Tornado Motion Synthesis from Video Footage

Yootai Kim* Jinho Lee† Raghu Machiraju‡

The Ohio State University

Abstract

In this paper we present a novel method to generate 3D animation of tornados from video clips taken from a single camera. Our source of input data is a variety of tornado video clips of widely-varying fidelity available on the world wide web. By extracting the silhouette of the foreground figure and using a model-based approach, the problem is reformulated as a local, optimized search of the shape space. The silhouette of the figure from the captured video is compared against the silhouette of a synthetic revolute using a pixel-by-pixel cost function. By using temporal coherence, physical constraints, and knowledge of tornado motion, the motion can be reconstructed. Additionally, we use this reconstruction to synthesize animations of the captured phenomena. Our main contribution is a method to synthesize a motion of highly dynamic objects like the tornado from a single video source. In addition, we employ simple yet robust 2D shape descriptors to reconstruct the shape and form of the tornado.

1 introduction

The tornado is one of the most spectacular natural phenomena on the planet. The highest local wind velocities on earth are observed in tornados. These columnar whirlwinds with a diameter up to 500 m and a height of about 3 km, reach rotational velocities up to 200m/s [16]. Given their ability to inspire amazement, they are often employed as special effects in many feature films such as *Twister (1996)*, *X2 (2003)*, and *The day after tomorrow (2004)*. *Twister* was the first feature film successfully generating realistic tornado animations through the exploitation of modern computer graphics hardware and custom-made software [5]. However, without any prior model, creating a realistic animation of a tornado requires a great amount of manual work. Though there have been a significant number of efforts directed towards the modeling of natural phenomena like smoke, fog, cloud, water, and fire, efforts which model tornados are relatively rare if non-existent.

In this paper we present a method to create 3D animations of tornados from video footage recorded on a single camera. There has been much reported work on reconstructing 3D models of solid objects from image sequences in computer vision and computer graphics. However, there is again a relatively small amount of work available that describes the reconstruction of dynamic natural phenomena from 2D image sequences.

The source of our data is a repository of video clips available on the world wide web recorded by various users. The quality of the video clips varies from very good to poor and are often of different resolution and duration. The illumination is often poor and the camera calibration is unknown. A tornado shape is reconstructed for each frame of the original video using its corresponding silhouette images. For this purpose we employ a silhouette-based inverse rendering framework [13]. A cost function is minimized for each frame, which measures the difference between a model-generated silhouette image and the input silhouette image. After extracting a series of 2D tornado structures from images, we construct 3D tornado surfaces leveraging knowledge of problem domain. Later, we synthesize a keyframe animation of successive 3D tornado reconstructions. We render the surface of tornado using textured particles and achieve a swirling effect by rotating the particles along with the surface.

Our contribution to the literature is two-fold. We report one of the first attempts to synthesize highly dynamic objects like the tornado from a single video source. Secondly, we employ simple yet robust 2D shape descriptors to reconstruct the shape and form of the tornado.

In Section 2 we describe relevant previous work on the image based synthesis of natural phenomena. Then, in section 3 we describe our tornado model and formulate its use for extracting the shape of tornado from silhouette images. Our method for creating a 3D animation of a given tornado is described in Section 4. We present the results of applying our methods to some input tornado videos in Section 5. In

*e-mail: yootai@cse.ohio-state.edu
†e-mail:leeji@merl.com
‡e-mail: raghu@cse.ohio-state.edu

Section 6 we provide discussion and future work.

2 Related Work

There are many methods of reconstructing a version of natural pheonmena from an observed sequence of 2D images. Wang and Zhu [22] use a generative model to analyze textured motion to learn about basic patterns motion. After weeding out photometric and geometric influences, they use the underlying dynamics of observed motion to synthesize new motion. Schödl et al. [20] present a new paradigm of representation namely video texture. They capture cyclic patterns in a relatively short video clip, which are very common among video depictions of natural phenomena (e.g. fire, waterfall), and generate infinitely varying dynamic video. Although video textures were also applied to applications in 3D through the use of view-morphing techniques, these methods are essentially two dimensional. Hasinoff and Kutulakos [8] use flame sheet decomposition method to reconstruct fire as a 3D density field from a set of images from two or more views. The method is based on the observation that a pair of photos of a semi-transparent scene defines a unique density field, called a flame sheet. Fire is reconstructed as the convex combination of sheet-like density fields. Dobashi et al. [3] also use a volume density field to reconstruct large-scale cloud-like objects such as a typhoon from satellite images. Their approach search an optimal configuration of metaballs which leads to the closest match to the given satellite image. They also generate an animation of clouds by interpolating the parameters of the constitutive metaballs using user-specified flow curves defined between two subsequent images.

Related to our framework for extracting tornado structure using silhouette images, there exists work which applies similar techniques to problems of motion analysis and synthesis. Internal/external camera parameters are recovered using exact geometry information of an object and its silhouette images [9, 14]. In [21], a method is presented to search the optimal configuration of human motion parameters by applying a silhouette/contour likelihood term. In [13, 17], the shape parameters of a 3D face model are recovered from multiple silhouette images through an inverse silhouette rendering framework.

To the best of our knowledge, there is very little published research for creating 3D tornado animations from video clips. In the feature film *Twister*, tornados are modeled using conical solids. The surface of the tornado consists of multiple layers of particles, which are rendered individually with a variety of light sources [5]. The deformation of the tornado surface is selected by animators driven by the need to obtain certain visual effects. In our approach, the overall shape and surface deformation is extracted from the real video clips automatically.

3 Extracting Tornado Structure

In this section we present our tornado model and a method to extract the structure of a tornado from a video sequence using the proposed model. Before describing our techniques we first describe some of the challenges we faced and also explain the choices and trade-offs we made to ward off the challenges.

3.1 Modeling a Tornado

Shape: The tornado can assume many forms. The shape is often amorphous and enormous. Besides the primary structure, there often exists secondary structures that are not easy to define given the high amount of turbulence the tornado generates. For this effort, we limit ourselves to a single primary structure. An essential primary structure of a tornado is a large vortex plume rising above the ground [15], therefore exhibiting a strong geometric signature in the form of a funnel. Moreover, the strong swirling wind field is mostly confined to the funnel-shaped region. Therefore, a characterization of the funnel shape will suffice and any additional visualization or animation can be limited to particles or the field inside or around the funnel.

Segmentation: Given the often poor and highly diffused illumination that exists in many of the video clips it is often a challenge to eliminate the background and extract the tornado. Also, artifacts in the form of buildings, electric poles, etc. make the extraction process even more tenuous. The silhouette extraction is performed by foreground/background segmentation. By inspecting the histograms of several frames of the target sequence, we determine the appropriate threshold of the image region containing the tornado. Then, we apply the threshold to all the frames in a video to extract the tornado silhouettes automatically.

Camera: The calibration of the camera must be precisely known to extract a high-fidelity version of the object. However, in our problem, the camera information is unknown since tornado video clips are obtained from the internet. Instead of estimating the camera parameters, we assume the projected appearance of a tornado is orthographic. This is reasonable because the distance between the camera and the tornado is large in general.

3.2 Geometric Model

We use a simple two dimensional tornado model to create segmented funnel shapes. The resolution of a tornado model is controlled by the number of segments (N_s). Figure 1 shows an instance of the model and its parameters ($\mathbf{p} = (p_1, .., p_{15})$) when $N_s = 6$. The first two parameters (p_1, p_2) describe the offset of the model along the X and Y

principal axis from the image origin respectively. (p_1, p_2) is considered the origin of model coordinate system. Parameter p_3 controls the height of a tornado. The remaining parameters $(p_4, ..., p_{15})$ describe the X coordinates of the end-points of each segment in model coordinate system. An alternative parameterization is to take the spine offsets and radii instead of end-points. We noticed a slightly better performance in the shape fitting process with the former parameterization since it provides more locality during optimization. Note that the position and length along the Y-axis of each segment can be determined by overall height of a tornado (p_3) assuming that the heights of all segments are equal and the segments are parallel to X-axis. Thus, in our model, given N_s, the total number of parameters to describe a 2D tornado shape is $2N_s + 3$. Our model is thus a collection of frusta segments that are blended together.

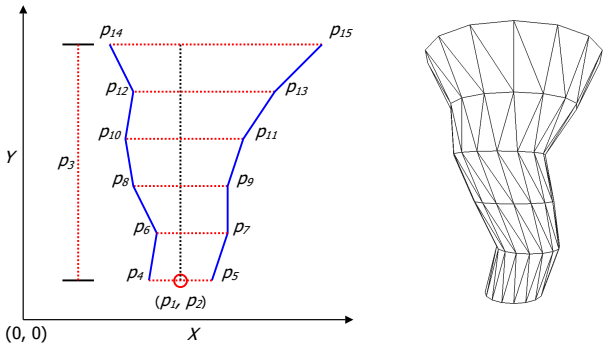

Figure 1. (left) A tornado model. (right) 3D instance of the tornado model

There are several advantages of using this simplified tornado model for extracting tornado shapes from video clips. Though it is a constrained model of small dimensionality, the model is flexible enough to conceive several segmented funnel shapes. Essentially it is a 2D model which is suitable for video-based matching especially when the depth information is hard to recover. Moreover, it can be easily converted to a 3D mesh object by generating circles parallel to the Z-axis with the segments as their diameters and pairing neighbor circles (Figure 1).

3.3 Fitting Tornado Model to Silhouette Images

We fit our model of the tornado to the silhouette images of a tornado, which are extracted from a video sequence. Model fitting is performed by minimizing a residual between an input silhouette and a model-generated silhouette image. The residual is obtained by computing the XOR of two silhouette images. Given n-dimensional model parameter $\mathbf{p}$, the cost function based on XOR operation is defined as follows:

$$F_{xor}(\mathbf{p}) = \sum_{i}^{W}\sum_{j}^{H} c[i,j] \tag{1}$$

$$c[i,j] = \begin{cases} 0 & \text{if } S_I[i,j] = S_M(\mathbf{p})[i,j] \\ 1 & \text{otherwise,} \end{cases}$$

where W is image width and H is image height, S_I is input silhouette image and $S_M(\mathbf{p})$ is silhouette image generated by a model parameter $\mathbf{p}$. To reduce the time required for evaluating the cost function, instead of creating the silhouette image of a given tornado model, we compute XOR by determining whether a pixel location is inside of a model tornado region and then checking the intensity of the corresponding pixel in the input silhouette image.

In most cases, this cost function is enough to confine the shape of tornado model to the real silhouette of a tornado. However, we need to deal with the inherent ill-posedness that is present in model-generated silhouette images. This happens because the same silhouette images can be generated by different model parameters. If we allow part of a tornado model to be outside the image boundary, some model parameters may not affect the cost function. Therefore, those parameters tend to be uncontrolled during minimization. To cope with this problem we impose constraints on our model parameters in the following way:

$$\begin{array}{lll} (a) & 0 < p_1 < W, 0 < p_2 < H, & \\ (b) & 0 < p_3 < H - p_2, & \\ (c) & 0 < p_{2n+1} - p_{2n} < fW & \text{for } n = 2..N_s + 1 \\ (d) & p_{2n+1} - p_{2n} < p_{2n+3} - p_{2n+1} & \text{for } n = 2..N_s \end{array}$$

Constraint (a) limits the location of a tornado in an image. (b) is the constraint on the height of the tornado given p_2. Constraint (c) confines the width of each segment to multiples of image width by a weight factor, f. Note that Constraint (d) is not based on the image extent but on pertinent domain knowledge: usually particles in a tornado have broader circulation (engaged in rotation) radii when they are farther above from the ground. These constraints are combined with our XOR error metric to yield another comprehensive cost function:

$$F(\mathbf{p}) = F_{xor}(\mathbf{p}) + \sum_{i}^{m} w^i F_c^i(\mathbf{p}) \tag{2}$$

$$F_c^i(\mathbf{p}) = \begin{cases} 0 & \text{if } i\text{'th constraint is satisfied} \\ 1 & \text{otherwise,} \end{cases}$$

where m is the number of constraints and w^i is the weight of i'th constraint, which controls how strictly each constraint needs to be satisfied.

We optimize the cost function (Eq. 2) using *downhill simplex method* [18], a non-linear multi-dimensional optimization algorithm. The downhill simplex method is not

very efficient in terms of the number of function evaluations. However the method does not require gradient information, which is usually hard to obtain analytically or expensive to compute numerically in very high dimensional space. Furthermore, it is robust in the presence of noise and allows the solution to be rescued from a local minima by simple repetitions of the algorithm [19].

At the very first frame of a target video sequence, the difference between model and input silhouette is large. This implies that the optimization has more chance to be stymied by local minima. With this observation, we start the optimization process with a model of lower resolution (e.g $N_s = 5$) and repeat the process 2-3 times after increasing the model resolution by a factor of two each time. In this way we fit a higher resolution model to the initial frame with a smaller chance of being trapped by local minima. Figure 2 depicts this hierarchical fitting process for a given initial frame. For fitting in subsequent frames, we do not need to lower the model resolution and just initiate the optimization process from the position and the shape found in the previous frame since contiguous frames have no abrupt changes in their tornado silhouettes. To reiterate, the result of the optimization process is the vector $\mathbf{p}$ that defines the shape and the position of the tornado.

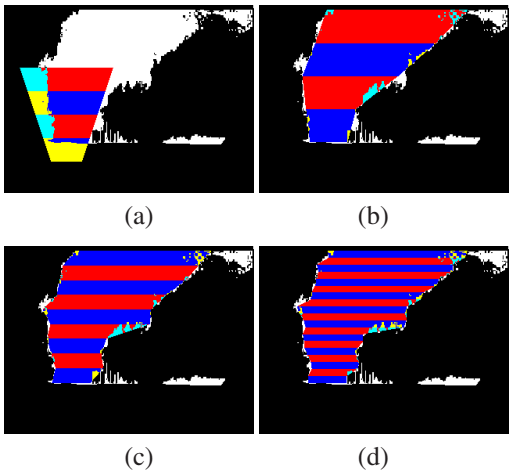

Figure 2. Initialization and optimization for the first frame: (a) initial position and shape of a tornado model, (b,c,d) fitting from lower resolution to higher resolution ($N_s = 5, 10, 20$)

4 Motion Synthesis and Animation

The synthesis of tornado motion from a single video clip is inherently ill-posed. Therefore the motion (and hence animation) may at best be superfluous and not match in appearance with the tornado under observation. It is therefore imperative that we incorporate inherent knowledge about tornados in our methods. We now list salient features of a tornado that are useful for exploitation in an animation:

- To reiterate (see Section 3) a tornado is a relatively organized structure. In its simplest form it is a single vortex plume in which air loaded with dust and debris, moves at very high speeds in an upward spiral. And certainly, we can use the extracted funnel-shape as our basic entity of animation.
- The rising air enters the vortex at its base and exits in the upper part of the funnel [7, 15].
- Tornados rotate counterclockwise with few exceptions since they are in most cases derived from cyclones in the northern hemisphere [16].
- The tornado is a swirling body of air and dust. Therefore, it can be treated as vortex and will have vortex core line where the velocity is zero [11].

We use these facts to reconstruct tornados that may seem real enough. Thus, we first build a multi-layer tornado structure to approximate a tornado as a volumetric entity. Then, we move particles along each surface layer procedurally. Therefore, the original motion of the tornado is closely emulated in this manner. The surface representation of the tornado allows users to exercise many flexible controls. Many attributes such as position, scale, and rotational speed can be controlled and animated in modern keyframe animation systems. Furthermore, the tornado wind field can be effectively simulated to align a vortex field along the medial axis curve of the tornado. Finally, sprites are used to render the tornado.

We now explain the animation reconstruction process in detail. To illustrate the method clearly, we use a single-layer tornado construction since the multi-layer extension is straightforward. First, a tornado skeleton consisting of cross sectional circles is reconstructed from the shape parameters obtained from the first frame of silhouette matching (Figure 3(a)). Since a tornado has mostly a funnel shape, we can assume axial symmetry along the medial axis curve of the tornado surface to reconstruct three dimensional coordinates of the surface. In the subsequent step, the tornado surface is created as a lofted NURBS surface passing through all cross sectional circles (Figure 3(b)). Because we do not estimate the depth information(Z-axis direction) in the extraction step, the reconstructed tornado surface has a symmetric profile in Y-Z plane. However, we found this artifact is less noticeable in practice when the tornado surface has an irregular X-Y profile and deforms in every frame. We can also add variations along the z-axis of the center points to avoid the problem. Once the tornado surface is created,

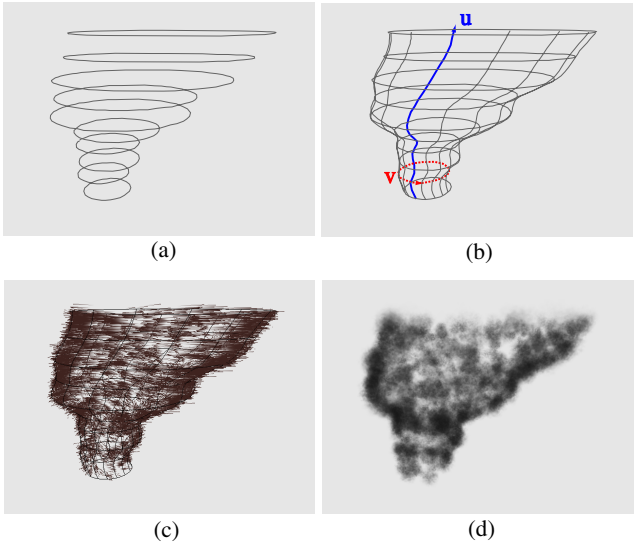

Figure 3. Animation Process Overview:(a) Tornado section curves, (b) Tornado surface: red dotted line for v direction and blue line for u direction, (c) Multi-streak particles on a tornado surface, (d) Tornado rendered using sprites

the keyframe is set for every control point of the surface for each time step. Hence, the synthesized keyframe tornado animation is created to depict the deformation of the tornado shape.

In the third major step, we emulate the spinning wind by advecting many particles around the tornado surface (Figure 3(c)). Instead of generating real wind field, we use a procedural approach. The surface coordinates are denoted by u and v which represent the tornado height direction and the circular direction respectively (Figure 3(b)). Particles are uniformly created on the surface, and the movement of each particle is constrained on the surface. Accelerating each particle in the direction of the v coordinate axis, the spinning particle motion is achieved with ease. We determine the direction of rotation based on the original video footage if possible. Otherwise, we elect the counterclockwise direction since most tornados in our videos were created in North America. Realistic tornado motion is generated by adding some upward motion in u direction.

Finally, sprites are used for efficient and realistic rendering. Several sprite images are prepared that have prerendered a dust volume based on a fractional brownian function [4]. A ray marching method is empolyed to render the hypertexture density volume [2]. The color of the volume is determined by spherically mapping the tornado texture from an input video to match the visual appearance of the original tornado. Figure 4 shows a couple of sprite images. One of the few sprite images is randomly assigned to each particle. As the tornado moves, the scale and orientation of the sprites are changed to overcome the limited frequencies covered by the sprite images and directional artifacts. Figure 3(d) shows the rendering results obtained with sprites.

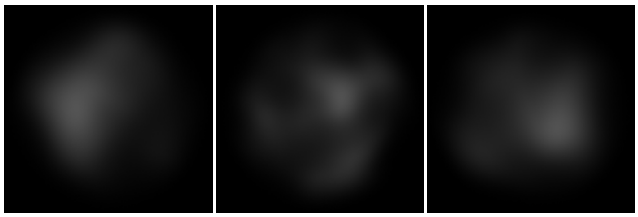

Figure 4. Sprite images

5 Results

Among the vast collection of tornado videos on the internet, some of them have very rapid camera movements while others exhibit a non-uniform range of grayscale intensities in the region of the image containing the tornado. Such video clips are not suitable for our silhouette-based shape reconstruction methods. However on the other hand, many of them exhibit relatively constant intensity for the regions of the tornado with stable or slow camera movements. We acquired several such tornado video clips and applied the proposed methods to create tornado animations based on those video clips. The resolution of the videos was often 320×240.

A video footage from a web site [10] was chosen to present our primary result. Since the camera was following the tornado, a period of sequences with a seemingly constant viewpoint was extracted from the whole sequence. The extracted sequence consists of a total of 90 frames. A set of selected frames is shown in Figure 6. The first row shows the original frames and the second row displays the extracted silhouette images. The third row depicts the results of our model fitting, while the fourth row shows the final synthesized 3D tornado animation. A frontal view is used for the rendering to compare the original motion with the synthesized one. The images shown on the same column have the same frame number. Only one frame is shown for the input video clips in Figure 7. Figure 8 shows the bird eye view of reconstructed 3D tornados. The 2D tornado fitting process is performed within 0.3-4 seconds per frame for $N_s = 20$ on a Intel Pentimu IV 3.0 GHz machine with 1 GByte RAM. The animation is automatically reconstructed by using MEL script [6], and the rendering is done with

Maya 6.0 [1]. Six layers are used for the results. The reconstruction of a 90-frames animation takes about 130 seconds. The rendering takes less than 1 second per frame.

Once having the tornado feature data, we can create various tornado animations by manipulating them. For instance, a rigid transformation(rotate and translate) is applied to generate a path animation as shown in Figure 9. Another example is a spline-based control as illustrated in Figure 5. A spline is the line along the central axis of a tornado, which is shown as a green chain. Each joint of the spline defines the local transformation of the corresponding section. Thus, a new animation can be produced by controlling the spline. The left column represents an original spline configuration and its rendering. The right column is a controlled version of the data in the left column.

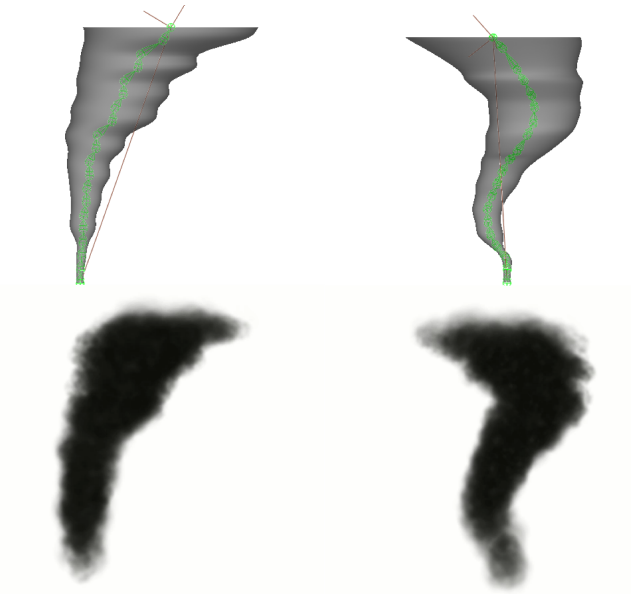

Figure 5. spline-based controls: (left) an original spline configuration and its rendering below, (right) a controlled version

One problem of our approach is the length of a reconstructed animation is limited by the length of an input video. We alleviate this limitation by using a video texture idea [20]. We arbitrarily extend the length of an animation by forming random loops in shape feature vector sequence instead of image sequence. An extended data sequence is used in the animation of Figure 9, in which the length of the data from Figure 6 is increased from 90 to 200 frames. We also generate the wind field of a reconstructed tornado to simulate interactions between the tornado and environments. The wind field is procedurally created by approximating a tornado vortex core as a discrete chain of vortex vectors. The rigid body simulation is performed in Maya [1]. All the result animations are provided as supplementary movies.

6 Conclusion and Future Work

We presented a novel method to create realistic 3D tornado animation from video clips captured by a single camera. The reconstruction process consists of three stages. First, we extract the shape of a simple 2D tornado model from silhouette images using a non-linear optimization algorithm and silhouette-based XOR cost function. Second, we generate a 3D tornado surface for the first frame from the extracted 2D shape of the tornado using NURBS surfacing. A tornado animation is created by keyframing every control point on the surface for the rest of frames. Finally, the tornado is rendered by associating a sprite image for each particle.

Our work is one of the first to reconstruct a tornado animation from a single video camera. We employ a simple 2D shape descriptor to extract and generate the tornado wind field. Our techniques are easy to implement, and the obtained 3D surface dataset is handy for animators to process further to add their own special effects. Currently, we assume constant intensity for the tornado region across a given images and relatively slow camera movements. In future, we plan to explore more advanced techniques in computer vision to segment the tornado region and therefore consider a wider range of tornado clips. It would be also useful to further explore video texture idea in a shape feature space in order to extend a tornado animation to an arbitrary length. Lastly, we want to employ a physics-based tornado field model such as Rankine or Burgers-Rott vortex [12] with tornado shape features to produce wind fields of a tornado.

References

[1] Alias. *Maya 6 Unlimited*, 2004.

[2] A. A. Apodaca and L. Gritz. *Advanced Renderman: Creating CGI for Motion Pictures*. Morgan Kaufmann, 2000.

[3] Y. Dobashi, T. Nishita, H. Yamashita, and T. Okita. Using metaballs to modeling and animate clouds from satellite images. *The Visual Computer*, 15(9):471–482, 1998.

[4] D. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, S. Worley, B. Mark, and J. Hart. *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann, 3 edition, 2002.

[5] S. Fangmeier. Designing digital tornados. *Computer Graphics World*, Aug 1996.

[6] D. Gould. *Complete Maya Programming: An Extensive Guide to MEL and C++ API*. Morgan Kaufmann, 2002.

[7] T. Grazulis. *The Tornado: Nature's Ultimate Windstrom*. University of Oklahoma Press, 2001.

[8] S. W. Hasinoff and K. N. Kutulakos. Photo-consistent 3d fire by flame-sheet decomposition. In *International Conference on Computer Vision*, pages 1184–1191, 2003.

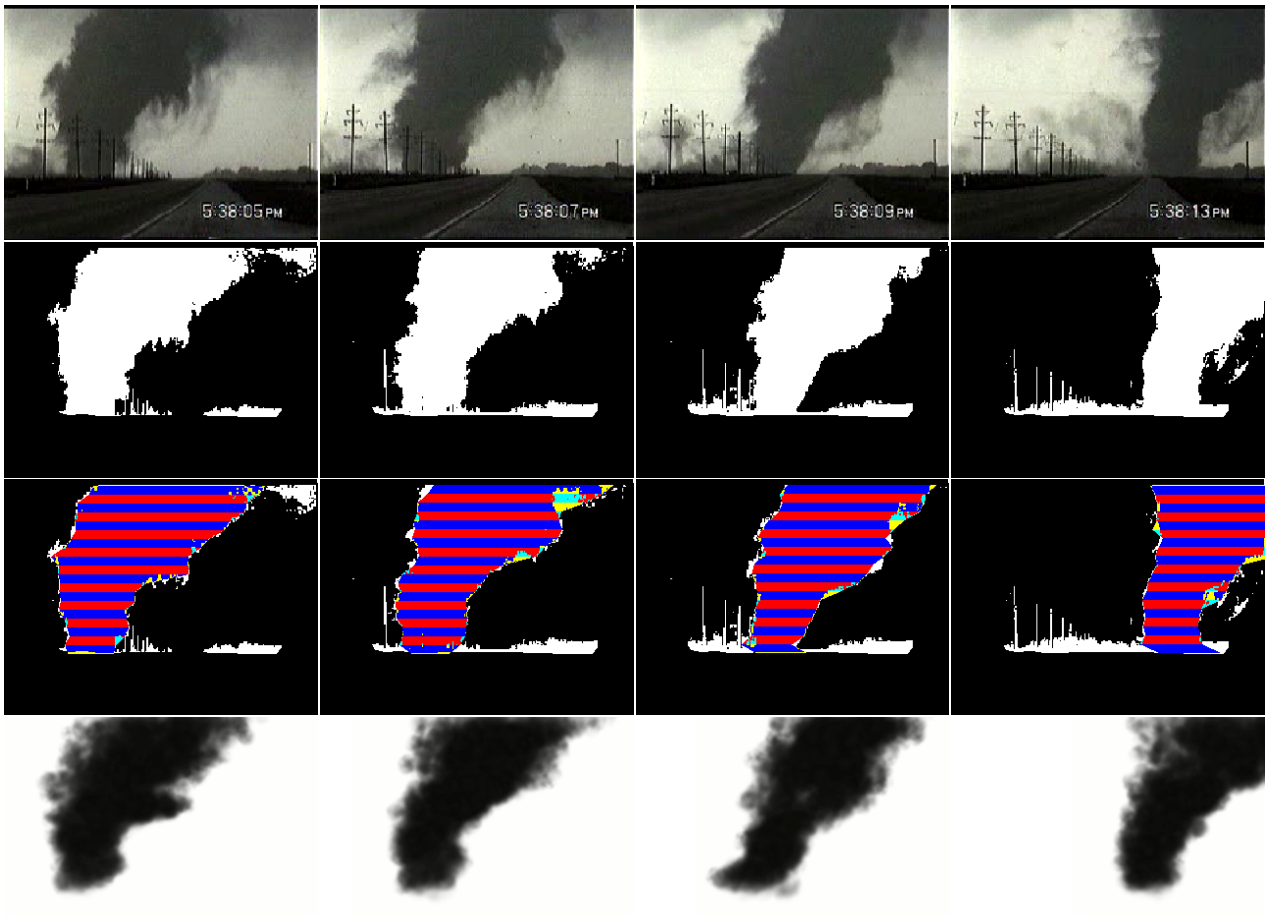

Figure 6. Resulting images for selected frames from a tornado video clip(1st row: input images, 2nd row: silhouette images, 3rd row: model fitting result, 4th row: frontal view rendering of 3D tornado

[9] Y. Iwakiri and T. Kaneko. PC-based realtime texture painting on real world objects. In *Proceedings of Eurographics 2001*, pages 104–113, 2001.

[10] B. F. Jewett. *Tornado Video*. http://redrock.ncsa.uiuc.edu/ jewett/.

[11] M. Jiang, R. Machiraju, and D. Thompson. Geometric verification of swriling features in flow fields. In *Proc. IEEE Visualization '02*, pages 307–314, 2002.

[12] K. T. Kilty. *Tornado Project*. http://www.kilty.com/tornado.htm.

[13] J. Lee, B. Moghaddam, H. Pfister, and R. Machiraju. Silhouette-based 3D face shape recovery. In *Proceedings of Graphics Interface*, pages 21–30, 2003.

[14] H. P. A. Lensch, W. Heidrich, and H. Seidel. Automated texture registration and stitching for real world models. In *Proceedings of Pacific Graphics 2000*, 2000.

[15] W. S. Lewellen. *The Tornado: Its Structure, Dynamics, Prediction, and Hazards*, chapter Tornado Vortex Theory, pages 19–39. American Geophysical Union, 1993.

[16] H. Lugt. *Vortex Flow in Nature and Technology*. John Wiley & Sons, 1983.

[17] B. Moghaddam, J. Lee, H. Pfister, and R. Machiraju. Model-based 3D face capture with shape-from-silhouettes. In *IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, pages 20–27, October 2003.

[18] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[19] W. H. Press, B. P. Flannery, S. A. Teukolosky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, 1988.

[20] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *Proc. SIGGRAPH '00*, pages 489–498, 2000.

[21] C. Sminchisescu. Consistency and coupling in human model likelihoods. In *IEEE International Conference on Automatic Face and Gesture Recognition*, May 2002.

[22] Y. Z. Wang and S. C. Zhu. A generative method for textured motion: Analysis and synthesis. In *Proceedings of European Conference on Computer Vision*, June 2002.

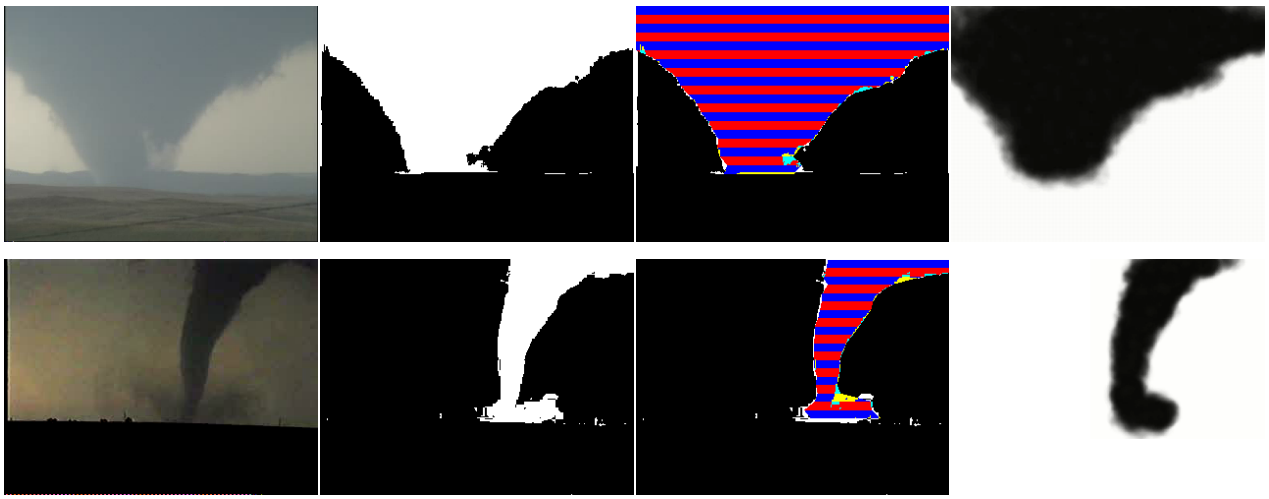

Figure 7. Resulting images for a frame for two tornado video clips

Figure 8. A bird eye view of 3D reconstructed tornados

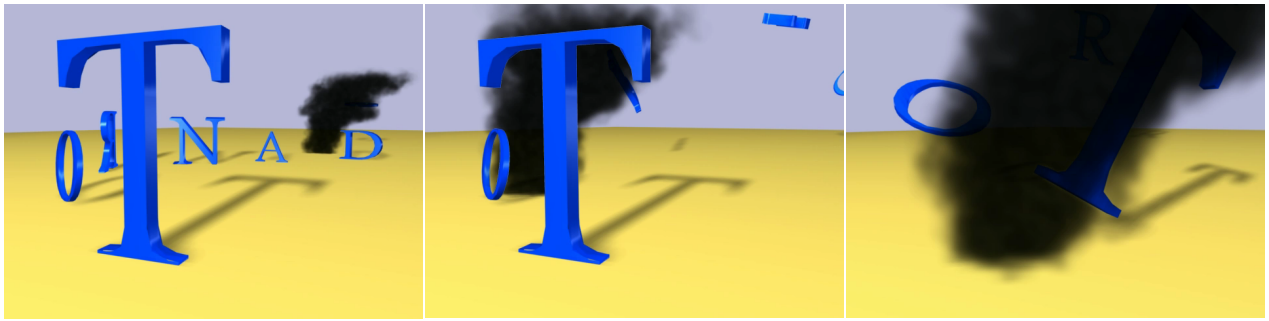

Figure 9. Tornado animation derived from the analysis of Figure 6