, the midpoint positions depend on size and shape of the elements, e.g. the distance between element 1 and element 2 in Figure 7 is large even though they are adjacent and belong to the same object. Element 4 and 5 are clearly two different objects, but the distance between them is smaller. Midpoint position distance is inadequate for elements with varying shapes and sizes.

The discrete Hausdorff distance [10] can be used to get a more accurate measure of the proximity of two elements. But the Hausdorff distance is not well suited for our implementation, since two elements may belong to different objects and still be close to each other (e.g. the two cups shown Figure 5) and this measure would keep them from being separated.

This lead us to consider the direct neighbourhood of an element instead. Based on the assumption that elements sharing a relatively long, smooth border are likely to belong to the same object, we experimented with another criterion, using the sum of the colour space differences along the border between the two elements relative to the element's pixel area. This measure increases as the length of the border relative to the element's area increases. We intended to use this criterion to keep the elements covering the borders

of the objects (the same elements causing problems with the edge detector) from being treated as separate objects. Problems occurred when dealing with small elements attached to large ones as the similarity between those objects is always low, even when belonging to the same object.

In our final implementation, the similarity function only uses the colour information of each element and calculates the IHLS [11] colour space distance. We transform every pixel to the IHLS colour space and calculate the mean IHLS colour space coordinates for each element. We define the final similarity function as follows:

$$w_{sim}(v_1, v_2) = |\mathbf{x}_{v_1} - \mathbf{x}_{v_2}|_2 \tag{1}$$

where $\mathbf{x}_{v_1}$ and $\mathbf{x}_{v_2}$ are the IHLS colour space coordinates of the elements v_1 and v_2. The coordinates consist of lightness and the two-dimensional hue vector. For better results, we enhance the saturation before calculating the similarity function. Optionally, the saturation is stretched according to a specific function (see Figure 8), which keeps unsaturated regions and enhances the saturation in more saturated regions. Any function with similar effects can be used.

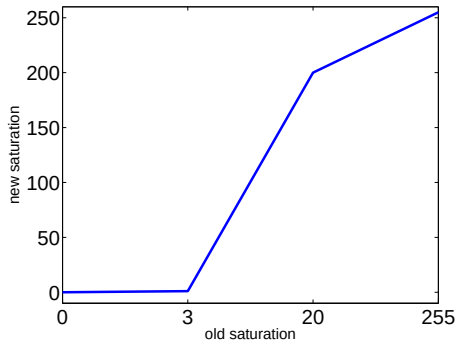


Figure 8: The saturation stretching function.

The similarity values are stored as weights in the adjacency matrix of the graph, the similarity matrix.

The dendogram of recursive cuts is constructed with fixed depth. In every vertex of the dendogram we divide the graph by calculating a new Ncut on the similarity matrix of the graph and using the second smallest eigenvector as an indicator vector for the bi-partition. The leaf vertices of the fully constructed dendogram represent the initial segments. For solving the eigenvalue problem we use the ARPACK open source library [12], which is available for download on the website of the RICE University, Houston, Texas. The matrix supplied to ARPACK should not contain too many zeros which are not in band format, otherwise the Arnoldi iterations [13] will take too long to converge, or not converge at all.

As mentioned above, some eigenvectors obtained in the process of constructing the dendogram result in partitions leading to oversegmentation. Since we use a dendogram with fixed depth, most of these eigenvectors are calculated when trying to split a homogenous region in the deeper levels of the dendogram. To avoid splitting these regions, we search for a stopping criterion. We experimented with the eigenvalues belonging to the

eigenvectors, but found out, that there were always frames were the quality of the cut and the magnitude of the eigenvalue were not related. Next we tried to calculate the Ncut value of each bipartition and tried to find a threshold to discard partitions with high value. We noted, that Ncut values of partitions of different image regions are not comparable.

Since eigenvectors leading to inadequate partitions may be followed by eigenvectors at deeper levels leading to sound partitions, we chose to use the Simple Merge as described in Section 3. Partitions are removed by iteratively merging the most similar segments (by the similarity measure $w_{sim}(v_1, v_2)$ in Formula 1) until a minimum similarity threshold is reached. Optionally, the saturation can be stretched while calculating the segment similarity, as shown in Figure 8. For our implementation, we assume that all objects are spatially connected in every frame; only taking colour and position information into account, it is not possible to distinguish between two unconnected elements belonging to the same objects and two elements belonging to different objects. Segments consisting of multiple unconnected components are split up. No effort is done to separate occluded and occluding object of similar colour.

To identify the corresponding segments in consecutive frame windows fw_t and fw_{t+1}, we calculate the area of overlapping pixels of segments of window fw_t and segments of window fw_{t+1} (see Figure 2). Segments of fw_{t+1} are considered corresponding those segments of fw_t, to which they have the maximum percentage of pixel overlap. This can be done efficiently, we only need to sum the area of every element. Elements in corresponding frames of the successive frame windows must be identical. The number of frames passed since a segment has first been recognized (called age of the segment in our implementation) is also used in the calculation. The longer a segment exists, the more likely it is that a segment from a new frame window is assigned to it:

$$nop(seg_i, seg_j) = \frac{2 * |p_{seg_i} \cap p_{seg_j}|}{|p_{seg_i}| + |p_{seg_j}|} \tag{2}$$

$$fw_{sim}(seg_i, seg_j) = \begin{cases} nop(seg_i, seg_j) & (age(seg_j) \geq adultAge) \\ nop(seg_i, seg_j) * \big(1 - age\_weight * (1 - \frac{age(seg_j)}{adultAge})\big) & (age(seg_j) < adultAge) \end{cases}$$

$$with\ seg_i \in fw_t\ and\ seg_j \in fw_{t+1}$$

p_{seg_i} is the set of pixels in segment i, p_{seg_j} the set of pixels in segment j, $nop(seg_i, seg_j)$ is the number of overlapping pixels of segments of consecutive frame windows fw_t and fw_{t+1} relative to the pixel area of both segments (Figure 9). This value is then weighted with the age of the segment from fw_t to get the frame window similarity $fw_{sim}(seg_i, seg_j)$ of these segments. *adultAge* is a constant describing the age after which segments should not be lost if possible, it can be any positive integer. *age_weight* specifies the influence of the age on the overall frame window similarity and takes on values from 0.0 to 1.0 (for the parameter values used in our experiments refer to Table 2).

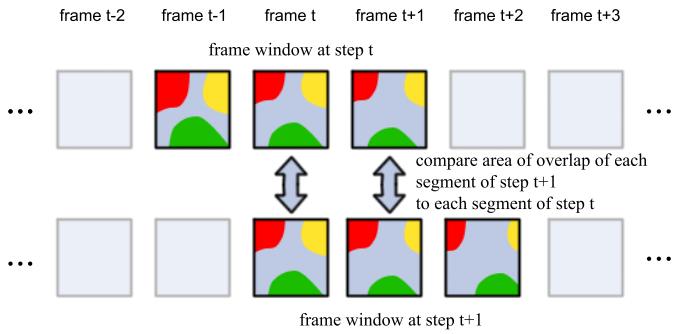


Figure 9: Regions in different frame windows through area of pixel overlap

Since we make no assumptions as to the shape and size constancy of elements, finding corresponding elements in consecutive frame windows is too unreliable to be of use. Thus, it is not possible to use the motion information of an element.

5 Experiments

Three videos were used to evaluate the MSTNcut algorithm with respect to

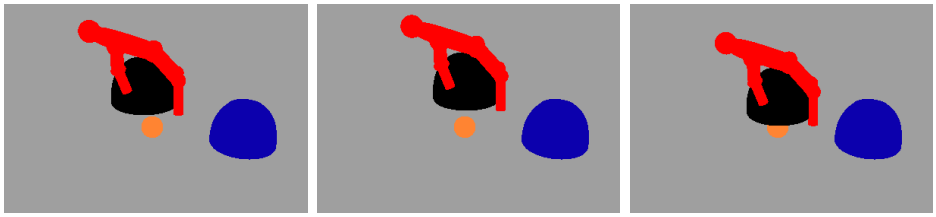
- spatial resolution
- temporal resolution

A simple synthetic video showing two cups [14] (Figure 10(a)), a ball and a hand moving the cups, the same movie with real objects [15] (Figure 10(b)) and another synthetic video of a person moving down a corridor (Figure 10(c)), a part of a video taken from the muscle benchmarking project [17]. The synthetic video has 201 frames with a size of 400×275 pixels. Pre-segmentation is necessary when using this video material, taking every pixel of a video with a frame size of 400×275 pixels as vertex of fully-connected graph would result in 110 000 vertices per image, 330 000 vertices per frame window and about $109 * 10^9$ edges. For the other videos see Table 1.

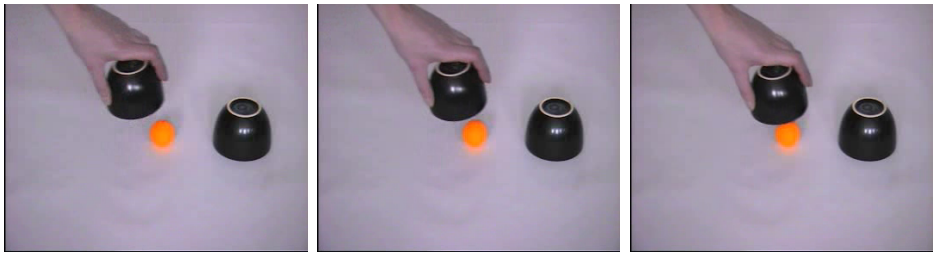
Videos	Synthetic	Cup	Corridor
size	400×275	352×288	640×480
number of frames	201	796	356
elements per frame	$4 - 6$	$65 - 90$	$200 - 230$

Table 1: Some properties of the videos.

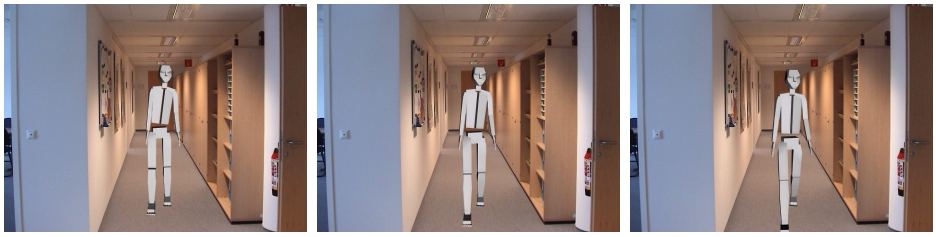
We chose the first Video 10(a) to benchmark the MSTNcut algorithm on an optimal pre-segmentation. The second video 10(b) illustrates difficulties encountered with many motion segmentation algorithms: large areas of homogenous colour, dark, shadowed areas,



(a) Synthetic Video



(b) Cups Video



(c) Corridor Video

Figure 10: Some frames of the tested videos.

soft transitions between objects and bright highlights. The third video contains motion towards and away from the camera, as well as non-rigid objects.

All videos used were compressed causing degraded video quality (e.g. blurry edges, motion artifacts, etc.).

5.1 Results

We applied our implementation of the Ncut segmentation, with MST [9] as pre-segmentation and no reduction of the temporal resolution, to the three videos in Table 1.

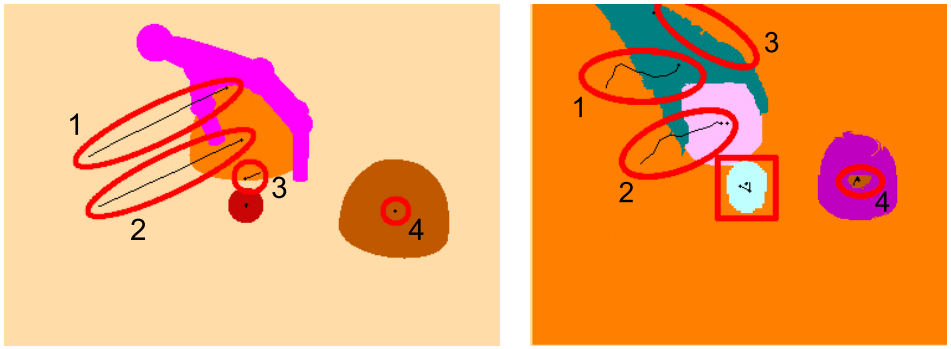
First we calculated the center of gravity midpoints of each final segment then we evaluated the videos by calculating the difference of these midpoints to the midpoints of the reference objects. For the first and third video ground truth was available. In the second video (Cups Video) reference midpoints were created by manually defining a bounding box for each object and calculating the trajectory of the center of gravity of each object by using [18].

Table 2 shows which parameters were used to segment each video. "Dendogram depth" is the maximum depth of the dendogram. "Merge threshold" is the minimum similarity two segments must have to be merged. The larger the value the more segments are merged resulting in less final segments. Possible merge threshold range from 0 to 500, recommended values are around 25 - 40. "Similarity saturation stretch" and "Merge saturation stretch" indicate whether the saturation was stretched when calculating the similarity matrix and respectively whether the saturation was stretched in the merge step. "Segment age weight" defines the percentage of the influence of the segment age on the frame window similarity. Possible values are in the interval 0.0 to 1.0. We used the value 0.5 for the corridor video, because the elements constituting the person are not stable and the age weight provides a means to increase the constancy of the final segments over time.

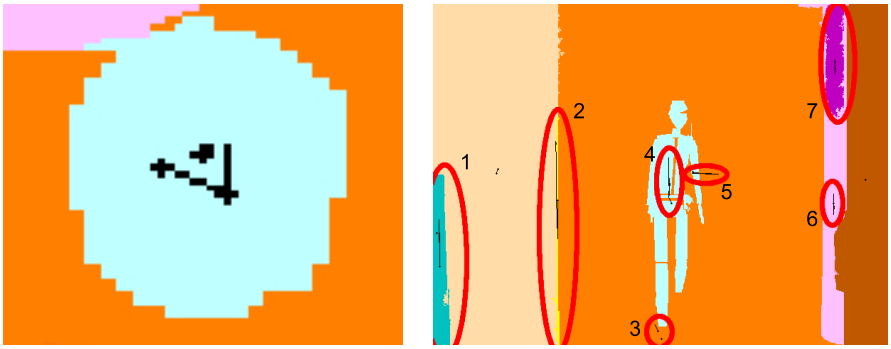
parameter	Synthetic	Cup	Corridor
Dendogram depth	4	4	3
Merge threshold	30	30	35
Similarity saturation stretch	on	on	off
Merge saturation stretch	off	off	on
Segment age weight (*age_weight*)	0.0	0.0	0.5

Table 2: Parameters used for segmenting the videos

The first two videos were segmented with a dendogram depth of 4. Since the third video contains less objects the depth was reduced to 3. Only the second smallest eigenvector was used for the segmentation of all videos and a new Ncut was calulated at every dendogram vertex. As a pre-processing step we enhanced the saturation of the sequences before calcuating the similarity function. Additionally, the saturation was stretched according to the function illustrated in Figure 8. To keep shadows from separating from the objects, the saturation was stretched when calculating element similarity in the cups video. Shadows have the same hue as their objects and by stretching the saturation, the influence of lightness on the similarity decreases. The corridor video has more textured regions. These regions have widely varying hues. To keep the Ncut step from producing heavy oversegmentation in the textured regions, the saturation was not stretched while calculating element similarity. The merge threshold was adapted for each video. Only connected segments are considered objects.



(a) Synthetic Video (b) Cups Video



(c) Zoomed in on the Ball (d) Corridor Video

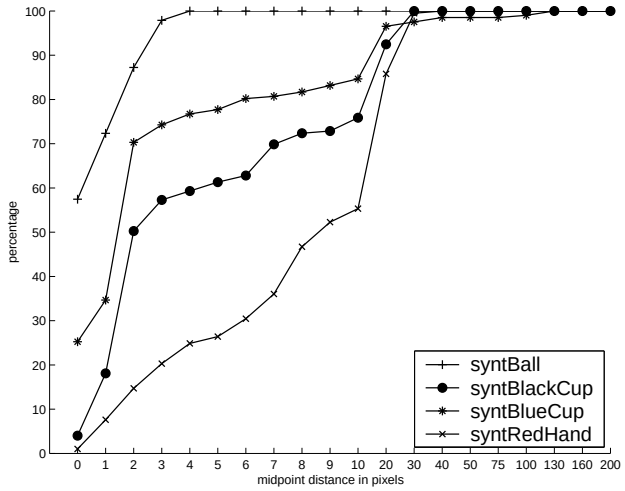
Figure 11: The results of the segmentation of the pictures above. The black lines are the midpoint position trajectories.

Figure 11 shows one frame of the resulting segmentation of each video. The black lines are trajectories of the objects obtained by calculating the center of gravity of each final segment. The Synthetic Video (Figure 10(a)) shows straight and smooth trajectories. Object outlines are sharp in this video, so pre-segmentation does not distort object outlines and midpoint position estimation is accurate (exact percentages are given later in this section). Item number 1 in Figure 10(a) is the trajectory of the Hand object, item 2 the trajectory of the Cup object, item 3 the trajectory of the Background object (which shifted its center of mass as a result of the objects in the foreground moving) and item 4 the trajectory of the second Cup object, which has not moved yet.

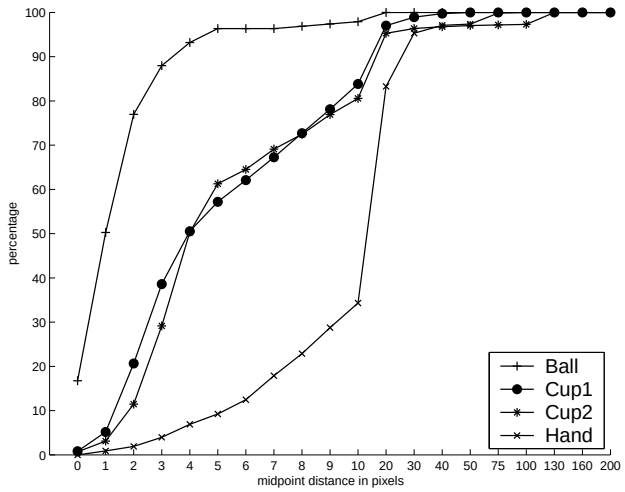
The trajectories in the Cups video (Figure 11(b)) are more distorted, as can be seen in item 1, 2 and 4. In this video, object outlines are blurry, resulting in less stable segment borders (see item 3). Item 4 marks the trajectory of the second cup which has not moved yet. The enclosed region results from the bright highlight on the cup. Figure 11(c) shows a zoomed-in view of the Ball object. There are two midpoints close together, one midpoint of the Ball object and one of the Background object.

The oversegmentation seen in the frame of the Corridor Video (Figure 11(d)) is caused by the textured background of this sequence (see items 1,2,6 and 7). The black line in item 5 is the Background object trajectory. Item 4 shows the trajectory of the Person object. Its distortion is caused by the Person consisting of multiple unconnected components.

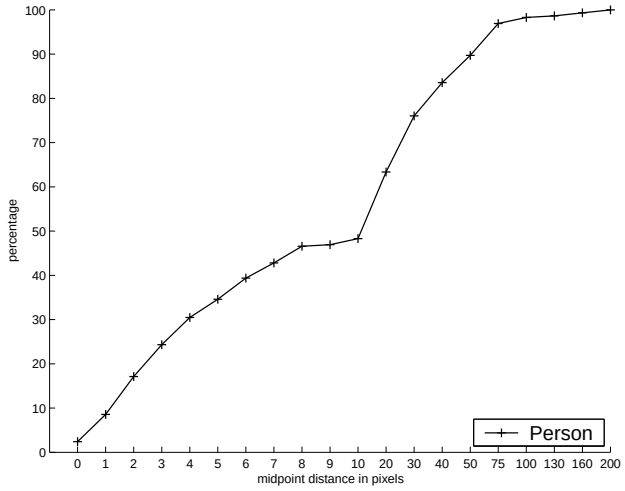
The plots of Figure 12 show how close the positions of the tracked regions correspond



(a) Synthetic Video



(b) Cups Video



(c) Corridor Video

Figure 12: Midpoint distance of the tracked regions to the reference midpoints in percentage of number of frames of the whole sequence.

to the reference positions. On the y-axis they show the percentage of frames in the video with midpoint distance smaller than a specific value on the x-axis in pixels. Only frames where the objects are recognized correctly were used in this plot. In the synthetic video and the cups video, only few frames (2%, respectively 5%) have a midpoint tracking error of more than 30 pixels. Pre-segmentation distorting the object outlines are the main cause for the midpoint tracking errors. In the corridor video, 97.6% of the frames have a midpoint tracking error of less than 100 pixels. The only object in this video, the person, consists of three unconnected components. The midpoint of only one component is tracked, which results in this tracking error.

Cups Video (total frames: 796*)*

	tracking percentage	*number of frames*
Hand	99.854%	795
Ball	88.426%	704
Cup1	93.401%	743
Cup2	94.275%	750

Table 3: Percentage of frames where objects are recognized correctly.

In the synthetic video the objects are recognized in every frame. Because of unhandled occlusions and slight undersegmentation in the cups video, in few frames objects are lost, see Table 3. The table shows the percentage of frames in which the object is recognized correctly. Cup1 and Cup2 are affected by these occlusions, so their recognition percentage is not optimal. The Ball's recognition percentage is lowest because it is lost in the last frames because of soft transitions and reflections of the ball on the table. In the corridor video, the person is recognized in 96.7% of the frames.

The MSTNcut algorithm shows good results with untextured objects although texture may cause oversegmentation since the algorithm only takes colour space distance in account, for this reason objects with high colour difference are preferable. The velocity of moving objects must be low enough so that the objects overlap in consecutive frames (see section 5.2.2). This method does not loose objects if they become static. Since elements from the pre-segmentation step will never be split the choice of the pre-segmentation algorithm strongly influences the final result. In our implementation the pre-segmentation algorithm had problems with soft blurry edges mostly caused by video compression and could not handle partial occlusions of objects with similar colour. Strong, pronounced shadows are difficult to handle because of their high contrast with the object. Objects consisting of multiple parts that are not touching are split up into the parts as well as objects with sharp colour edges between parts.

The videos were segmented on a AMD Athlon 64 bit processor 3000+ with 512 MB ram, gcc 3.3.6 (with full compiler optimization) on Ubuntu amd64 Linux. The synthetic video took 26.16 seconds (approximately 0.13 seconds per frame). The cups video took 206.8 seconds (approximately 0.26 seconds per frame) and the corridor video 626.28 seconds (approximately 1.75 seconds per frame). These times do not include pre-segmentation.

5.2 Reduction of the Spatial and Temporal Resolution in a Pre-processing Step

In each experiment we used pre-segmentation to reduce the number of vertices treated in the Ncut step. As mentioned in the introduction, our main motivation was to experiment with methods to reduce the spatial and temporal resolution of the videos before processing them. In the next subsections we will compare pre-segmentation to simple spatial sub-sampling and explain its advantages and disadvantages, as well as experiment with different methods to reduce the temporal resolution of the videos and evaluate its impact on the overall performance of the MSTNcut algorithm. We only present the results of the interpolation methods that yielded the best results. These are

- bilinear,
- bicubic and
- nearest neighbour

interpolation to reduce the spatial resolution.

To reduce the temporal resolution

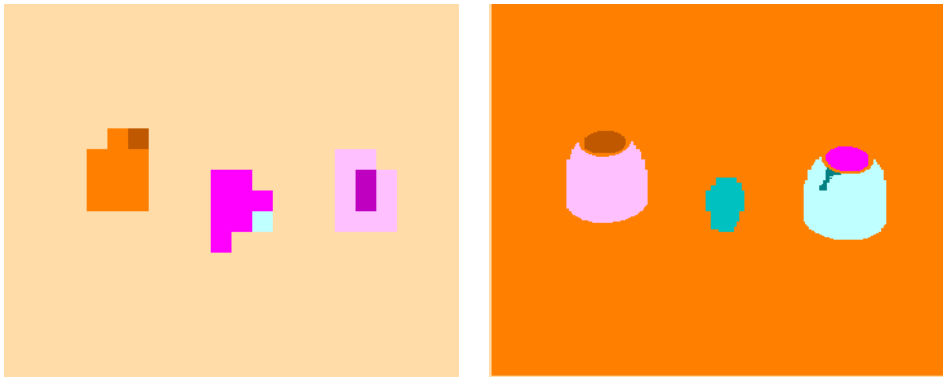
- subsampling,
- mean and
- median

interpolation were used.

5.2.1 Spatial Resolution

Since the MSTNcut algorithm uses a matrix storing similarity for each possible pair of elements, the number of elements in the Ncut step is the main factor for the performance of the algorithm. Taking every pixel as element is prohibitively time consuming, so reducing the number of elements in a pre-processing step is necessary. Simple sub-sampling is the fastest way of doing this. In this section we compare sub-sampling to pre-segmentation as a method of reducing the number of elements.

Figure 14 shows the results of sub-sampling a small range of representative frames of the cups video (frames 1-9 and 218-227) before applying the Ncut segmentation. The original image size was 352×288 pixels and bicubic sub-sampling was used with rates of 8×8, 16×16, 32×32 and 64×64, resulting in image sizes of 44×36, 22×18, 11×9 and 5×4 pixels. On the y-axis the plots show the percentage of frames in the video with midpoint distance smaller than a specific value on the x-axis, measured in pixels of the original image. The same video was used for some of the experiments using MST pre-segmentation, that are described earlier in this section. Remember that pre-segmentation resulted in 65-90 elements per frame. To compare results, refer to Figure 12(b).



(a) sub-sampling (b) MST pre-segmentation

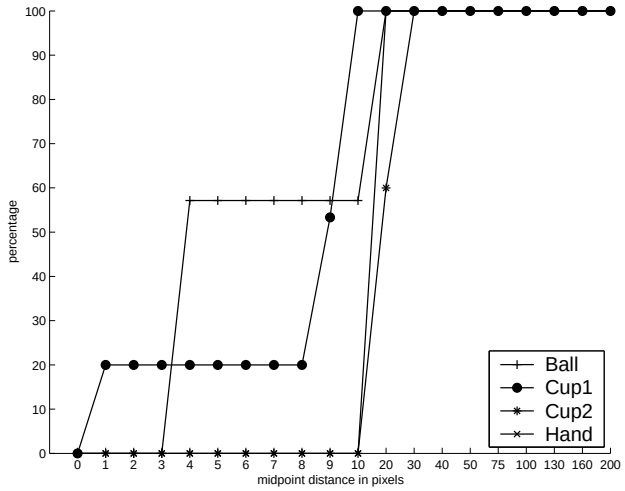
Figure 13: The result of segmenting a frame of the cups video which was sub-sampled to a size of 22x18 pixels and the result of segmenting the same frame using MST pre-segmentation.

Sub-sampling down to 44×36 pixels results in 1584 elements per frame (4752 elements per Ncut step), which is more elements than necessary when comparing to pre-segmentation. At this image size, the midpoint position error (distance of the tracked midpoint to the reference midpoint) is less than 30 pixels in every frame. Note from Figure 14(a) that the error is about 10-20 pixels in most frames. This error results from distorted object outlines (see Figure 13) and increases as the sub-sampling rate increases. At a size of 11×9 pixels (99 elements per frame, 297 per Ncut step), objects are lost in some frames, since too little of the object colour information remains in the resulting video. At a size of 5×4 pixels (20 elements, 60 per Ncut step) only the Ball object is tracked in every frame and one Cup object in about 50% of the frames. The other objects are not recognized.

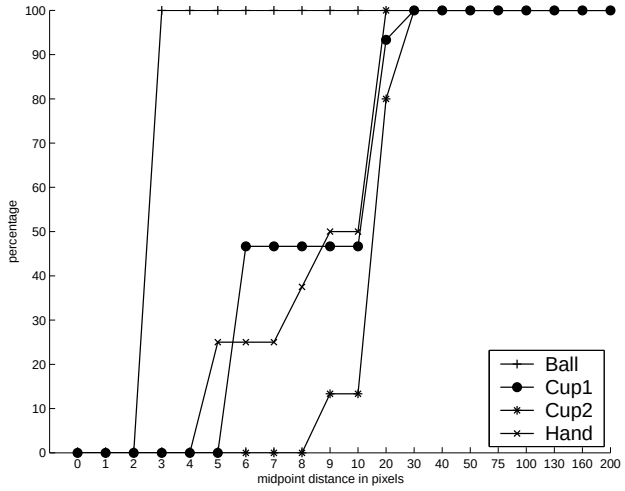
The experiment has shown three drawbacks of using sub-sampling as a method to reduce the number of elements. First, segment outlines are distorted because the number of pixels constituting the outline of a segment decreases as the sub-sampling rate increases (see Figure 13). Second, at image sizes that yield results comparable to pre-segmentation (e.g. 44×36), the number of resulting elements is larger than the number of elements obtained when using pre-segmentation. This is only true if there are large connected regions of similar colour, which can be found in most real-world images. Third, small or thin objects tend to be lost if they fall in between the sampling points (see Sampling Theorem [16]).

To clarify how the number of resulting elements and lost regions resulting from sub-sampling depend on region size and how they compare to pre-segmentation, tests were made on a synthetic image consisting of fifteen squares, the smallest one having a size of 1×1 pixel, the largest one a size of 99×99 pixels (see Figure 15).

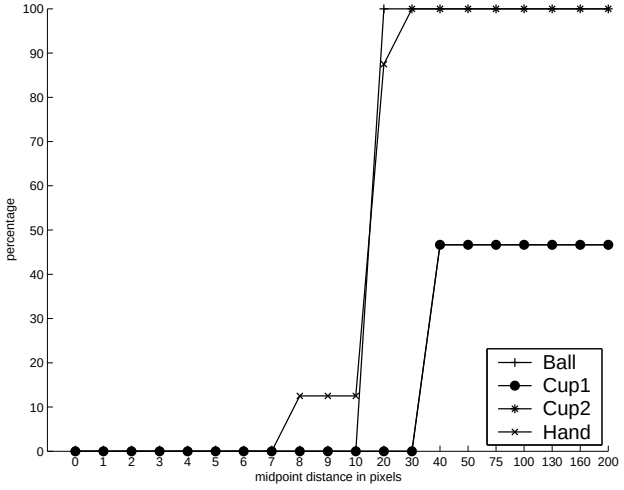
Table 4 shows the number of resulting elements, the number of lost squares (the number of squares that are not recognized by the MSTNcut algorithm anymore, out of a



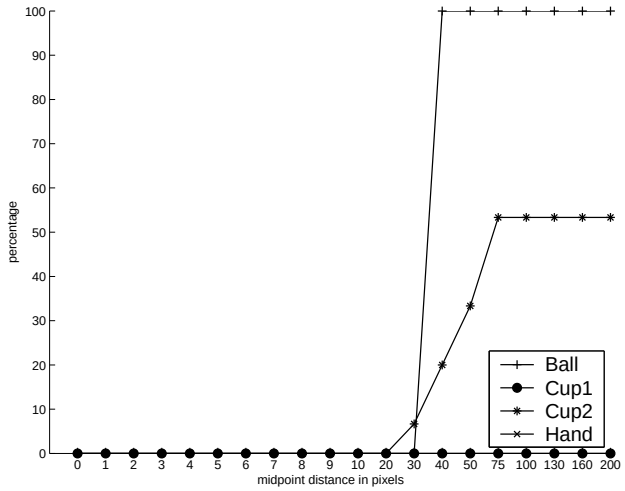
(a) subsampling to 44×36 pixels



(b) subsampling to 22×18 pixels



(c) subsampling to 11×9 pixels



(d) subsampling to 5×4 pixels

Figure 14: Midpoint distance of the tracked regions to the reference midpoints in percentage of number of frames of the whole sequence.

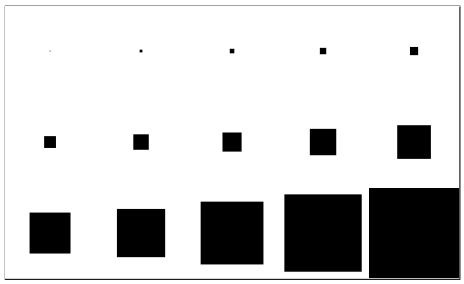


Figure 15: The image used for determining minimum object size at a given subsampling rate. The image size is 500×300 pixels.

total of 15) and the side length of the smallest recognized square (in number of pixels in the original image) for each image size and sub-sampling method. Since time and memory consumption grow quadratically with the number of elements, resolutions above 80×48 cannot be used as the MSTNcut algorithm would have prohibitive memory consumption. Already at the lowest sub-sampling rate, some of the rectangles are not recognized by the MSTNcut algorithm anymore. At 80×48 two to five rectangles are lost (depending on the sub-sampling method), which translates to maximum object diameter of 5 to 9 pixels. These numbers increase as the sub-sampling rate increases. At a resolution of 20×12 14 objects are lost and the minimum object diameter is 29 pixels.

Table 5 shows the results of pre-segmenting of the same synthetic image with the MST method. Although more time-intensive, this method results in a smaller number of elements while only discarding redundant information. In this synthetic image no objects are lost during pre-segmentation.

Sub-sampling is the fastest way of reducing the number of elements, but the resulting region outlines are less accurate and thin or small objects may be lost. Pre-segmentation is more time-intensive, but results in less elements if applied on a video containing large

image size	sub-sampling method	# elements	# lost squares	min. side length
80x48	bicubic	3840	4	7 pixels
80x48	bilinear	3840	5	9 pixels
80x48	nearest neighbour	3840	3	5 pixels
60x36	bicubic	2160	5	9 pixels
60x36	bilinear	2160	5	9 pixels
60x36	nearest neighbour	2160	4	7 pixels
40x24	bicubic	960	7	17 pixels
40x24	bilinear	960	5	9 pixels
40x24	nearest neighbour	960	4	7 pixels
30x18	bicubic	540	7	17 pixels
30x18	bilinear	540	8	21 pixels
30x18	nearest neighbour	540	7	17 pixels
20x12	bicubic	240	9	29 pixels
20x12	bilinear	240	9	29 pixels
20x12	nearest neighbour	240	9	29 pixels

Table 4: The number of resulting elements, the number of lost squares and the minimum square side length vs. sub-sampling method and image size in pixel.

image size	pre-segm. method	# elements	# lost squares	min. side length
500x300	MST	16	0	1 pixel

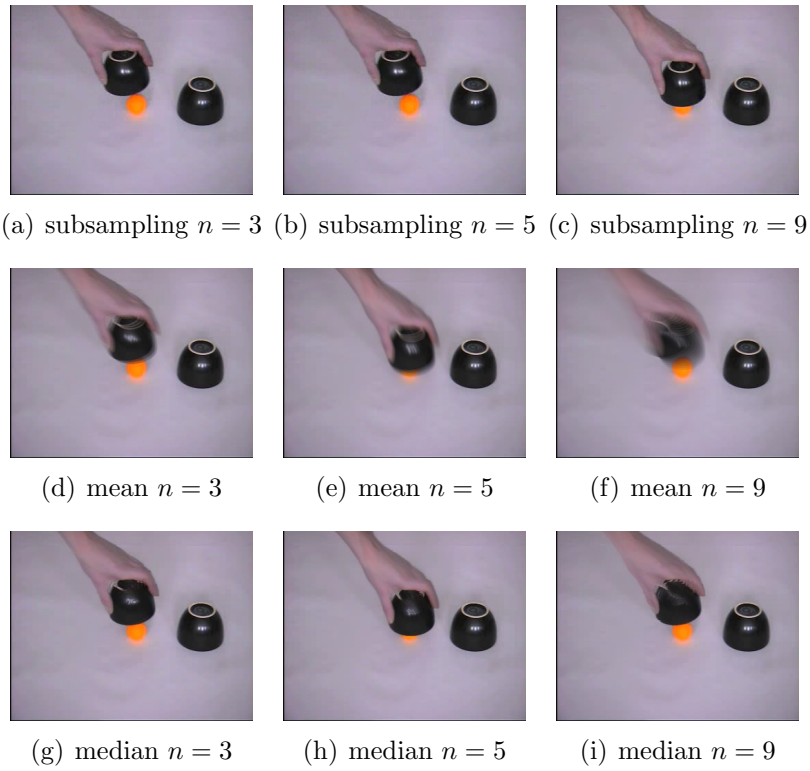
Table 5: This table shows the results of pre-segmenting the synthetic image with the MST segmentation method.

connected regions of similar colour and does not loose small objects. Since the main factor for the speed of the Ncut algorithm is the number of elements, the overall performance of the MSTNcut algorithm benefits from using pre-segmentation. When sub-sampling the cups video spatially to 44×36 pixels (1584 elements per frame), the MSTNcut algorithm needs approximately 21 seconds per frame, when sub-sampling to 22×18 pixels approximately 1.25 seconds and when using MST pre-segmentation approximately 0.3-0.4 seconds per frame (including the 0.1-0.2 seconds needed for pre-segmentation).

5.2.2 Temporal Resolution

Another way to reduce the amount of video data is the reduction of the temporal resolution. There are several possibilities to do so. Sub-sampling the video by only taking every n-th frame into account is the most performant variant. A different approach is to calculate the mean values of each colour channel of n frames. This approach is more time consuming but information from across the n frames is incorporated into the remaining frame. Instead of taking the mean it is also possible to take the median to eliminate outliers.

To evaluate the performance of the MSTNcut algorithm, experiments were done with all frames of the cups video. For all experiments tests were made with $n = 3, 5$ and 9 because for calculating the exact median value uneven numbers have to be used. The amount of data is reduced to 1/3, 1/5 respectively 1/9 of the original data. The images in Figure 16 show frames generated with subsampling (which is identically to the original frame) and frames gained by calculating the mean and the median of every n-th frame.



(a) subsampling $n = 3$ (b) subsampling $n = 5$ (c) subsampling $n = 9$

(d) mean $n = 3$ (e) mean $n = 5$ (f) mean $n = 9$

(g) median $n = 3$ (h) median $n = 5$ (i) median $n = 9$

Figure 16: Sample images with different interpolation methods

Like in Section 5 the performance was measured by calculating the distance of the mid-point positions and the percentage of the frames where the objects are tracked correctly. The plots in Figures 17-19 illustrate the results and Table 6 shows the rate of frames where objects are tracked correctly. The MSTNcut algorithm showed nearly the same mid-point distances when using subsampling with $n = 3$ and 5 and with no temporal reduction except that Ball's recognition rate dropped because the algorithm was not able to track it at the end of the video, since the effect of soft transitions and reflections increased. With $n = 9$ the objects are lost often because they were moving too fast and motion blur occurs.

When using the mean value to reduce the temporal resolution the MSTNcut algorithm showed no good performance. Most of the edges of the regions become very blurry and the pre-segmentation step was not able to provide the MSTNcut algorithm with a good segmentation of the images. This effect amplifies with increasing n. Even with $n = 5$ the

hand is not tracked and with $n = 9$ the MSTNcut algorithm is only able to track one cup. Using the median for temporal reduction the MSTNcut algorithm performed well with $n = 3$ and $n = 5$ and showed pretty good results even with $n = 9$ for objects with sharp edges and uniform colour (Cups) but had problems with the Hand, with shadowed regions and soft edges to the background.

Temporal subsampling and the median value showed the best results, since the images still contain sharp edges while the mean value makes them blurry. Using the median and the mean value eliminates outliers, but only the median provides the Ncut step with a high differences in colour space.

subsampling	$n = 3$	$n = 5$	$n = 9$
Hand	100%	100%	44.595%
Ball	57.746%	57.143%	0%
Cup1	95.038%	90.446%	64.368%
Cup2	90.076%	91.667%	58.140%

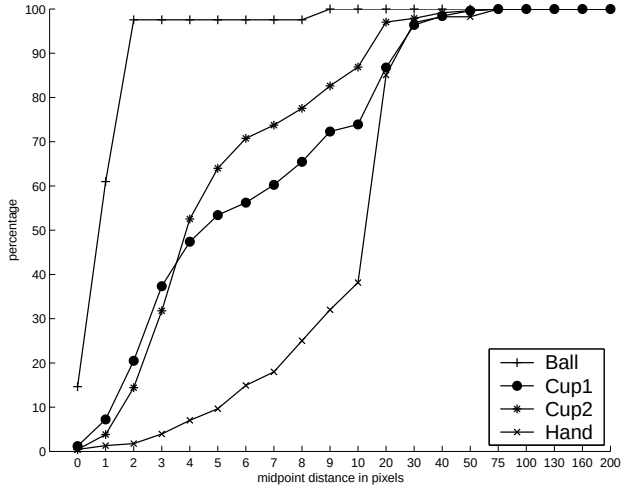
mean	$n = 3$	$n = 5$	$n = 9$
Hand	76.646%	0%	0%
Ball	95.775%	57.143%	0%
Cup1	72.137%	64.968%	26.744%
Cup2	32.824%	85.256%	0%

median	$n = 3$	$n = 5$	$n = 9$
Hand	99.558%	100%	60.811%
Ball	57.746%	57.143%	0%
Cup1	95.038%	95.541%	98.851%
Cup2	90.076%	91.667%	75.581%

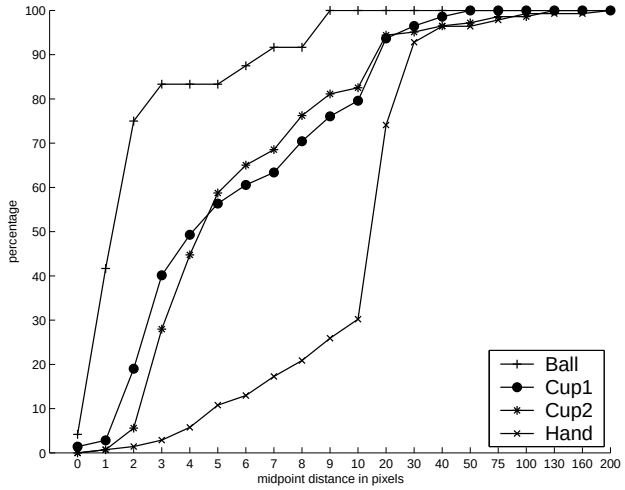
Table 6: Percentage of frames where objects are recognised correctly using the cups video and different methods to decrease temporal resolution.

The decrease of the temporal resolution makes objects move faster and the MSTNcut algorithm will lose them after a certain velocity. To evaluate this effect and the connection between the velocity and the object size we made an experiment. Seven new synthetic videos are introduced, each with a circle with a different size and linear acceleration moving in a circular path. Sample images of the frames are shown in Figure 20. The circles in the video are solid-coloured. We observed when the MSTNcut algorithm loses the circle and so we can determine the maximum speed the algorithm can handle with a given size.

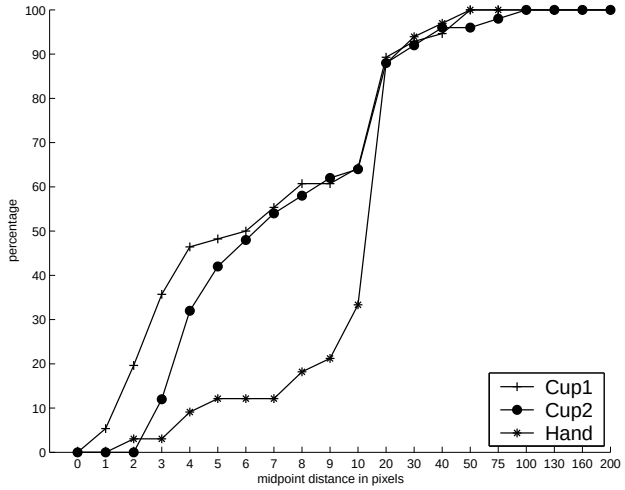
Tables 7 - 10 show the results of these experiments. They show the radius of the moving circle and the maximum velocity (in pixel distance) of the circle the MSTNcut algorithm can handle before the circle gets lost. Without any reduction of temporal resolution the maximal pixel distance between the circle in two frames is the double radius (Table 7) since the objects have to touch each other in the different frames. When using subsampling



(a) subsampling $n = 3$

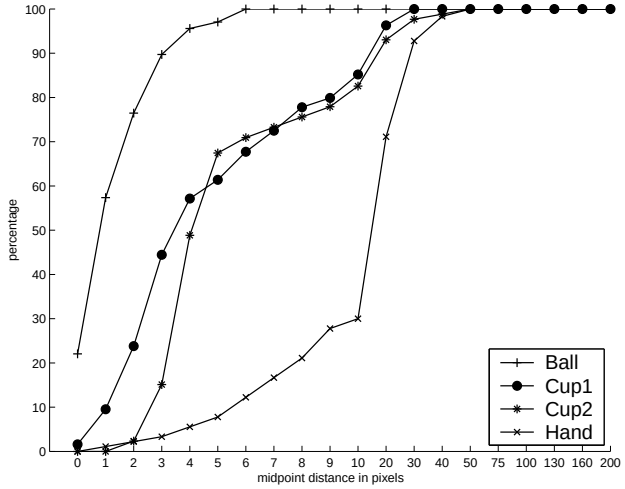


(b) subsampling $n = 5$

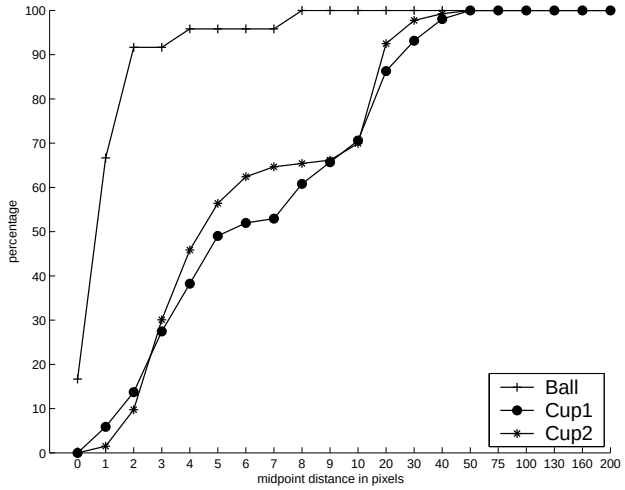


(c) subsampling $n = 9$

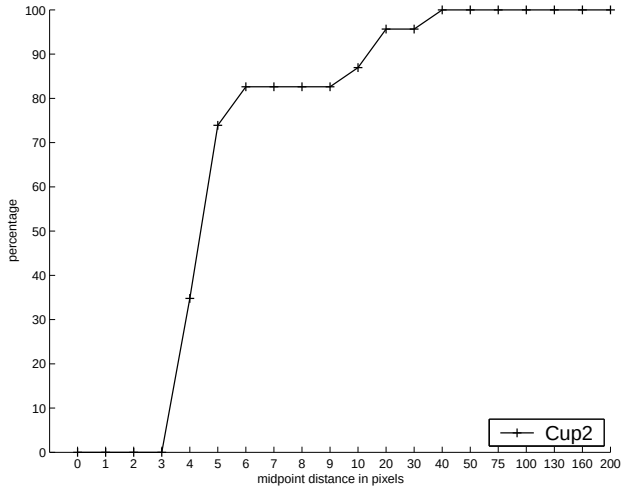
Figure 17: Midpoint distance of the tracked regions to the reference midpoints in percentage of number of frames of the whole sequence using every $n-th$ frame with subsampling.



(a) mean $n = 3$

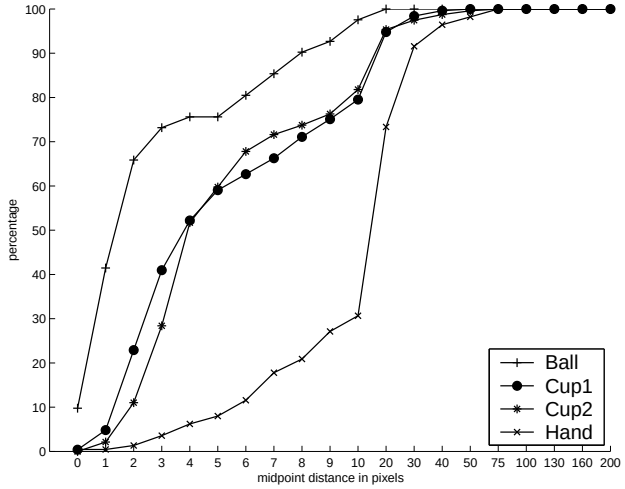


(b) mean $n = 5$

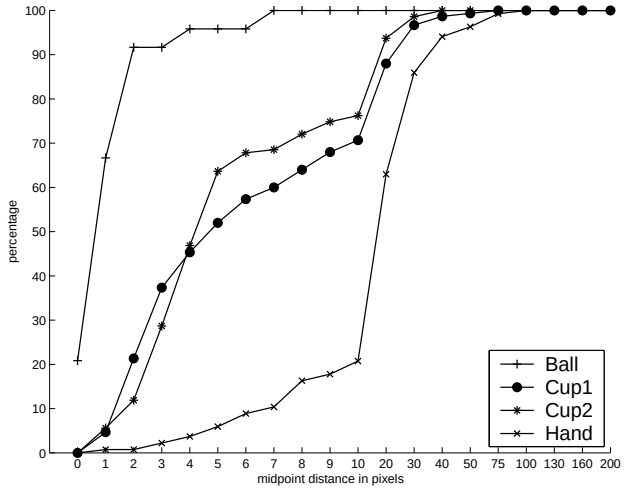


(c) mean $n = 6$

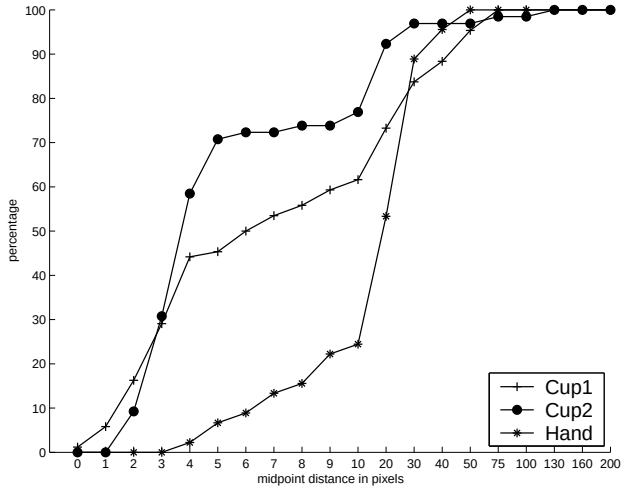
Figure 18: Midpoint distance of the tracked regions to the reference midpoints in percentage of number of frames of the whole sequence using every $n - th$ frame with the mean value.



(a) median $n = 3$



(b) median $n = 5$



(c) median $n = 9$

Figure 19: Midpoint distance of the tracked regions to the reference midpoints in percentage of number of frames of the whole sequence using every $n - th$ frame with the median.

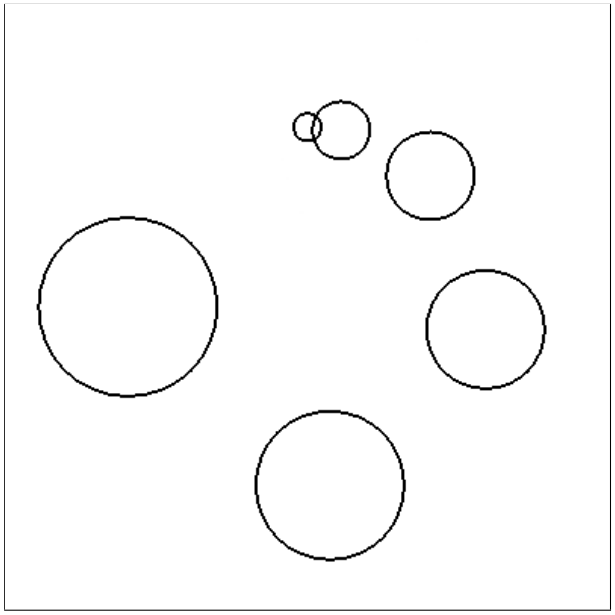


Figure 20: Videos of circles with increasing sizes and speed were used to evaluate the connection between the velocity and the object size.

circle radius	without reduction
radius = 10	20
radius = 20	40
radius = 30	60
radius = 40	80
radius = 50	100
radius = 60	120
radius = 75	150

Table 7: Maximum pixel difference of the mid-points of a moving object in two frames, with no reduction of temporal resolution,so that it is recognised correctly.

the results are quite good (Table 8). The maximum distance is approximately 2/3 of the radius, with $n = 3$ the object is allowed to move 2/3 of the radius per frame to get the distance of the double radius after subsampling. It has to be mentioned that a synthetic video has been used without any outliers which are a disadvantage of using subsampling. Using the mean value for the reduction, the worst results are produced (Table 9). The circle becomes very blurry and the MSTNcut algorithm is no longer able to follow him. If the median is used to reduce the temporal resolution the results are comparable with the ones with subsampling but with the advantage that outliers would be eliminated.

circle radius	subsampling $n = 3$	subsampling $n = 5$	subsampling $n = 9$
radius = 10	7	5	-
radius = 20	14	9	-
radius = 30	21	14	4
radius = 40	28	16	4
radius = 50	35	20	12
radius = 60	35	25	16
radius = 75	53	34	20

Table 8: Maximum pixel difference of the mid-points of a moving object in two frames, using subsampling to reduce temporal resolution,so that it is recognised correctly.

circle radius	mean $n = 3$	mean $n = 5$	mean $n = 9$
radius = 10	8	9	-
radius = 20	10	7	-
radius = 30	14	9	-
radius = 40	18	14	-
radius = 50	22	14	-
radius = 60	26	16	12
radius = 75	34	20	12

Table 9: Maximum pixel difference of the mid-points of a moving object in two frames, using the mean to reduce temporal resolution,so that it is recognised correctly.

circle radius	median $n = 3$	median $n = 5$	median $n = 9$
radius = 10	8	5	-
radius = 20	15	9	-
radius = 30	22	14	4
radius = 40	28	18	12
radius = 50	35	23	12
radius = 60	43	27	16
radius = 75	55	34	30

Table 10: Maximum pixel difference of the mid-points of a moving object in two frames, using the median to reduce temporal resolution,so that it is recognised correctly.

6 Conclusion

We have presented a method for segmenting videos using the Ncut method with an additional pre-segmentation step. First we identify the most salient region of each frame using the Felzenszwalb et.al. MST method [9]. An energy function, calculating the similarity between each pair of elements in a frame window, is used to construct a weighted, fully connected graph. Using the Ncut criterion repeatedly we build up a dendogram. The leaf nodes of the dendogram identify the initial segments of this frame window. To reduce the oversegmentation initial segments are merged into the final segments. The MSTNcut algorithm performs equally well on moving and on static objects. The MSTNcut algorithm has problems with blurry object borders, shadows and objects consisting of multiple components. We have showed that pre-segmentation is in most cases more reliable than subsampling, resulting in more accurate elements. This becomes particularly apparent when using sequences with small objects. Furthermore we tested methods to reduce the temporal resolution increasing the overall performance. Using the median of a group of frames yielded at the best performance since it eliminates outliers and does not blur edges.

6.1 Open Problems

Currently our implementation does not handle occlusions. As described above two occluding objects with similar colour are treated as a single object in the pre-segmentation step. To overcome this problem it would first be neccessary to chose a pre-processing algorithm that can handle occlusions since our implementation never splits up regions formed by the pre-processing algorithm. Another way improving the MSTNcut algorithm would be to include motion information for example by measuring pixel based motion flow and including this information in similarity calculation or by using final segment motion information obtained by observing motion of the final segments in previous frames and predicting the position in the current frame.

7 Acknowledgements

We would like to thank W. Kropatsch and Y. Haxhimusa for their support and many useful tips and A. Ion for providing us with image segmentation libraries and useful suggestions. We also thank A. Hanbury for making video material available to us through the Muscle Benchmarking Project [17] and the Advanced Computer Vision Gmbh. for their video material.

We gratefully acknowledge the support by the Austrian Science Fund grants P18716-N13 and S9103-N04.

A Definitions of the used terms

MST Method: Minimum Spanning Tree. A method to segment an image. See [9] for details
elements: The regions resulting the pre-segmentation.

initial segments: The regions resulting from applying the Ncut Dendogram method to elements.
final segments: The regions resulting from merging the initial segments. This is the output of the MSTNcut algorithm.
frame window: Three consecutive frames of a video sequence. The frame window is moved over the whole sequence. In each step the MSTNcut algorithm considers only information of one frame window.
similarity: The measure of the similarity of two elements. In our case the distance in the IHLS colour space.
similarity function: The energy function calculating the similarity of two elements.
similarity matrix: The adjaceny matrix of the fully connected graph of all elements in a frame window with the similarity as weights on each edge.

B Pseudo Code

Below, the dendogram construction is shown in pseudo code:

```
method: initial_segments = ncut(similarity_matrix)
  split the similarity matrix using ncut
  if not maximum depth is reached then
    return ncut(left_half)
    return ncut(right_half)
  else
    elements_segment_left = get_elements(left_half);
    elements_segment_right = get_elements(right_half);
    return initial_segments.add(elements_segment_left, elements_segment_right);
  end /* if */
```

`get_elements` returns the elements of a given similarity matrix.

The pseudo code below shows how the MSTNcut algorithm merges initial segments until a given minimum similarty or a minimum number of segments is reached:

```
method: final_segments = merge(initial_segments)

  repeat
    (seg_i,seg_j) = find most similar neighbours in initial segments;
    if similarity(seg_i, seg_j) > threshold
      initial_segments.add( merge(seg_i,seg_j) );
    else if number_of_segments =< minimum_segment_count
      return inital_segments;
    else
      return intial_segments;
```

References

[1] D.G. Lowe - Robust model-based motion tracking through the integration of search and estimation - International Journal of Computer Vision, Vol. 8, no 2, 1992, p. 113-122

[2] F. Dufaux, F. Moscheni, A. Lippman - Spatio-temporal segmentation based on motion and static segmentation - Proc. IEEE Int. Conf. Image Processing, pp. I.306–I.309, Oct. 1995.

[3] M.J. Black - Combining intensity and motion for incremental segmentation and tracking over long image sequences - Proc. European Conf. Computer Vision, pp. 485–493, May 1992. Stiller & Konrad

[4] Y. Huang, T.S. Huang, H. Niemann - A region-based method for model-free object tracking - In Proceedings of 16th International Conference on Pattern Recognition (ICPR'02) - Volume 1, 11-15 Aug. 2002 Pages:592 - 595 vol.1

[5] V. Rehrmann - Regionenbasierte Bewegungssegmentierung, In Proceedings "3. Workshop Farbbildverarbeitung" - Erlangen, 25.-26. September 1997

[6] M. Gelgon, P. Bouthemy - A region-level motion-based graph representation and labeling for tracking a spatial image partition - Pattern Recognition, 33(4):725-740, April 2000

[7] J. Shi and J. Malik - Normalized Cuts and Image Segmentation - In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97), pages 731–737, 1997

[8] J. Shi and J. Malik - Motion Segmentation and Tracking Using Normalized Cuts - In IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 8, pages 888–905, August 2000

[9] P.F. Felzenszwalb, D.P. Huttenlocher - Efficient Graph-Based Image Segmentation - In International Journal of Computer Vision 59(2), pages 167-181, 2004

[10] G. Rote - Computing the minimum Hausdorff distance between two point sets on a line under translation - Information Processing Letters, 38: pages 123– 127, 1991

[11] A. Hanbury and J. Serra - A 3D-polar coordinate colour representation suitable for image analysis - Technical Report PRIP-TR-77, PRIP, T.U. Wien, 2002

[12] R. Lehoucq, K. Maschhoff, D. Sorensen, C. Yang - ARPACK - Arnoldi Package, http://www.caam.rice.edu/software/ARPACK/

[13] W. E. Arnoldi - "The principle of minimized iteration in the solution of the matrix eigenvalue problem" - Quarterly of Applied Mathematics, volume 9, pages 17?25, 1951.

[14] Synthetic Video http://addresse.wird.bekanntgegeben.sobald.wir.wissen.wos.liegt

[15] Video Material of the Austrian Joint Research Program on Cognitive Vision http://www.prip.tuwien.ac.at/Research/FSPCogVis/index.html

[16] C. E. Shannon - Communication in the presence of noise - In Proc. Institute of Radio Engineers, vol. 37, no.1, pp. 10-21, January 1949

[17] Muscle Benchmarking, http://muscle.prip.tuwien.ac.at

[18] Viper-GT, http://viper-toolkit.sourceforge.net/products/gt/