Video Tooning

Jue Wang[1,2] Yingqing Xu[2] Heung-Yeung Shum[2] Michael F. Cohen[3]

University of Washington[1] and Microsoft Research (Asia[2] and Redmond[3])

Figure 1: Two examples of Video Tooning.

Abstract

We describe a system for transforming an input video into a highly abstracted, spatio-temporally coherent cartoon animation with a range of styles. To achieve this, we treat video as a space-time volume of image data. We have developed an anisotropic kernel mean shift technique to segment the video data into contiguous volumes. These provide a simple cartoon style in themselves, but more importantly provide the capability to semi-automatically rotoscope semantically meaningful regions.

In our system, the user simply outlines objects on keyframes. A mean shift guided interpolation algorithm is then employed to create three dimensional semantic regions by interpolation between the keyframes, while maintaining smooth trajectories along the time dimension. These regions provide the basis for creating smooth two dimensional edge sheets and stroke sheets embedded within the spatio-temporal video volume. The regions, edge sheets, and stroke sheets are rendered by slicing them at particular times. A variety of styles of rendering are shown. The temporal coherence provided by the smoothed semantic regions and sheets results in a temporally consistent non-photorealistic appearance.

1 Introduction

Animated imagery brings life to the screen. The stylized abstraction of reality one sees in animation adds an immediate impact that cannot be captured by simply pointing a video camera at a scene. But such animation is both labor intensive and requires considerable artistic skill.

We present methods to lower these barriers by allowing video to be transformed into a cartoon-like style. Stylized rendering of video, which we dub *Video Tooning* is an active area of research in non-photorealistic rendering (NPR). Our work was motivated in part by the still image stylization and abstraction approach presented by DeCarlo and Santella [2002]. Unfortunately, a direct frame-by-frame application of this approach to stylize video results in a temporally very incoherent result. Our goal then has been to produce temporally coherent stylization while also allowing a user significant freedom in choosing the final look of the video.

Generally, there are three major criteria that a successful video tooning system should meet:

1. The result sequence should maintain spatio-temporal consistency to avoid significant jumps in frame transitions.
2. The content of the video should be abstracted in such a way as to respect the higher level semantic representation.
3. The artist should be able to express control over the style of the result.

When NPR methods designed for static images are applied to video sequences on a frame-by-frame basis, the results generally contain undesirable temporal aliasing artifacts. We overcome the coherence problem by accumulating the video frames to create a 3D data volume and directly cluster the pixels in the three dimensional space (x,y,t). This avoids many of the robustness problems of optical flow methods that track pixel or object movements only

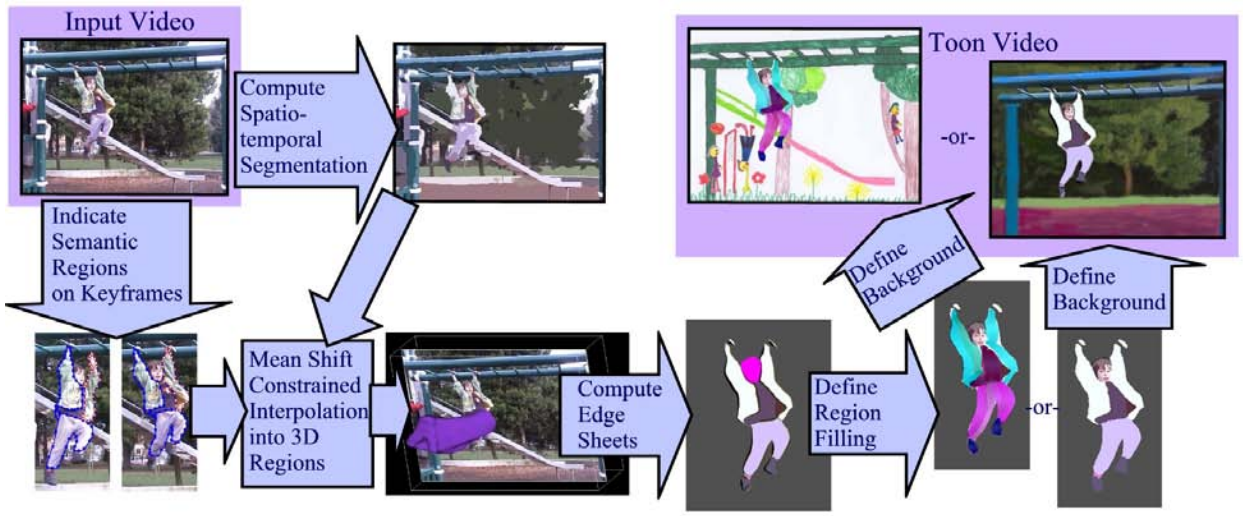

Figure 2: The flow chart of our Video Tooning system.

between successive frames. We have developed a spatio-temporal video segmentation technique called *anisotropic kernel mean shift* [Wang et al. 2004]. It is based on the basic mean shift image segmentation method [Comaniciu and Meer 2002] but is extended to achieve better results on video segmentation. The segmentation results in clusters of pixels that have similar visual attributes and are contiguous in space and time. These sub-volumes provide the low-level spatio-temporal coherence information to create higher-level semantic abstraction.

Cartoon animations are typically composed of large regions which are semantically meaningful and highly abstracted by artists. A region may simply be constantly colored as in most cel animation systems, or it may be rendered in some other consistent style. To achieve similar results, we build a rotoscoping interface that enables the user to outline the semantic regions within the spatio-temporal video volume. A contribution of our interface is, instead of requiring the user to draw in a frame-by-frame way like the traditional cartoon art, we leverage the low-level spatio-temporal coherence information provided by the segmentation to facilitate the user's work and greatly reduce the required labor. The user draws on several keyframes which are sparsely distributed along the whole sequence. A mean shift guided shape interpolation method then propagates the user's editing from keyframes to all other frames to create 3D *semantic regions*.

Semantic 3D regions are further processed into polyhedral surface representations and smoothed. From these, we create a set of *edge sheets* from portions of surfaces. Following a similar interface, the user is able to add strokes within regions on keyframes. These are also propagated through time guided by the semantic regions. These result in smooth "*stroke sheets*" within the solid regions.

At this point, the video is represented as a set of 3D polyhedral semantic regions, 2D edge sheets along the surface of the regions and stroke sheets within the regions. These primitives, when sliced at a frame time yield solid areas and curves along their edges and within the interior. We present a variety of rendering options for these primitives to construct a frame of the stylized video.

Figure 2 shows a flow chart of our system. The system to transform a video sequence is summarized as follows:

- A set of volumetric objects is determined by mean shift video segmentation;
- The user draws on a limited number of keyframes to indicate how small segments should be merged into larger, semantic regions;
- The user's indications are interpolated between keyframes by a mean shift guided interpolation technique propagating the user's input to all frames;
- The user can optionally draw *paint strokes* within regions at keyframes. These are similarly interpolated.
- Semantic regions and surfaces are reconstructed and smoothed. Edge and stroke sheets are determined.
- At each frame time regions and sheets are sliced to yield area and curve primitives.
- These primitives are rendered in desired style to create final stylized video frame and output.

2 Related Work

We are not the first to present methods to stylize video. Some approaches apply NPR rendering methods frame-by-frame or used simple interpolation techniques. Recent films such as *A Waking Life* and *Avenue Amy* were painstakingly modified one or a few frames at a time. Although the stylized look of this work has some appeal, the jitter produced by these methods may or may not have been the goal of the artists. In any case, the tedious workload made these productions very expensive to complete.

In 1997, Litwinowicz [1997] proposed an automatic approach to produce painterly animations from video clips. Optical flow fields were used to push brush strokes from frame to frame in the direction of pixel movements. Hertzmann et. al. [2000] modified each successive frame of the video by first warping the previous frame to account for optical flow changes and then painting over areas of the new frame that differ significantly from its predecessor. This was extended in [Hertzmann 2001] by guiding paint strokes with a general energy term consisting of both pixel color differences and optical flow. Our mean shift based system can be thought of as a more global optimization, similar in spirit to the optical flow employed locally in these systems.

Manual rotoscoping tools for extracting figures from video are commercially available, for example, Commotion and Adobe After Effects. These systems most likely use optical flow to track individual points forward in time. The animators for *Waking Life* were given some automation tools such as an ability to interpolate individual strokes between two separate frames and/or move large billboard-like layers [Linklater 2001]. In contrast we perform a more global optimization to interpolate loops and curves drawn on keyframes to provide temporal coherence and fidelity to the user's specification.

There is a vast literature on finding and tracking objects and features in the vision literature. One classic approach is the Snakes system [Kass et al. 1987] to find curves in an image and then adapted by Hoch and Litwinowicz [1996] for tracking. There is an equally vast literature on image segmentation in which one finds the mean shift method [Comaniciu and Meer 2002] we have extended.

One can also view part of our work as extending the in-betweening problem in keyframe animation. A good recent addition to this problem for NPR is the work by Kort [2002]. The SnakeToonz system [Agarwala 2002] used spline-based active contours for tracking user sketched contours of simple objects displayed in front of solid backgrounds. The goals of this work are similar to ours. However, our methods also work in more unconstrained environments and provide more stylistic choices to the user through the use of edge and stroke sheets. The active contours of SnakeToonz are also implicitly supported by the mean shift result.

Some recent video processing techniques treat video as a space-time volume of image data. In the Stylized Video Cubes approach [Klein et al. 2002], a set of "rendering solids" were created in the volume as a function defined over an interval of time; when evaluated at a particular time within that interval, each rendering solid provided parameters necessary for rendering an NPR primitive. Although this system resulted in interesting abstractions of the underlying video it had no facilities to respect larger semantic regions as they evolve through time. In [Collomosse et al. 2003] the authors describe aspects of their *Video Paintbox* which include the creation of spatio-temporal subvolumes by associating segmentation results between contiguous frames. They also describe the creation of *stroke surfaces* in much the same spirit as the edge sheets we describe later. Our system extends 2D computer vision segmentation techniques to directly provide 3D primitives in the video volume that maintain spatial and temporal coherence. These provide the underlying structure for the user to then define semantic regions.

Our choice to modify mean shift for video was inspired by its success on still images as shown by DeCarlo and Santella for NPR in their paper *Stylization and Abstraction of Photographs* [2002]. In their system, images were transformed into a style combining of line-drawing and filling large regions with constant color. For abstraction, the system used eye-tracking data to determine where to remove extraneous details and to highlighting important objects. We achieve a similar goal on video sequences by extending mean shift segmentation to leverage temporal coherence. We also render video frames using line-drawing and large regions with constant color, which are temporally consistent. In our case, the regions are determined by an artist rather than eye tracking. We also have extended the stylistic choices by allowing the artist to embed stroke sheets within the larger semantic regions.

3 Spatio-Temporal Coherence Analysis

Typical video contains strongly correlated frames as well as sudden changes across shot boundaries. This paper focuses only on the finer level, in other words, only the frames within each shot. We consider the pixels within each shot as a 3D (spatio-temporal) lattice of color values. We first process the video pixel data to determine discrete 3D shapes, or spatio-temporal video *segments*. For this purpose, we have extended a mean shift algorithm [Comaniciu and Meer 2002] to better address the type of features found in video data [Wang et al. 2004].

Figure 3: A frame of the mean shift result. Each pixel is colored as the average of the pixels within each segment.

3.1 Mean Shift Segmentation

An image segmentation is simply a partition of an image into contiguous regions of pixels that are similar in appearance. Mean shift is a general nonparametric technique for the analysis of a complex multimodal feature space and the delineation of arbitrarily shaped clusters [Comaniciu and Meer 2002]. Although mean shift estimation was developed decades before [Fukunaga and Hostetler 1975], its properties of data compaction and dimensionality reduction have been explored only recently for low level computer vision tasks such as non-rigid object tracking [Comaniciu et al. 2000] and image segmentation [Christoudias et al. 2002]. Its convergence on lattices has also been proven. Recent work adopted this method for creating non-photorealistic stylization of images [DeCarlo and Santella 2002]. We extend this work to 3D video segmentation.

Pixels in an image can be thought of as lying in a 5 dimensional space defined by the 2 image axes and 3 color components. Pixels that lie close to each other in this multivariate feature space form *denser* regions. The density measure at any point in the space is defined by a weighted sum of nearby pixels. The contribution to the density of the point of each nearby pixel is defined by a *kernel*, which has its maximum at the point and drops off with distance. Mean shift is an algorithm to find each pixel's mode (local maxima) of the density estimate. It proceeds by iteratively moving a *mean shift point* associated with each pixel upwards along the gradient of the density function until a mode is reached. All pixels associated with points that move to the same mode are then considered part of the same segment.

One of the most actively studied aspects of mean shift is how best to define the kernel which implicitly defines a measure of *distance* between pixels. In an associated paper [Wang et al. 2004] we discuss the details of the mean shift method and our contribution of the use of an anisotropic kernel to better address the types of structures seen in video sequences.

3.2 Mean Shift Segmentation for Video

For video, one could simply perform mean shift segmentation on each frame individually, but as noted earlier this results in a temporally very noisy result (see video). Instead, we perform a video

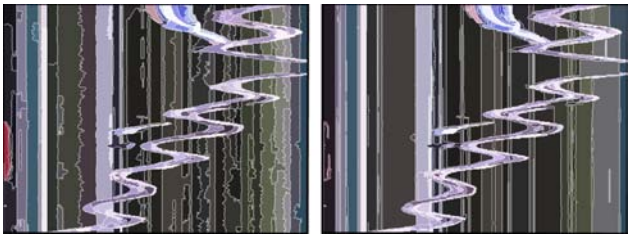

Figure 4: Comparison of radially symmetric kernel mean shift (left) and anisotropic kernel mean shift(right) on a spatio-temporal slice of the monkey bars sequence. The straighter edges in vertical (temporal) dimension lead to improved temporal coherence.

segmentation, a partition of the video as a whole into contiguous volumes of pixels defined on the 3D lattice (x,y,t).

We could adopt the same radially symmetric kernels typically used for images but extended by the one extra dimension of time. This does, in fact, improve the temporal coherence considerably. However, if one examines spatio-temporal slices of video data (see Figure 4), particular features are very evident that do not arise normally in images. Many long thin features are common indicating the motion, or lack thereof, of objects across the visual field. Radially symmetric kernels thus do not represent well an intuitive notion of nearby in this domain.

Figure 4(left) illustrates the problem with relying on radially symmetric kernels for video segmentation by examining a spatio-temporal slice (parallel to the temporal axis) of the video. Such a slice is as representative of the underlying data as any single original frame (i.e., a slice orthogonal to the temporal axis). Radially symmetric kernels tend to attract pixels across the thin strips which clearly should be kept separate. This causes a noisy result in the segmentation when the segmented video is subsequently sliced in time for rendering.

In [DeMenthon 2002], this problem was recognized and addressed by adding optical flow information to the pixel itself. However, as the author pointed out, introducing motion components into the features will destroy the boundaries of moving regions and make them jagged. The higher dimension also make the points much more sparse and contributes to the noisy results.

As we show in [Wang et al. 2004], we overcome the problems of radially symmetric kernels by creating a different kernel for each pixel that adapts to the local density. These kernels take on a general ellipsoidal shape in the spatio-temporal domain determined by the local covariance of the positions of nearby pixels of similar color. Thus the anisotropic kernels adapt to long thin shapes both in the spatial and time dimensions leading to better temporal coherence as can be seen in the smoother segment boundaries in Figure 4(right).

Anisotropic kernels also provide a set of handles for application-driven segmentation. For instance, we may desire that the still background objects be more coarsely segmented while the details of the moving objects to be preserved when segmenting a video sequence. Details can be found in [Wang et al. 2004]. Pseudocode of the anisotropic mean shift algorithm can be found in the appendix at the end of this paper as well.

4 Non-photorealistic Rendering of the Video

The mean shift segmentation results in volumes of contiguous pixels with similar color. The simplest abstraction of the video can then be created by simply coloring all pixels the average color of the segment and then slicing this in time to create a sort of ”paint-by-number” non-photorealistic rendering (see Figure 3). Segment colors can be changed (see Figure 5) and/or segment edges drawn for artistic effect.

Figure 5: Mean shift segments selectively saturated.

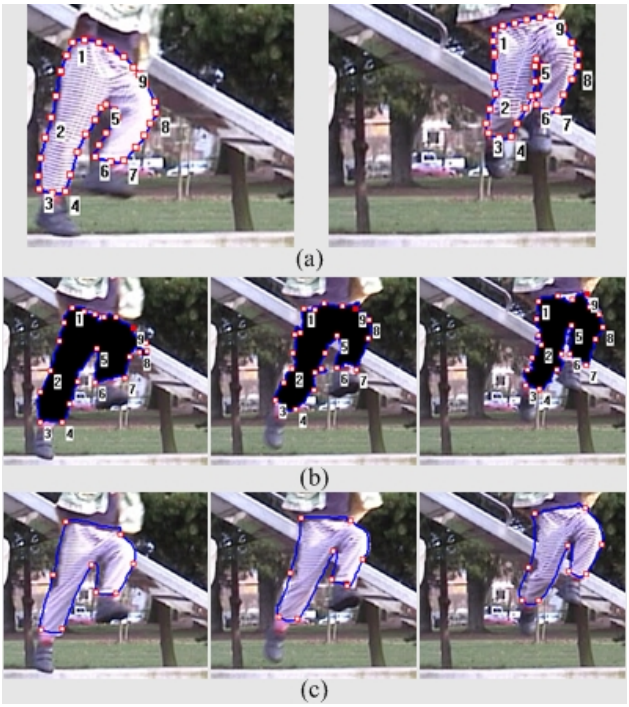

Figure 6: Mean shift constrained keyframe interpolation algorithm. (a) The user outlines the trousers on the 12^{nd} and 25^{th} frames. (b) Mean shift constraints on the 15th, 18th and 22nd frame, from left to right. Black regions represent the merged volume S^* and the boundaries represent the mean shift constraints $L_{ms}(t)$. (c) Final interpolated results, $L_s(t)$, balancing mean shift constraints and smoothness.

4.1 Interactive Specification of Semantic Regions

Our next goal is to allow a user to provide input to collect segments into more semantically meaningful groups, for example, the girl's pants in the monkey bars video. The segments derived directly from

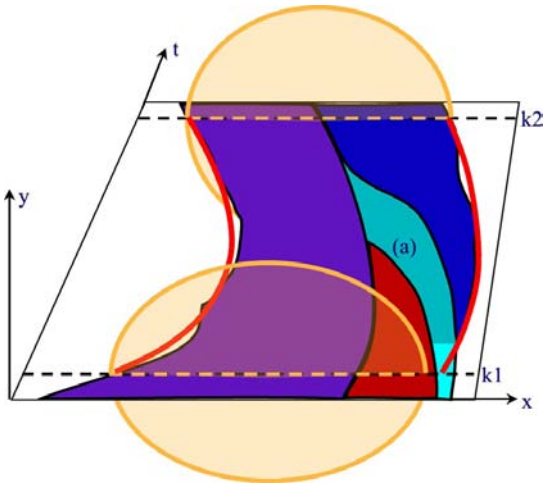

Figure 7: Illustration of mean shift guided interpolation on a 2D spatio-temporal slice. Loops on keyframes appear here as yellow ellipses on x, y planes at times $k1$ and $k2$. The red area represents a segment surrounded only by the loop at $k1$, and the blue area a segment surrounded at $k2$. The purple area shows a segment surrounded by both. The region (a) is also included in the final union since the majority of its pixels (shaded darker) lie between surrounded segments of the simple union. The boundaries of the semantic region are smoothed in a final optimization step as indicated by the red lines.

the mean shift procedure will typically be too low level to have such semantic meaning.

A user interface is provided to outline semantic regions in keyframes. The outlines drawn by the user indicate which low-level segments should be merged together to form a high-level semantic region. The mean shift results provide the temporal information needed to spread the user's region indication on keyframes to in-between frames. This avoids tedious frame-by-frame rotoscoping.

The number of keyframes needed may vary according to the intensity of the motion. Generally, the more intensive and complex an object moves, the more keyframes are required to achieve satisfying interpolated results in in-between frames. However, even for the girl's legs in the monkey bars sequence which is full of complex motions, keyframes are typically needed only every ten or fifteen frames.

In addition to the outlines, we ask the user to indicate a few key points on each outline to help later with interframe correspondence.

4.2 Mean Shift Guided Interpolation

Suppose the user draws two loop boundaries $L(k1)$ and $L(k2)$ on two keyframes $k2 > k1$. Each loop encircles a set of segments, $S(k1)$ and $S(k2)$, that extend forward and backward in time. A segment is considered inside the loop if the majority of pixels on the keyframe lie inside the user drawn outline. We can take the union of the two sets $S(k1, k2) = S(k1) \cup S(k2)$ to get a first approximation of the semantic region being specified between the keyframes. This union may also encompass other segments that do not cross either $k1$ or $k2$. These are added to the union to create the augmented union $S^*(k1, k2)$. A segment is considered encompassed by other segments through the following two step process:

- For each frame, t, $k1 < t < k2$, pixels that are fully surrounded by pixels contained in $S(k1, k2)$ are marked.
- Each segment for which a majority of its pixels are marked is considered encompassed and added to $S^*(k1, k2)$.

By slicing the union, S^*, at each frame time we get a series of in-between boundaries $L_{ms}(t), t = k1 + 1, ..., k2 - 1$ between $L(k1)$ and $L(k2)$. Typically, the set of boundaries $L_{ms}(t)$ exhibit significantly noisy boundaries. The mean shift segmentation is sensitive to small perturbations on the surfaces of the segments. This results in noisy boundaries between regions due to high frequency detail occurring in the images or resulting from video interlacing. We thus incorporate smoothness objectives into the interpolation procedure, both spatially and temporally as described next.

1. The user inputs loops $L(k1)$, $L(k2)$, and *keypoints* defining the correspondence between them, as shown in Figure 6a.
2. Compute a simple linear interpolant $L_s(t), t = k1+1, ..., k2-1$ by direct linear interpolation of loops, $L(k1)$ and $L(k2)$, including the keypoints between $k1$ and $k2$.
3. Compute the mean shift loops $L_{ms}(t), t = k1 + 1, ..., k2 - 1$ by computing the underlying merged volume $S^*(k1, k2)$ and slicing it successively along the time axis. This results in a finely tesselated polygon at each frame time without keypoints.
4. Build correspondences between $L_s(t)$ and $L_{ms}(t)$ by using a shape correspondence algorithm to transfer the keypoints from the user defined loops to the mean shift defined loops. Results are shown in Figure 6b.
5. Using $L_s(t)$ as a starting guess, iteratively adjust vertex positions of $L_s(t)$ to minimize a weighted sum of the difference from $L_{ms}(t)$ and a *smoothness energy*. Results appear in Figure 6c.

Step 4 is a general shape matching problem used to establish which points on the mean shift defined loops correspond to the keypoints defined by the user. We use the robust "shape context" approach [Belongie et al. 2002]. This method defines a descriptor, the *shape context*, to any reference point on a polygon. The shape context captures the statistical distribution of all points on the polygon relative to the reference point. This offers a globally discriminative characterization. Corresponding points on two similar shapes will have similar shape contexts, and all the correspondences are solved as an optimal assignment problem by the shortest augmenting path algorithm proposed in [Jonker and Volgenant 1987].

In our case, at each frame time t, we compute the *shape context* for each keypoint on $L_s(t)$ relative to all points on the loop. We also compute a *shape context* for all points on the mean shift defined loop $L_{ms}(t)$. The algorithm then finds the mean shift loop points that have a similar *shape context* to the keypoints on $L_s(t)$ and assigns these points as keypoints on the mean shift loops.

In step 5 we put the problem into an optimization framework and solve it iteratively. Five points per keypoint are distributed evenly along $L_s(t)$ and $L_{ms}(t)$. This provides two finely tesselated polygons with the same number of vertices. These vertices are denoted as $P_s^i(t)$ and $P_{ms}^i(t)$, $i = 1, ..., N_p$. We define a spatio-temporal smoothness energy as:

$$E_{smooth}(t) = \sum_{i=1}^{N_p-1} \left\{ \left\| \overrightarrow{P_s^i(t+1), P_s^{i+1}(t+1)} - \overrightarrow{P_s^i(t), P_s^{i+1}(t)} \right\|^2 \right\} + \sum_{i=1}^{N_p} \left\{ \left\| \overrightarrow{P_s^i(t), P_s^i(t+1)} - \overrightarrow{P_s^i(t-1), P_s^i(t)} \right\|^2 \right\} \quad (1)$$

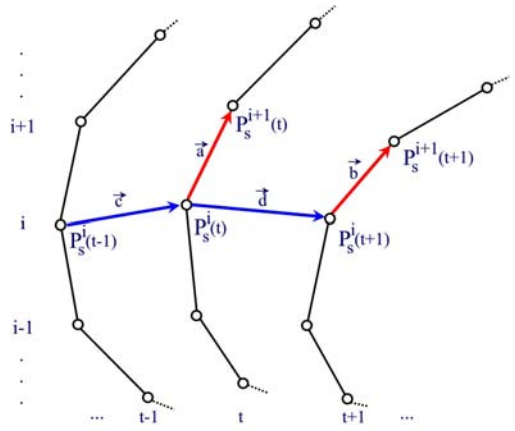

Figure 8: Illustration of the two terms of the spatio-temporal smoothness energy in Equation 1. The diagram shows the evolution of a portion of a loop over three time steps, $t-1$, t, to $t+1$. Each smoothness term is an L2-norm of the difference of two corresponding vectors. The first *shape* term is the difference between vectors $\vec{a}$ and $\vec{b}$. The second *temporal* term is the difference between vectors $\vec{c}$ and $\vec{d}$.

The first term tries to keep the boundary shape the same from one frame to the next. The second term tries to minimize the 2^{nd} finite difference of a single point through time. The two terms are illustrated in Figure 8.

A mean shift difference energy is defined simply as the sum of squared offsets of the current guess, L_s, from the mean shift boundaries, L_{ms}:

$$E_{ms}(t) = \sum_{i=1}^{N_p} \left(\left\| P_{ms}^i(t) - P_s^i(t) \right\|^2 \right) \quad (2)$$

The complete objective function for minimization is then defined as the weighted sum:

$$E = \sum_{t=k1+1}^{k2-1} \left[E_{smooth}(t) + w_{ms} E_{ms}(t) \right] \quad (3)$$

where w_{ms} weights the mean shift objective relative to the smoothness objective. A typical setting for w_{ms} is 3.0 as we do not want to oversmooth the mean shift results.

The locally optimal positions for the $P_s^i(t)$ are achieved by iteratively perturbing each point to lower the overall smoothness energy. This continues until a local minimum is achieved. We illustrate the constrained interpolation on a simplified diagram of a spatio-temporal slice (see Figure 7).

5 Stylized Rendering

Given the pixelized representation of the semantic regions, we then convert them to 3D polyhedral surfaces. The new representation serves two purposes: we can smooth the reconstructed surfaces further using traditional object smoothing operations; and in addition, surface reconstruction makes the computation of *edge sheets* possible, which are used to render temporally coherent strokes. We also show the use of *stroke sheets* within regions to allow modification of the region interiors.

A side benefit of the continuous representation, which we do not explore here, is that the resulting shapes are now resolution independent in both space and time. Thus, final rendering can be performed at any spatial or temporal resolution and compression/transmission methods no longer need to deal with discrete frame times.

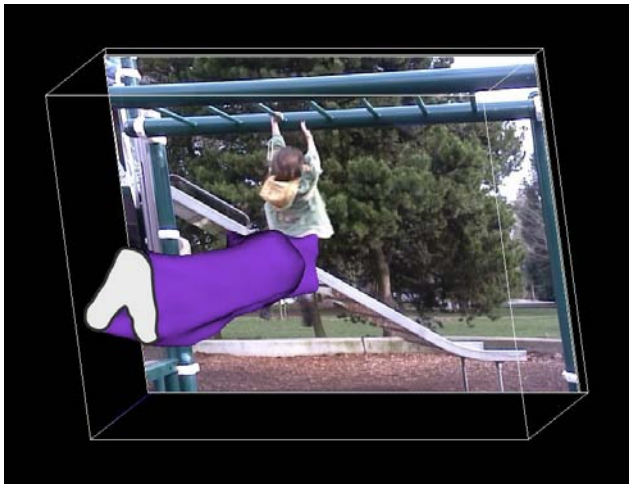

Figure 9: A smoothed semantic region sliced at time t.

5.1 Semantic Region Surface Construction

We use the marching cubes algorithm [Lorensen and Cline 1987] to convert the pixelized data into surface data resulting in polygonal surfaces separating the semantic regions. In addition to the surface geometry, each semantic region is annotated with a color, and a region edge importance, I_r, supplied by the user. I_r is set between 1 and 0. The edge importance of the background in our examples is by default set to 0 but could be set otherwise.

We then iteratively smooth the region volumes. Each vertex is perturbed along the two spatial dimensions, x and y. One smoothing step moves each vertex's spatial position to the average of its current position and 0.25 times the mean (x, y) position of all its connected neighbors. Although the time component of a vertex remains unchanged, all connected neighbors including those at adjacent frame times are considered when perturbing the spatial position. Gaps between regions are avoided since regions share the set of vertices forming their separating walls. We have also not experienced any self intersection region surfaces, we believe due to the fact that the original vertices are regularly spaced from the video lattice.

The smoothed regions can easily be rendered as solid colored polygons at any time t by intersecting them with a plane perpendicular to the time axis (Figure 9).

5.2 Edge Sheets

We also want to add solid strokes to the final rendering much like inked lines in a drawing. Selecting lines and their location on a frame-by-frame basis causes a lack of temporal coherence. To avoid this, we construct a set of smooth two dimensional sheets, or *edge sheets* embedded in the 3D video volume. We slice these sheets at each frame time to extract a curved line for rendering.

5.2.1 Edge Sheet Construction

Edge sheets are derived from the surface representations of the 3D semantic regions. Each pair of adjacent regions will share one or more sections of their surfaces. Sets of contiguous shared triangles between each pair of regions are copied into their own vertex/edge data structure. This forms two dimensional edge sheets embedded

in the 3D video volume. Small sheets containing less than a user set number of triangles are discarded.

The remaining polygonal sheets are smoothed in two ways: 1) The boundaries are low pass filtered to avoid jagged edges that could cause temporal artifacts, and 2) internal vertex positions are averaged with their adjacent vertices to provide geometric smoothness.

An *edge importance* value denoted as I_e for each sheet is set to either the max or the difference of the two region importance values, I_r, of the semantic regions it separates. I_e is compared to a user set threshold between 0 and 1 to decide if the sheet should be used to generate edge strokes at render time.

5.3 Rendering Edge Sheets

Edge sheets are two dimensional sheets embedded in the 3D video volume. When sliced by a plane at a particular frame time, the edge sheets produce smooth curves that approximately follow the surfaces of the regions. The smoothing step may pull some edges slightly away from the exact boundary between colored regions but this provides a good balance between stroke smoothness and region shape. Figures 11 and 12 show examples of the inclusion of edge sheets.

Rendering a sheet at some time t involves first intersecting the sheet with a plane at time t to produce a curve. The curve can then be drawn with a number of styles. In [DeCarlo and Santella 2002], the overall thickness was set simply by the length of the stroke in the 2D frame and had a profile that tapered it at its ends. We begin in a similar way by defining a basic *style* for the line that defines its profile along its length in the spatial domain parameterized by arc length. Many drawing and sketching systems provide such a choice, such as [Hsu and Lee 1994].

In our case, we have the added information provided by the edge sheet as a whole that we can leverage to modify the stroke color and thickness or other properties. In addition to an edge's length in a particular frame, the edge also has a limited duration in time. We will use the current time t relative to the beginning, t_s, and end, t_e of an edge sheet's existence to modify the edge thickness. More specifically, as $(t - t_s)/(t_e - t_s)$ varies from 0 to 1, the thickness will grow to a maximum at 0.5 while being thinner at the beginning and end of its lifetime.

At each vertex, the sheet also has an implied normal, $N = (N_x, N_y, N_t)$, taken as the average of the normals of adjacent triangles. The direction of the normal is taken to point outwards from the region with the higher I_r.

N_t indicates the rate of an edges's motion across the frame. In particular, a positive value of N_t indicates a *trailing edge* that we will thicken during rendering, and a negative N_t indicates a *leading edge* that is thinned.

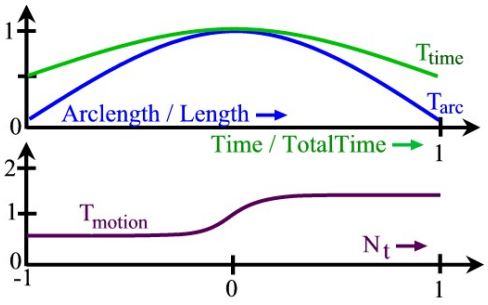

Figure 10: Variables to determine edge thickness.

The sum effect on thickness of a point on an edge based on its position along its arclength, the frame time relative to its lifetime, and the edge's motion is summarized as

$$Thickness = T_{base} * T_{arc} * T_{time} * T_{motion} \quad (4)$$

T_{base} is set by the user and represents the thickness of the center of a still edge at the middle of its existence in time. The other terms vary as shown in Figure 10. The formulas of these terms are given in the Appendix. As the graphs imply strokes thin at their ends both in space and time.

Finally, the spatial components of the normal (N_x, N_y) can be used to shade the edge based on a dot product with a virtual *light direction* given by the user. We'll denote this dot product D_l. For example, we may wish to make edges facing the upper right brighter and those facing down and to the left darker. An example can be seen in Figure 11(right).

Figure 11: Left: solid regions with no edges. Center: Black edges modulated by thickness. Right: Edges further modulated by an implied lighting direction and thickness modified by motion.

For those who prefer to visualize the edge sheet as whole, one can imagine a curved sheet that is thin along all its edges. It is thickest in the center both along its spatial and temporal extent. It also tends to be thicker in portions that face along the time axis as opposed to facing backwards in time. Finally, the whole sheet is lit from an infinite point source in some $(x, y, 0)$ direction.

5.4 Filling the Region Interiors

In addition to drawing edges, we also fill the interiors of slices of the regions. There are three ways the interiors can get filled; by direct pixel coloring, dividing the regions into subregions and then coloring, or by filling the regions with paint-like strokes. In fact, all three can be combined through standard compositing if desired.

5.4.1 Pixel Coloring

There are three colors (in RGB space) associated with each pixel; the original pixel color, the average pixel color within a segment as determined directly by the mean shift procedure, and a user defined color for the larger semantic regions defined by the interaction procedure. These three colors can be combined as a weighted combination. Such a combination can be seen in the facial region in Figures 1 and 12. Note that a full weight on any of the three will default to the original video, the mean shift result, or a solid colored shape respectively. Finally, any other color space transformation could be used to modify the result such as brightness and saturation, and hue controls.

5.4.2 Subregions

In some cases we have found it useful to allow the user to define their own subregions. The same interface used to define the semantic regions is used to allow the user to draw subregions on keyframes,

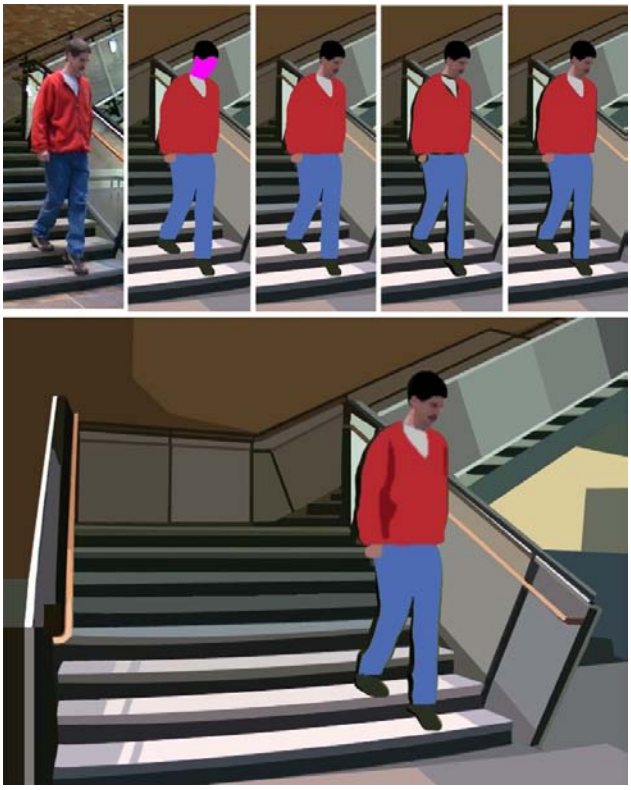

Figure 12: Construction of the final frame. Top row from left to right: original frame, regions as solid shapes, head rendered with segmentation result, edges on all region boundaries, edges only at important boundaries based on edge importance for each region. Bottom image: final frame that now includes subregions within the shirt to show shadowing.

Figure 13: User adding a paint stroke at a keyframe and the resulting strokes in a frame of final animation.

with the only exception being that the result is constrained to lie fully within a specified semantic region. This allows a second color region within a larger region in the final rendering. This was used for the shadowing within the shirt seen in Figure 12.

5.4.3 Stroke Sheets

We can also leverage the mean shift results to allow the user to lay down paint strokes within regions at keyframes (see Figure 13) and

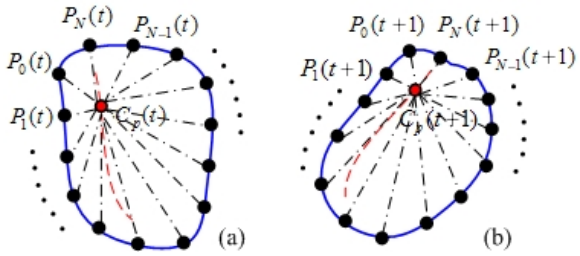

Figure 14: Illustration of "flowing" a point on a stroke from one frame to the next, as shown in (a) and (b).

have them automatically interpolated to create temporally coherent motion of the strokes. In much the same way that the edge sheets are created, we create 2D *stroke sheets* that are embedded fully within a semantic region. A user draws strokes within semantic regions on a keyframe, defining the stroke skeleton, color, and style [Hsu and Lee 1994]. On subsequent keyframes intersecting the same semantic region, the user must draw the new skeletons of each stroke.

Between keyframes, k_1, k_2, strokes are *flowed* forward in time from k_1 and backward in time from k_2. We sample each stroke at 15 points along its length. Each sample, $C_p, p = 1..15$, is separately interpolated with the corresponding stroke points on other keyframes. We now consider one such point C_p on the stroke (see Figure 14). From the semantic region interpolation we have N sample points along the boundary of the semantic region in which the stroke lies, denoted as $P_i(t), i = 1, ..., N$. By computing the distances $d_i(t)$ between $C_p(t)$ and the $P_i(t)$, we get a vector of weights $< w_i(t) = (1 - d_i(t)/d_{max}(t))^2 + \epsilon >$. These are then normalized by dividing each by the sum of the weights.

For the next frame, we examine the motion of points along the boundary. We then compute the motion of each stroke point as a weighted average of the motion of the boundary points using the normalized $< w_i(t) >$ as the weights. Thus

$$C_p(t+1) = C_p(t) + \sum_{i=1}^{N} w_i(t) \cdot (P_i(t+1) - P_i(t))$$

Each stroke point along the stroke is processed in the same way. For each frame time from k_1 to $k_2 - 1$, new weights are computed and the points are iteratively flowed forward.[1]

In the same way the strokes are flowed backwards from k_2. The final position is a linearly weighted average of the forward and backward flow, with the first weight dropping from 1 to 0 as time goes from k_1 to k_2 and from 0 to 1 for the reverse.

The interpolation of the strokes creates a two dimensional stroke sheet lying within the semantic region (although the final rendering of the strokes may overlap region boundaries). These sheets are sliced at a time t to provide a skeleton for a stroke to be rendered. Figures 1 and 13 show an example frame of rendering the stroke sheets in a watercolor style.

5.5 The Background

The background is defined as a single semantic region including all portions of the video lying outside the user defined semantic regions. The background can be filled just like any other semantic region.

[1]The iterative radial basis weights, $w_i(t)$, could be replaced in theory by a single set of barycentric coordinates computed at the keyframes. However, we do not know of a general barycentric formula for irregular non-convex polygons. Barycentric coordinates for most points within non-convex polygons can be found in Floater [2003].

In our examples, the camera was still and the foreground objects (people) moved across the field of view. In these cases it is quite easy to extract a constant background frame with a median filter of each pixel through time. This can then be rendered as is or modified with a paint program or simply replaced. The foreground regions are then rendered over this background. A number of different background styles can be seen in our examples.

5.6 Performance and Interface Timings

The processing of a video through the mean shift procedure to optimizing for the semantic regions given user input, to creating the edge sheets and rendering the final frames requires varying amounts of processing and user time. The automated mean shift procedure runs overnight on a ten second (300 frame) video shot (please see the appendix for some details). The two examples required about $1\frac{1}{2}$ hours of user time to draw the keyframe loops. The paint strokes required about an additional $\frac{1}{2}$ to complete. Given the user's loops, and/or paint strokes on keyframes, the interpolation takes about 1 second per in-between frame thus these can be carried out iteratively in an interactive setting. The final optimization for the semantic regions and computation of the edge and stroke sheets requires about one half hour. Rendering the final frames takes about 1-2 seconds each.

6 Conclusions

We have presented a system for interactively transforming video to a cartoon-like style. Our system provides a possible means to overcome the main challenge of providing temporal stability by leveraging a new mean shift method applied to video data. We have shown how the mean shift results together with the artist's input can provide a variety of non-photorealistic styles.

We are currently working a number of extensions and enhancements to the Video Tooning system. There are many more stylistic choices to explore for rendering the solid regions, edges, and paint strokes. We also hope to explore the use of layers, as were exploited in the movie *Waking Life*, particularly if the source video is derived from hand held cameras and thus subtle parallax issues are present.

As in any research system, there are many enhancements to the user interface that should be added. Although the current system provides methods to glue segments together to create semantic regions, it is possible that a segment may need to be cut. We intend to add this capability.

We are also working on a vectorized encoding of the result. The monkey bars video contains about 120,000 3D vertices vs. approximately 120 million pixels in the original video, a ratio of 1:1000. We hope to leverage mesh simplification methods to lower this ratio further. There is clearly a lot of coherence to leverage for compression, plus the added benefit of resolution independent encoding.

The combination of 3D segmentation, an efficient semantic abstraction interface, edge sheets and stroke sheets provides a very powerful system for the stylization of video. We look forward to experimenting on new video sources, with new styles, and tackling many practical aspects of the Video Tooning approach.

Acknowledgements

A great thanks is due to the reviewers for helping to make this a better paper than the one originally submitted. We also want to acknowledge Lena Joesch-Cohen who performed the monkey bar motion and drew the background in the title page image.

References

AGARWALA, A. 2002. Snaketoonz : A semi-automatic approach to creating cel animation from video. In *Proceedings of NPAR 2002*.

BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence 24*, 4, 509–522.

CHRISTOUDIAS, C., GEORGESCU, B., AND MEER, P. 2002. Synergism in low-level vision. In *Porc. of 16th International Conference on Pattern Recognition*, 150–155.

COLLOMOSSE, J. P., ROWNTREE, D., AND HALL, P. M. 2003. Stroke surfaces: A spatio-temporal framework for temporally coherent non-photorealistic animations. *University of Bath, Technical Report CSBU 2003-01 (June 2003)*.

COMANICIU, D., AND MEER, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence 24*, 5, 603–619.

COMANICIU, D., RAMESH, V., AND MEER, P. 2000. Real-time tracking of non-rigid objects using mean shift. In *Porc. of IEEE Conf. on Comp. Vis. and Pat. Rec (CVPR00)*, 142–151.

DECARLO, D., AND SANTELLA, A. 2002. Stylization and abstraction of photographs. In *Proceedings of SIGGRAPH 2002*, 769–776.

DEMENTHON, D. 2002. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *Porc. of Statistical Methods in Video Processing Workshop*.

FLOATER, M. S. 2003. Mean value coordinates. *Computer Aided Geometric Design 20*, 19–27.

FUKUNAGA, K., AND HOSTETLER, L. 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Information Theory 21*, 32–40.

HERTZMANN, A., AND PERLIN, K. 2000. Painterly rendering for video and interaction. In *Proceedings of NPAR 2000*, 7–12.

HERTZMANN, A. 2001. Paint by relaxation. In *Proc. Computer Graphics International 2001*, 47–54.

HOCH, M., AND LITWINOWICZ, P. C. 1996. A semi-automatic system for edge tracking with snakes. *The Visual Computer 12*, 2, 75–83.

HSU, S. C., AND LEE, I. H. H. 1994. Drawing and animation using skeletal strokes. In *Proceedings Computer Graphics (ACM SIGGRAPH)*, ACM Press, 109–118.

JONKER, R., AND VOLGENANT, A. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing 38*, 325–340.

KASS, M., WITKIN, A., AND TERZOPOULOS, D. 1987. Snakes: Active contour models. *International Journal of Computer Vision 1*, 4, 321–331.

KLEIN, A. W., SLOAN, P.-P. J., FINKELSTEIN, A., AND COHEN, M. F. 2002. Stylized video cubes. In *Proceedings of SCA 2002*.

KORT, A. 2002. Computer aided inbetweening. In *NPAR 2002: Second International Symposium on Non Photorealistic Rendering*, 125–132.

LINKLATER, R. 2001. Waking Life DVD. *Twentieth Century Fox Home Video*.

LITWINOWICZ, P. 1997. Processing images and video for an impressionist effect. In *Proceedings of SIGGRAPH 1997*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, ACM, 151–158.

LORENSEN, W., AND CLINE, H. 1987. Marching cubes: a high resolution 3d surface reconstruction algorithm. In *Proceedings of SIGGRAPH 1987*, 163–169.

WANG, J., THIESSON, B., XU, Y., AND COHEN, M. F. 2004. Image and video segmentation by anisotropic kernel mean shift. In *Proc. European Conference on Computer Vision, 2004*.

Appendix

Formulas of variables to determine edge thickness

The formulas of three components T_{arc}, T_{time} and T_{motion} in Eqn. 4 are given as follows:

$$T_{arc} = 1 - 3.2 \cdot (s - 0.5)^2, \quad s \in [0, 1]$$

$$T_{time} = 1 - 2 \cdot (s - 0.5)^2, \quad s \in [0, 1]$$

$$T_{motion} = \begin{cases} -0.5 & : \ -1 \leq N_t < -0.2 \\ 0.5 + sin(N_t \cdot \frac{5\pi}{2}) & : \ -0.2 \leq N_t < 0.2 \\ 1.5 & : \ 0.2 \leq N_t \leq 1 \end{cases}$$

Anisotropic Kernel Mean Shift Pseudocode

Some terms:

x_i, x_j	pixels located in 6D space (position, time, color)
x_i^r	the pixel location in the 3D Luv color space
x_i^s	a pixel location in the 3D spatial domain (position, time)
h_0^s	an initial spatial bandwidth, i.e., the distance within space considered nearby
h^r	the color bandwidth, i.e., the distance in Luv color space considered nearby
H_i^s	a 3D spatial kernel that defines nearby-ness The kernel is represented by a 3×3 covariance matrix.
I	a 3×3 identity matrix
$M(x_i)$	the *mean shift point* associated with x_i
$M_v(x_i)$	the *mean shift vector* associated with $M(x_i)$

Initialization

1. Data and kernel initialization.
   - Transfer pixels into multidimensional (5D for image, 6D for video) feature points, x_i.
   - Specify initial spatial bandwidth h_0^s. We used h_0^s = 6 pixels.
   - Associate kernels with feature points, initialize means to these points.
   - Set all initial bandwidth matrices in the spatial domain as the diagonal matrix $H_i^s = (h_0^s)^2 I$. Set the bandwidth in the range domain as h_0^r. We set h_0^r to be a color distance of 6 in an Luv color space.
2. For each point x_i, determine anisotropic kernel and related color radius:
   - Search the neighbors of x_i to get all the points x_j, $j = 1, ..., n$ that satisfy the constraint:

$$|(H_i^s)^{-1/2} (x_j - x_i)| < 1 \tag{5}$$

   and

$$\frac{|x_i^r - x_j^r|}{h^r} < 1 \tag{6}$$

   where x_i^r and x_j^r are the pixel colors.
   - Update the spatial bandwidth matrix H_i^s as:

$$H_i^s \leftarrow \frac{\Sigma_{j=1}^n \left\| \frac{x_i^r - x_j^r}{h^r} \right\|^2 (x_j^s - x_i^s)(x_j^s - x_i^s)^T}{\Sigma_{j=1}^n \left\| \frac{x_i^r - x_j^r}{h^r} \right\|^2} \tag{7}$$

   - Optionally modulate H_i^s and color tolerance h^r to create larger segments for static objects. For details, see [Wang et al. 2004].
3. Repeat step (2) a fixed number of times (typically 3).

The Mean Shift Procedure

4. Associate a mean shift point $M(x_i)$ with every pixel, x_i, and initialize it to coincide with that point. Repeat for each $M(x_i)$
   - Determine the neighbors, x_j, of $M(x_i)$.
   - Calculate the mean shift vector summing the derivative of the Epanechnikov color kernel over the neighbors:

$$M_v(x_i) = \frac{\Sigma_{j=1}^n (x_j - M(x_i)) \left\| \frac{M(x_i^r) - x_j^r}{h^r} \right\|^2}{\Sigma_{j=1}^n \left\| \frac{M(x_i^r) - x_j^r}{h^r} \right\|^2} \tag{8}$$

   - Update the mean shift point:

$$M(x_i) \leftarrow M(x_i) + M_v(x_i) \tag{9}$$

   until $M_v(x_i)$ is less than a specified epsilon.
5. Merge pixels whose mean shift points are approximately the same to produce homogenous color regions.
6. Optionally, eliminate segments containing less than a given number of pixels.

Implementation Notes on Anisotropic Kernel Mean Shift

Mean shift segmentation is a global optimization problem and by its definition potentially requires access to the pixels of the full video equation 8. Memory constraints led us to approximate this process. We load the first 15 frames of the video into memory and allow the associated mean shift points to move within these frames. We record the segmentation results for the first 10 of the 15 frames and then jump forward 10 frames. Thus the second block to be processed consists of frames 11 - 25. For this and subsequent blocks we enforce the segmentation results for the first frame (e.g., frame 11) to remain the same as the previous block to provide temporal consistency. This results in an approximately optimal mean shift solution with few visible artifacts.

Similar to the basic mean shift algorithm, the anisotropic kernel mean shift also have two major parameters: the initial spatial radius h_s and the color radius h_r. In our experiments, we examine one second test cases on half resolution videos and vary h_s between 4 to 7, h_r from 3.5 to 6.5 to get the best results (the examples shown used values of 6 for both). These simple tests still take a few minutes each.