# VideoComposer - Motion-Object Capture With Active Contours to Compose Videos

GEORGIA ALBUQUERQUE TIAGO BARROS KASSIANA COSTA IVO NASCIMENTO
ALEJANDRO C. FRERY

UFPE–Universidade Federal de Pernambuco
CIn–Centro de Informática
Av. Professor Luis Freire, s/n
50740–540 Recife, PE, Brasil
{gpbca,tgfb,kmc,ifn,frery}@cin.ufpe.br

**Abstract.** Video editing is a very important step in midia production. There are many techniques that alleviate the complexity of video editing, being the selection, clipping and pasting of objects from a source one of them. Merging of a part of a video into another background is commonly used with chroma-key, bluescreen or green-screen compositing. When there is no prepared background available, i.e., when raw images or real world shots are used as input, frame-by-frame selection would be needed. Manual selection is too cumbersome to be efficiently performed, so computer-vision techniques are sought. The main goal of these techniques is identifying the object of interest (which is typically varying from frame to frame), and many proposals have been made in this direction. This paper presents a tool that, using an active contour model, is able to track multiple objects of interest in a sequence of video frames. The tool also allows performing the following steps of the editing process, copying the selected objects into an arbitrary background that can be static or another video. Users with little or no experience in video editing can use this tool to produce good quality results.

## 1 Introduction

Video and animation creation through the composition of elements extracted from different sources is an interesting approach, and particularly attractive to a fast growing group of untrained users, namely, people who want to make their own videos at home without having expertise in the field. Classical techniques rely on chroma key (bluescreen, green-screen etc.), but they require specialized studios, hardware and software. This paper presents a technique for object extraction based on computer vision models, implemented in a user-friendly Windows-based tool.

Extraction of objects of interest is a cumbersome task without the use of auxiliary techniques and tools for identification and capture. This recognition has to be performed on typically moving and shape-changing objects. In the field of computer vision, active contours or snakes are regarded as one of the most adequate techniques for this task. These were the models chosen to aid video editing.

The VideoComposer software here presented aims at helping users in the tedious tasks of composing their own videos using objects extracted from different sources.

This document initially presents a summary of previous works in this area (Section 2). Section 3 presents the initial steps to create a composed video, while Section 4 describes the tracking procedure. In the sequel, Section 5 comments the process of composing the final video sequence. Finally, sections 6 and 7 make comments about the results, conclusions and future directions for research.

## 2 Previous works

Many works have been developed in the area of active contours and motion tracking, being [1] the pioneering reference. This work presented a snake as an energy-minimization spline guided by external forces and influenced by internal forces. Dumitras and Venetsanopoulos [2] present a comprehensive comparison among different snake models. Lai and Chin [3] developed an API to work with snakes and motion tracking, called GSNAKE, implemented in `C++` for Unix. Another approach to motion tracking without active contours can be seen in the work by Cohen [4].

There are few published works dealing specifically with the application of active contours to animation, among them Hoch [5] uses snakes to track the facial feature of an actor, and the information is used to drive animated characters; Agarwala [6], presents an interactive system that allows children and others untrained users to create two-dimensional cartoons from video stream and images.

The following section describes in detail the process here proposed for video creation using several sources and snakes.

## 3 Creating one video frame

The process of creating a video composition from an existing image sequence starts with the selection of the target objects on the initial frame. We assume here that the objects consist of solid regions and are delimited by well-defined curves.

To select an object, the user draws a simple contour directly on the video's first image. This contour is a set of points joined by segments, where the starting and ending points are assumed to be the same in order to have a closed region.

A user-friendly system cannot require the user to provide an accurate or even approximate initial curve for the object of interest.

Such contours that do not lie directly on the high-frequency regions of the image data are hard to track, and then it is necessary to adjust the user's initial contour to allow image's objects clipping and tracking that is shown in Figure 3. We used snakes to fit the contour in the image characteristics, this technique is discussed in Section 3.1.

After selection of target objects, those are clipped and inserted in a new frame. It consists in a point-region problem, where is necessary to know which pixels are inside of the contour. A solution to this is discussed in Section 3.2.

## 3.1 Snakes

A Snake is a two-dimensional contour that is relaxed to a minimal energy configuration [1]. This energy is tailored in such a way that leads the contour to adapt around the desired or target object.

The curve has a initial position and a energy function defined as the weighted sum of an internal energy, responsible for geometrical features of the curve (smoothness, continuity etc.), and an external energy, which leads the contour to the salient characteristics of the image [7]. The energy can be then written as

$$e(V, f, \lambda) = \lambda E_{\text{int}}(V) + (1 - \lambda) E_{\text{ext}}(V, f), \tag{1}$$

where $f$ denotes the image, $V$ stands for the curve, $\lambda \in (0, 1)$ is the weight, and $E_{\text{int}}$ and $E_{\text{ext}}$ denote the internal and external energies, respectively. This form of energy function is a commonplace in image understanding (see [8]).

The internal energy is related to the shape of the curve and measures features like elasticity (how the curve can deform in consequence of a strength's action) and hardness (the curve's resistance to evolve into sharp spikes or corners). The external energy takes into account features of relevance in the image domain; from which a potential field is computed. The snake evolution seeks the minimization of the energy function presented in eq. (1). The equilibrium state or final configuration should be a contour that fits the object of interest.

The particular choice of the energies and of the weight defines the kind of snake to be used. Some approaches propose the use of a single $\lambda$ for the whole process, while others update this value. Some algorithms fix the energies, while others suggest the use of different energies for different parts of the curve.

Though skanes are continuous entities, they have to be described by some convenient finite model. Several splines-based descriptions are found in the literature, and the one employed in this work is a polyline with a varying number of (ordered) points, called GSNAKE (a `C++` for `Unix` API implementing this model is provided in [3]). Only closed snakes are considered here.

Following [3], varying energies will be used to enhance the performance of the algorithm.

$$E_{\text{int}}(V) = \sum_{i=0}^{n} \alpha(V_i) E_{\text{cont}}(V_i) + \sum_{i=0}^{n} \beta(V_i) E_{\text{curv}}(V_i) \tag{2}$$

What interest us about the GSNAKE was that its use of different internal energy function. The function differs from standard Kass [1], Witkind and Terzopoulous's. With simple image forces such as gradient map, gsnakes can detect corners or valleys in the image.

GSNAKE can efficiently detect corners or valleys because for each point on the snake, called it snaxel, it has different values for alpha and beta. GSNAKE determines alpha and beta adaptively from shape information. In other words, it guesses when to turn off the beta term (beta = 0) so that the snake will develop a corner.

GSNAKE converges to a solution more quickly than when standard internal energy is used. It requires fewer numbers of iterations.

## 3.2 Image clipping

Once the target contour has been found, it is necessary to decide which points belong to the interior of the region defined by the contour, a problem known as point-region problem. There are many ways to solve it, however they are often computationally intensive. The approach based on barycentric coordinates [9] was used for the sake of simplicity.

This method is based on the fact that it is easy to decide if a point belongs to the interior of a (proper) triangle. Then, the first step consists of partitioning the area of interest in triangles; this was done using Triangle Srip [10]. Once this is performed, each point can be tested to belong to the interior of each triangle. This test consists of solving a simple system of linear equations, as presented below for the two-dimensional case.

Consider three masses $w_1$, $w_2$ and $w_3$ placed at positions $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$ respectively, with $\mathbf{P}_i \in \mathbb{R}^2$ for every $1 \leq i \leq 3$. The center of gravity of these system of weights belongs to the interior of the triangle defined by these points, if and only if the masses are all positive. In this manner, given an arbitrary point $\mathbf{P} \in \mathbb{R}^2$ it will belong to the interior of the triangle defined by the points

$(\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3)$ if and only if there is a set of positive weights $(w_1, w_2, w_3)$ that makes it the center of gravity of the system of masses.

The system is then given by

$$\mathbf{P} = w_1\mathbf{P}_1 + w_2\mathbf{P}_2 + w_3\mathbf{P}_3.$$

In order to solve the system of equations, the additional assumption $w_1+w_2+w_3 = 1$ will be imposed on the masses. If, after solving the system of equations, some $w_i < 0$ then the point does not belong to the triangle.

Instead of applying this procedure to every pixel in the image, a coarse clipping can be performed in order to reduce the computational time. As previously said, the snakes considered in this work is a closed polygon formed by the set of points $((x_1, y_1), \ldots, (x_n, y_n))$. The search can be restricted to the points in the image satisfying $\{(x, y) : x_{1:n} \le x \le x_{n:n}, y_{1:n} \le y \le y_{n:n}\}$, where the subscripts $1 : n$ and $n : n$ denote the minimum and the maximum, respectively.

The barycentric procedure is then applied to every pixel in the restricted image, those that belong to the region defined by the polygon are sent to a buffer and, then, pasted over of the target background.

In principle, this procedure should be applied to every frame in the video how is shown in Figure 4. This greedy approach is too demanding, and diregards the fact that, hopefully, the contour of interest should not change too much from frame to frame. Next section describes an optimization technique that performs object tracking.

## 4 Tracking edges in image sequences

In order to optimize object tracking in a sequence of images, provided an appropriate contour is already detected in the first frame, the following assumptions are made about the object of interest:

- its mean brightness does not change;
- it suffers only small changes from one frame to the next.

In our implementation, the starting contour for the new object is the circunference with center the barycenter of the previous contour and radius the radius of the minimum circunference that inscribes the previous contour. This proposal aims at reducing the cost of searching in the whole image.

For tracking images in multiple frames, active contour models (snakes) have been successfully used in various applications, though they may be computationally intensive. However, in order to optimize this approach we used the Gradient-based Optical Flow technique to estimate the object motion. After this, the snake control points are pushed in the appropriated direction until detecting the relevant edges.

An optimization of the Optical Flow technique was used in our implementation: limiting the search region where the new control point will be placed by a radius. In order to perform this search a two-dimensional matrix was created around each previous control point. The matrix is compared, pixel by pixel, with a section of the same size in the next frame. This is illustrated in Figure 1. This procedure is repeated to each central pixel around the initial position up to a limited radius until the best mach is found and chosen as the future pixel position.

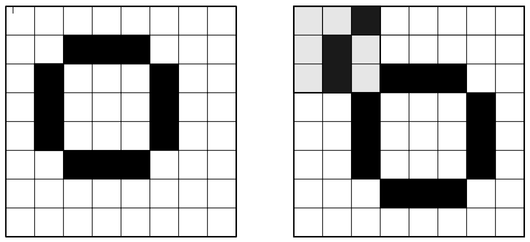

Figure 1: Tracking

The final new contour is obtained applying the procedure described in Section 3.1 to this starting configuration. This is repeated for every frame in the sequence.

## 5 Composing video

After extracting the object(s) from the source video, the composite output will be produced merging the chosen element(s) from each source into a single video sequence. For the sake of simplicity the procedure for only one object will be described.

The first step consists of defining the output background, which can be done in any of the following two manners:

- Solid color: the video background can be a single color picked up from a color choose dialog. The color will fill each frame of the video before pasting the object on it.
- Bitmap image: a user-defined bitmap image will be copied in each frame of the output video.

Once the background is defined, the desired object(s) are be pasted in the output video. The object of interest is chosen by mere clicking on the first image of the sequence. It will then be followed in the rest of the frames, extracted and pasted onto the selected background frame-by-frame in the position at which it was detected in the original video.

After inserting the first object, the resulting video will have the same number of frames as the sequence from which the object was imported. Other objects can then be inserted in this new video, or even the same object at different stages of evolution. If necessary, more frames will be added to video automatically, to complete the animation of the object that will be inserted.

The object is placed in the output video copying every detected pixel, so each new object inserted will replace the old pixel values, replacing any portion of object that was in the same position. In this manner, no layers are supported in the current version of the platform in order to generate simple video output.

## 6 Results

A composed video created by the VideoComposer is shown in Figure 6, Figure 7 and Figure 8. Those figures shows several frames of the source video, the extracted objects and the composite output video respectively. The Figure 5 shows the three steps: the initial frame, the selected and clipped frame and the final frame in the application window.

The quality of the extracted objects will depend on the input video quality. Excessive noise affects the behavior of the snake, making it hard to fit correctly the boundaries of the target object. The quality of the resulting video is satisfactory, mainly if we consider that a manual work like that, made by video editing professionals, would consume much more time that the time spent with our tool.

However, the quality looses are balanced with time gain, and this is a great advantage to untrained users, that are not, and do not need, to be professionals.

## 7 Conclusion and Future Work

This work has discussed an important use of active contours and motion tracking in the field of animation: an application that employs snakes and object following to help video composition. The tool showed in Figure 2 is stable and admints standard video input and generates standard video output: the AVI format.

These videos must have the following characteristics: 24 bits of pixeldepth, without audio, without compression and the rate of presentation equal to one frame per second. These characteristics had been defined to facilitate the extraction of the frames.

The tool can be further developed allowing the use of other formats, such as streaming video. Another feasible improvement is the support of automatic animation and face recognition.

## References

[1] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," *International Journal of Computer Vision*, pp. 321–331, 1987.

[2] A. Dumitras and A. N. Venetsanopoulos, "A comparative study of snake models with application to object shape description in bi-level and gray-level images," in *Proceedings of IEEE-Eurasip Workshop on Nonlinear Signal and Image Processing*, (Boston, USA), 2001.

[3] K. F. Lai, S. Chan, C. W. Ngo, E. L. Ang, and O. K. W., *GSNAKE API*. PhD thesis, Information Technology Institute, Singapore, 1995.

[4] I. Cohen and G. Medioni, "Detecting and tracking moving objects in video surveillance," in *Proceedings of IEEE Computer Vision and Pattern Recognition*, (Fort Collins, Colorado), 1999.

[5] M. Hoch and P. C. Litwinowicz, "A pratical solution for tracking edges in image sequences with snakes," *The Visual Computer*, vol. 12, no. 2, pp. 75–83, 1996.

[6] A. Agarwala, "Snaketoonz: a semi-automatic approach to creating 2d cartoons from video and image," in *NPAR 2002: Second International Symposium on Non-Photorealistic Animation and Rendering*, 2002.

[7] A. Blake and M. Isard, *Active Contours*. London: Springer-Verlag, 1998.

[8] D. Geman and S. Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, Nov. 1984.

[9] K. Join, "On-line computer graphics notes." `http://muldoon.cipic.ucdavis.edu/GraphicsNotes/Barycentric-Coordinates/Barycentric-Coordinates.html`, 1996.

[10] F. Evans, S. Skiena, and A. Varshney, "A. optimizing triangle strips for fast rendering. New York," *IEEE*, pp. 319–326, 1996.

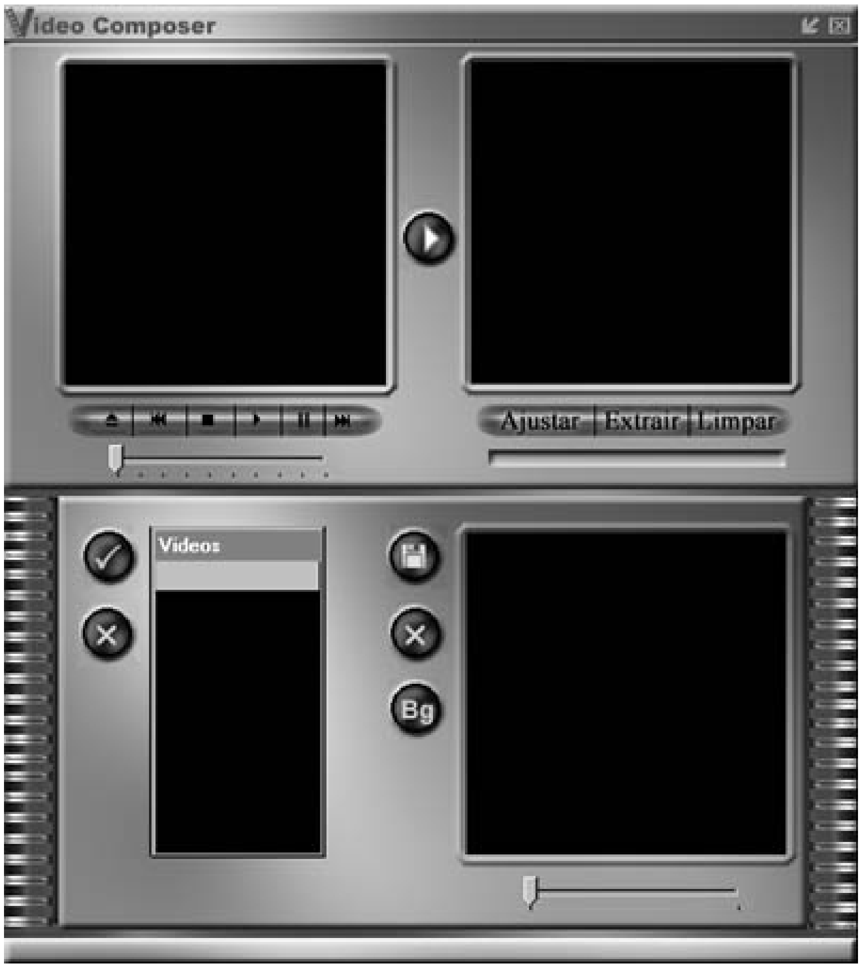

Figure 2: Application

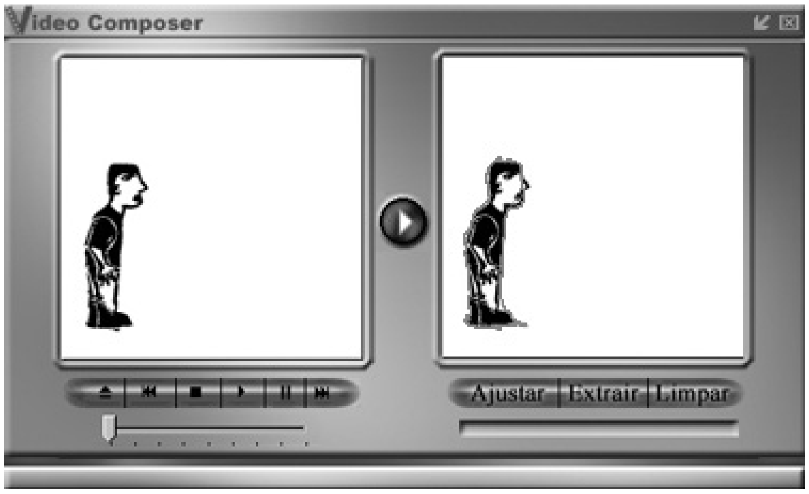

Figure 3: Minimizing the snake

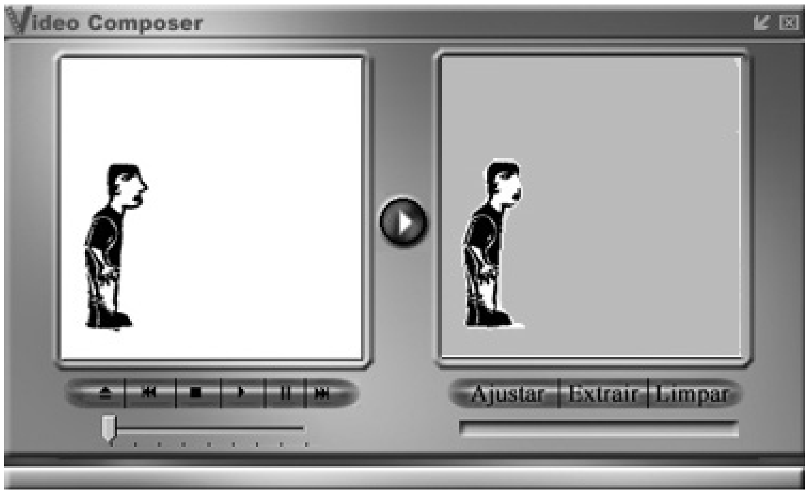

Figure 4: Clipping

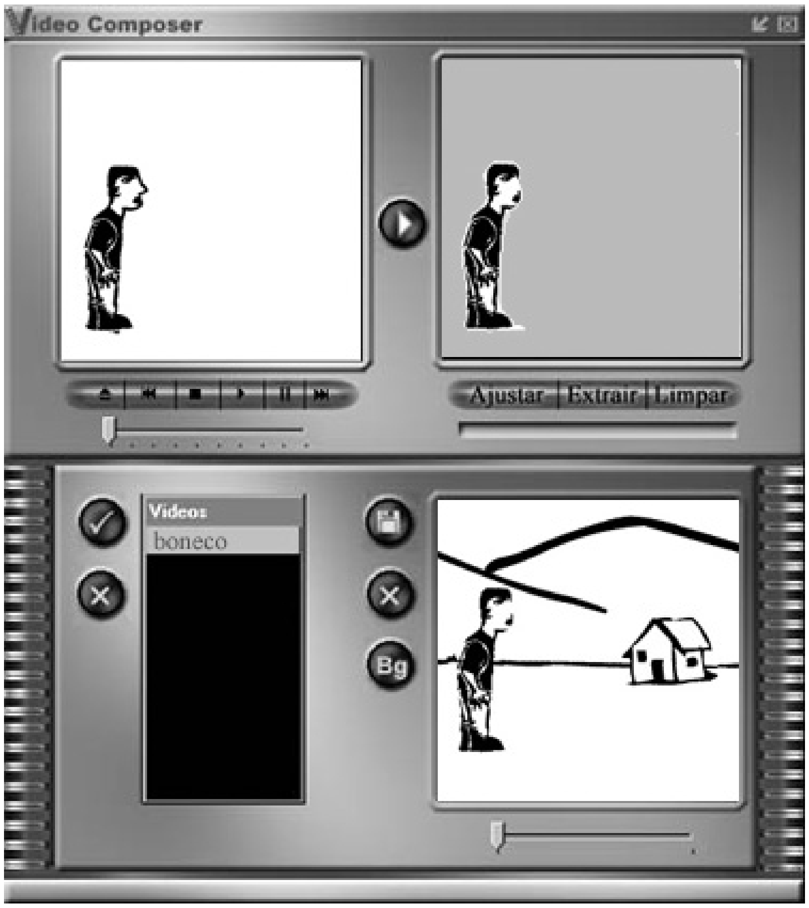

Figure 5: Final result

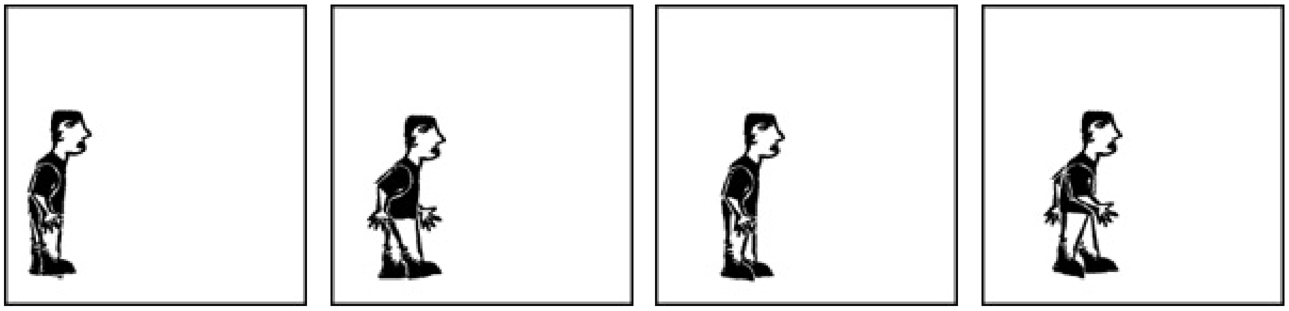

Figure 6: Initial video

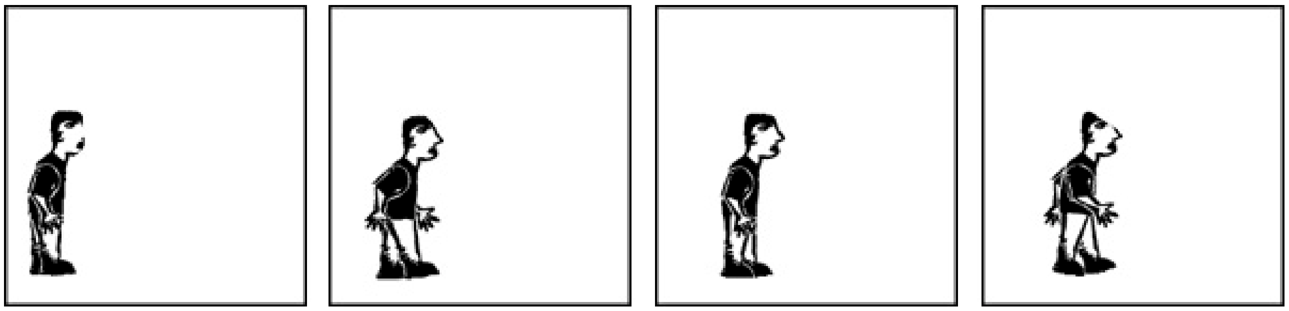

Figure 7: Object clipping

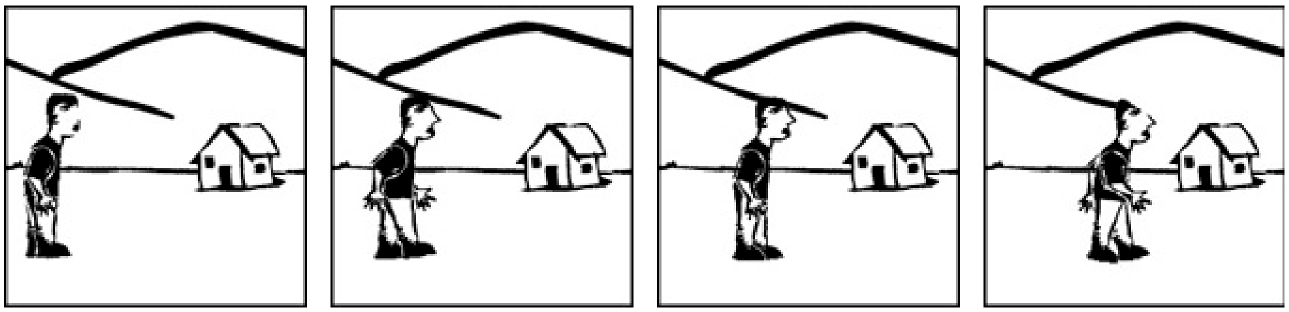

Figure 8: Result video