Computing the Physical Parameters of Rigid-Body Motion from Video

Kiran S. Bhat[1], Steven M. Seitz[2], Jovan Popović[3], and Pradeep K. Khosla[1]

[1] Carnegie Mellon University, Pittsburgh, PA, USA
{kiranb,pkk}@cs.cmu.edu
[2] University of Washington, Seattle, WA, USA
seitz@cs.washington.edu
[3] Massachusetts Institute of Technology, Cambridge, MA, USA
jovan@lcs.mit.edu

Abstract. This paper presents an optimization framework for estimating the motion and underlying physical parameters of a rigid body in free flight from video. The algorithm takes a video clip of a tumbling rigid body of known shape and generates a physical simulation of the object observed in the video clip. This solution is found by optimizing the simulation parameters to best match the motion observed in the video sequence. These simulation parameters include initial positions and velocities, environment parameters like gravity direction and parameters of the camera. A global objective function computes the sum squared difference between the silhouette of the object in simulation and the silhouette obtained from video at each frame. Applications include creating interesting rigid body animations, tracking complex rigid body motions in video and estimating camera parameters from video.

1 Introduction

The motion of real objects is governed by their underlying physical properties and their interactions with the environment. For example, a coin tossed in the air undergoes a complex motion that depends on its initial position, orientation, and velocity at the time at which it is thrown. Replacing the coin with a bowling pin produces a distinctly different motion, indicating that motion is also influenced by shape, mass distribution, and other intrinsic properties of the object. Finally, the environment also affects the motion of an object, through the effects of gravity, air drag, and collisions.

In this paper we present a framework for recovering physical parameters of objects and environments from video data. We focus specifically on the case of computing the parameters underlying the motion of a tumbling rigid object in free flight assuming the object shape and mass distribution are known. Our algorithm models tumbling dynamics using ordinary differential equations, whose parameters include gravity, inertia and initial velocities. We present an optimization framework to identify these physical parameters from video.

Our dynamic model captures the true rotational physics of a tumbling rigid body. This aspect distinguishes our work from prior work in motion tracking and analysis [4,7,11,5], where the focus is on identifying object kinematics, i.e., motion trajectories. Moreover,

A. Heyden et al. (Eds.): ECCV 2002, LNCS 2350, pp. 551–565, 2002.

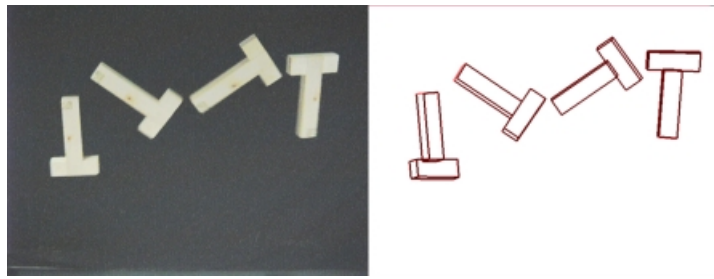

Fig. 1. Four frames of a tumbling object superimposed (left) and a physical simulation generated from the estimated motion parameters. The object is thrown from right to left. Our optimization framework computes the object, camera and environment parameters to match the simulated wireframe object with the video at every frame.

our algorithm uses information from all frames in the video sequence simultaneously, unlike feedforward filter based methods. Figure 2 compares the results of our offline batch algorithm with an online Kalman filter applied to a synthetic example of 2D ballistic motion with gaussian noise. Our algorithm tries to find the best fit parabola to the data, whereas a Kalman filter fits a higher order curve to the data. Although the Kalman filter approach *tracks* the data better, our algorithm finds the true parameters describing the physics of the ballistic motion. These parameters can now be used to *animate* a particle in simulation, that *moves like* the given data. However, most tracking tasks require only kinematic properties and therefore, our approach of accurately modeling the underlying physics might seem to be unnecessary or prohibitively difficult.

We argue that estimating physical parameters has important benefits for the analysis of rigid-body motion. First, the use of an accurate physical model actually *simplifies* the task of recovering kinematics, since the complete motion is determined by the initial state and a small number of other parameters. Second, the recovered model enables the behavior of the object to be *predicted* in new or unseen conditions. For instance, from a short video clip of a object at any point in its trajectory, we can reason from where it was launched and where and in what attitude it will land. This same ability allows the path to be followed *through occlusions*. In addition, we can predict how the object would behave in different conditions, i.e., with more angular velocity. In the same manner, by recovering parameters of the environment, we can predict how different objects would move in that environment. As one application of measuring the environment, we show how estimating the direction of gravity in an image sequence can be used to rectify a video to correct for camera roll.

We estimate parameters of a rigid-body motion with an optimization that seeks to match the resulting motion with the frames of the video sequence. Rather than operating in a feed-forward manner, as is typical of object tracking techniques [16,17, 7,4], we cast the problem in a global optimization framework that optimizes over all frames at once. Using this framework, we show how it is possible to simultaneously compute the object, camera, and environment parameters from video data. Unlike

previous analytical methods [15,14], our method does not require any velocity, acceleration, or torque measurements to compute the body state as a function of time. Furthermore, an important element of our estimation approach is that it relies only on easily computable metrics such as image silhouettes and 2D bounding boxes, avoiding the need to compute optical flow or track features on the object over time. Our optimizer employs general-purpose rigid-body simulators [1] that model a wide range of behaviors, such as collisions and articulated structures.

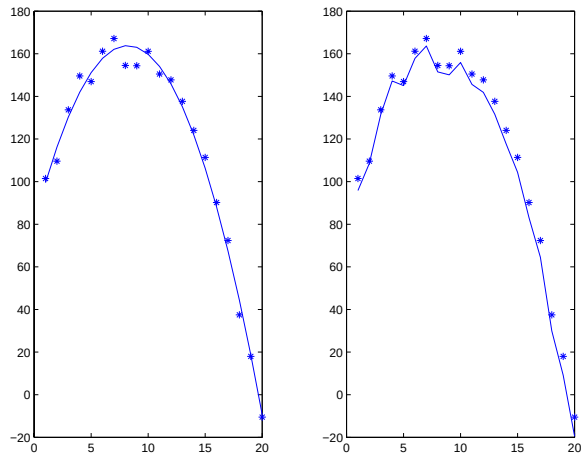

Fig. 2. Illustrating the differences between parameter estimation using optimization (left) and a Kalman filter (right) on a simplistic 2D example. The input data is the trajectory of a point mass in free flight, corrupted with gaussian noise. Our optimization algorithm uses all the data points simultaneously to fit a path that globally satisfies physical laws; in this simple case, a parabolic path. The Kalman filter processes the data sequentially and fits a higher order curve to the data.

2 Related Work

Several researchers in the computer vision community have focussed on the problem of extracting body shape and camera motion from video [23,12,8]. There is very little work on extracting the underlying physical properties of the object or the environment. Several groups [20,3] have modeled the dynamics of point masses (free flight and collisions) from video to design controllers for robots performing juggling or playing air hockey. Ghosh et al. [9] use estimation theory to recover the riccati dynamics and shape of planar objects using optical flow. Masutani et al. [15] describes an algorithm to extract the inertial parameters of a tumbling rigid body from video. Their system tracks feature points in the images to get instantaneous velocity measurements and uses the Poinsot's solution [22] to compute the inertial parameters. The problem of simultaneously recovering the physical parameters of the object, camera, and environment from a single camera has not been previously addressed.

Our work is closely related to prior work on model based tracking in computer vision [11,5,21,4,7,24,17,16]. However, the notion of a dynamic model in the tracking literature is different from the one presented here. We use ordinary differential equations to model the non-linear rotational dynamics of tumbling rigid bodies, and extract its parameters from video. These parameters include initial velocities, gravity and inertias. In contrast, most of the prior tracking algorithms use a Kalman filter to update the state variables of the moving object. In many instances, the dynamic model that relates the current and previous states is extremely simple [11,5]. However, they are sufficient for tracking rigid-body motions [11] or for navigation [5,21]. Kalman filters have also been successfully applied to track articulated [24] and non-rigid motion [16,17] in video. The filter based techniques are feed-forward in nature and are ideally suited for tracking applications. Our technique works offline using information from all frames simultaneously to estimate the physical parameters.

Several techniques have been developed [19,6] to estimate the parameters of a simulation to design computer animations of rigid bodies. Although these methods have been used for synthesis, to date, they have not been used to analyze motion from video. Our technique is similar to recent parameter estimation techniques for designing rigid body animations in computer graphics [19,18].

3 Problem Statement

Our goal is to infer the physical parameters underlying the motion of a rigid body in free flight from a pre-recorded video sequence of its motion. Free flight motion of the rigid body in video is determined by a relatively small set of parameters. They can be categorized into three groups:

- **Object Parameters**: We distinguish two types of object parameters: intrinsic parameters (object shape, mass distribution and location of center of mass), extrinsic parameters (initial position, orientation, velocity, angular velocity).
- **Environment Parameters**: Parameters such as gravity direction and air drag.
- **Camera**: We distinguish two types of camera parameters: intrinsic (focal length, principle point, etc.), extrinsic parameters (position and orientation).

In this paper, we extract the extrinsic parameters of the object and the direction of the gravity vector. We assume that the shape and inertial properties of the object are known and the effect of air drag is negligible. Without loss of generality, we place the origin of the world coordinate system at the camera center, with the principle axes aligned with the camera coordinate system. Due to the choice of this coordinate system, the direction of gravity depends on the orientation of the camera, and is not necessarily vertical.

We employ the standard mathematical model from classical mechanics to represent the motion of a rigid body in free flight. A set of nonlinear ordinary differential equations (ODE) model the motion of a rigid body by computing the evolution of its state, which includes the body's position and velocities, over time. As a result, the

motion of a rigid body is succinctly described by the parameters that affect the solution of the ODE. We use the term *simulation* to refer to the process of computing the body's motion by integrating the ODE.

4 Estimation from Video

In this section, we describe the equations of motion of a rigid body in free flight, and identify the parameters that affect the motion. We present an optimization framework to identify these parameters from a video sequence.

4.1 Equations of Motion

The tumbling motion of a rigid body in free flight is characterised by its position, orientation, linear and angular velocity. Let us define the state $\boldsymbol{q}(t)= [\boldsymbol{X}(t),\boldsymbol{\theta}(t),\boldsymbol{V}(t),\boldsymbol{\omega}(t)]$ to be the values of these parameters at time t. $\boldsymbol{X}(t) \in \mathbf{R}^3$ and $\boldsymbol{\theta}(t) \in SO(3)$ specify the position and orientation in a world coordinate system. We use a quaternion representation to represent the orientation with four parameters. $\boldsymbol{V}(t) \in \mathbf{R}^3$ and $\boldsymbol{\omega}(t) \in \mathbf{R}^3$ specify the linear and angular velocity coordinates. The time derivative of the state $\boldsymbol{q}(t)$, called the simulation function $\boldsymbol{F}(t,\boldsymbol{q}(t))$, is governed by the ODE

$$\boldsymbol{F}(t, \boldsymbol{q}(t)) = \frac{d}{dt}\left(\boldsymbol{q}(t)\right) = \frac{d}{dt}\begin{pmatrix} \boldsymbol{X}(t) \\ \boldsymbol{\theta}(t) \\ \boldsymbol{V}(t) \\ \boldsymbol{\omega}(t) \end{pmatrix} = \begin{bmatrix} \boldsymbol{V}(t) \\ \frac{1}{2}(\boldsymbol{\theta}(t) * \boldsymbol{\omega}(t)) \\ \boldsymbol{g} \\ -\boldsymbol{I}(t)^{-1}\left(\frac{d\boldsymbol{I}(t)}{dt}\boldsymbol{\omega}(t)\right) \end{bmatrix} \tag{4.1}$$

Here, $\boldsymbol{I}(t)$ is the inertia matrix in world coordinates, and $\boldsymbol{g} \approx (0, -9.81, 0)$ is the acceleration due to gravity. The product $*$ refers to the quaternion multiplication. The state $\boldsymbol{q}(t)$ at any time instant $t = t_f$ is determined by integrating Eq (4.1):

$$\boldsymbol{q}(t_f) = \boldsymbol{q}(t_0) + \int_{t_0}^{t_f} \boldsymbol{F}(t, \boldsymbol{q}(t))dt \tag{4.2}$$

The state at any time $\boldsymbol{q}(t)$ depends on the initial state $\boldsymbol{q}(t_0)$ and inertial matrix. In this paper, we assume that the inertia matrix is known. Consequently, it is sufficient to solve for the initial state, which we denote by $\mathbf{p_{obj}}$.

4.2 Estimating Parameters from 3D Data

The state of the body $\boldsymbol{q}(t)$ describes the configuration of the body at any time. From the state $\boldsymbol{q}(t_i)$ at any time t_i, we can compute the state $\boldsymbol{q}(t_j)$ at any other time t_j by running the simulation forwards or backwards. However, obtaining the full state information from the real world requires linear and angular velocity measurements, which are hard to measure accurately.

4.3 Estimating Parameters from Video

This section describes techniques to extract the simulation parameters $\mathbf{p} = (\mathbf{p_{obj}}, \mathbf{p_{env}})$ from video. In this paper, we recover the object parameters $\mathbf{p_{obj}}$ and gravity direction $\mathbf{p_{env}}$. The gravity direction in our framework is encoded by two angles (tilt and roll). Video provides strong cues about the instantaneous pose and velocities of an object. However, the complex motion of a tumbling rigid body limits the information that can be reliably extracted from video. In particular, it is difficult to track a point on a tumbling object over many frames because of self occlusion. The high speeds of typical tumbling motions induces significant motion blur making measurements like optical flow very noisy. In contrast, it is easier to measure the bounding box or silhouettes of a tumbling body.

We solve for simulation parameters $\mathbf{p}$ by minimizing the least square error between the silhouettes from video and silhouettes from the simulation at each frame. The details of this optimization are given in Section 5. The object parameters in our formulation include both initial position and velocities. Alternatively, we can reduce the search space by first recovering the 3D pose (position and orientation) from the sequence using prior vision techniques, and then optimizing for the velocities that generate the set of 3D poses. Several researchers in the recognition community describe algorithms to recover the 3D pose of an object from silhouettes [13,10]. Let $(\boldsymbol{X}(t_0), \boldsymbol{\theta}(t_0)), ..., (\boldsymbol{X}(t_k), \boldsymbol{\theta}(t_k))$ be the sequence of poses computed from a sequence of k frames. The optimization algorithm finds *a feasible* solution for the initial linear and angular velocity by minimizing the following objective function:

$$\min_{\boldsymbol{\omega}(t_0), \boldsymbol{v}(t_0)} \sum_{i=t_0, t_1 \ldots t_k} \left(\boldsymbol{X}(i) - \boldsymbol{X}^s(i)\right)^2 + \left(\boldsymbol{\theta}(i) - \boldsymbol{\theta}^s(i)\right)^2 \tag{4.3}$$

where $\boldsymbol{X}^s(t_j)$ is the position and $\boldsymbol{\theta}^s(t_j)$ is the orientation of the object (in simulation) at time $t = t_j$ for the current estimate of inital velocities $\boldsymbol{v}(t_0)$ and $\boldsymbol{\omega}(t_0)$. Section 5 provides details on computing the analytical derivatives required for this minimization.

Although this method reduces the number of variables to be optimized, its performance depends on the accuracy of the pose estimates obtained from silhouettes. Since recognition-based methods do not enforce dynamics (they operate on a per frame basis), they might introduce discontinuities in pose measurements, leading to incorrect solutions for initial velocities. Hence, we decided to optimize for the pose and velocity and gravity parameters simultaneously in our framework. However, the error space with our formulation has several local minima, and hence our technique is sensitive to initialization. Moreover, fast tumbling motion could result in aliasing problems, especially with slow cameras. Hence, the solution obtained from optimization is not *unique*. The details of this optimization are given in Section 5.

5 Optimization

This section describes the details of the unconstrained optimization employed to estimate physical parameters from video. The algorithm solves for the object, camera,

and environment parameters simultaneously which generate a simulation that *best matches* the video clip. We use shape-based metrics to compare the real and simulated motions. The resulting objective function computes the sum squared differences of the metric over all the frames in the sequence. Our algorithm works offline and analyzes all frames in the video sequence to compute the parameters. We first preprocess the video to obtain a background model of the scene. Then, we segment the moving object and compute its bounding box and silhouette at each frame. The bounding box **B** is stored a vector of four numbers, which are the positions of its extreme corners. The silhouette **S** is represented by a binary image enclosed within the bounding box.

Recall that the motion of a rigid body is fully determined by the parameters **p**. For a given set of parameters, the optimizer simulates the motion to compute the bounding boxe and silhouette at each frame. It also computes the gradients of the metrics, which is used to update the parameters in the next optimization step. The goal of the optimization algorithm is to minimize the deviation between the silhouettes of the simulated motion and the silhouettes detected in the video sequence. We provide an initial estimate for the parameters **p** and use a gradient descent to update it at each step of the optimization. Gradient based methods are fast, and with reasonable initialization, converge quickly to the correct local minima.

5.1 Objective Function

The objective function measures the difference between the motion observed in video and the motion computed in the simulation. For example, at every frame, our implementation performs a pixel by pixel comparison between the silhouette from simulation and silhouette from video and computes a sum squared difference. This amounts to counting the non-overlapping pixels between the two silhouettes at each frame. This difference is accumulated over all the frames in the sequence. Alternatively, we could compare the second moments of the silhouette at each frame, and compute a SSD of the moments over all frames. The objective function for the silhouette metric has the form:

$$E = \min_{\mathbf{p}} \sum_{i=t_1,t_2\ldots t_f} \left(\mathbf{S}^v(i) - \mathbf{S}^s(i,\mathbf{p})\right)^2 \tag{5.1}$$

where $\mathbf{S}^v(i)$ and $\mathbf{S}^s(i)$ is the silhouette obtained at time i from the video and simulation respectively. Gradient descent is used to minimize this error function. The update rule for parameters is:

$$\mathbf{p} = \mathbf{p} + \lambda \frac{\partial E}{\partial \mathbf{p}} \tag{5.2}$$

where λ is the magnitude of the step in the gradient direction. The following subsections describe the gradient computation in detail.

5.2 Gradient Computation

The optimization algorithm requires computing the gradients of the objective function. This in turn, requires computing the gradients of the silhouette at each frame. Although

the state $\mathbf{q}(t)$ is a continuous function of the parameters $\mathbf{p}$ (Eq. 4.1), quantities like bounding boxes and silhouettes are not continuously differentiable functions of parameters $\mathbf{p}$. One straightforward approach is to compute the gradients of the metrics (e.g. $\partial \mathbf{S}(\mathbf{q})/\partial \mathbf{p}$) numerically, using finite differences. This, however, has two major drawbacks. First, computing the gradients of the metric with respect to $\mathbf{p_{obj}}$ (initial conditions) using finite differences is extremely slow, since the simulation function has to be evaluated several times during the gradient computation. Secondly, determining robust step sizes that yield an accurate finite difference approximation is difficult. We resort to a hybrid approach for computing the gradients. First, we analytically compute the gradients of the state with respect to parameters $\partial \mathbf{q}(\mathbf{t})/\partial \mathbf{p}$. We then compute the derivative of the metric with respect to the state using finite differences, e.g. $\partial \mathbf{S}(\mathbf{q})/\partial \mathbf{q}$. We use the chain rule to combine the two gradients:

$$\frac{\partial \mathbf{S}}{\partial \mathbf{p}} = \frac{\partial \mathbf{S}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \mathbf{p}} \tag{5.3}$$

Since the metric (e.g. silhouette $\mathbf{S}$) depends only on the position and orientation terms of the state $\mathbf{q}$, the gradient $\partial \mathbf{S}(\mathbf{q})/\partial \mathbf{q}$ can be computed quickly and accurately using finite differences. Finally, we note that the camera parameters do not depend on the 3D state of the object. Therefore, we use finite differences to compute the gradients with respect to gravity vector $\mathbf{p_{env}}$.

Jacobian for Free Flight. The motion of a rigid body in free flight (in 3D) is fully determined by the control vector $\mathbf{p_{obj}}$. Rewriting Eq. (4.1) to show this dependence explicitly yields:

$$\frac{d\mathbf{q}(t)}{dt} = \mathbf{F}(t, \mathbf{p_{obj}}) \tag{5.4}$$

We evaluate the jacobian $\partial \mathbf{q}(t_f)/\partial \mathbf{p_{obj}}$ at time t_f by numerically integrating the equation

$$\frac{d}{dt}\left(\frac{\partial \mathbf{q}(t)}{\partial \mathbf{p_{obj}}}\right) = \frac{\partial \mathbf{F}(t, \mathbf{p_{obj}})}{\partial \mathbf{p_{obj}}} \tag{5.5}$$

until time t_f with the initial condition $\partial \mathbf{q}(t_0)/\partial \mathbf{p_{obj}}$. We use a fourth order Runge-Kutta method with fixed step size to perform this numerical integration.

Derivatives of Silhouettes. We use finite differences to compute the derivative of silhouette with respect to the current state $\mathbf{q}(\mathbf{t})$. We compute the silhouette at the given state by rendering the simulated object. The derivative of the silhouette with respect to a scalar component q_i of the state $\mathbf{q}$ has the form:

$$\frac{\partial \mathbf{S}}{\partial q_i} = \lim_{\Delta q_i \to 0} \left(\frac{\mathbf{S}(\mathbf{q} + \Delta q_i) - \mathbf{S}(\mathbf{q})}{\Delta q_i} \right) \tag{5.6}$$

The jacobian of the silhouette is obtained by applying chain rule

$$\frac{\partial \mathbf{S}}{\partial \mathbf{p_{obj}}} = \frac{\partial \mathbf{S}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \mathbf{p_{obj}}} \tag{5.7}$$

Derivatives with Respect to Gravity direction. We compute the jacobian of the silhouette with respect to a scalar component p^i_{env} of the gravity direction $\mathbf{p_{env}}$ using finite differences, as shown:

$$\frac{\partial \mathbf{S}}{\partial p^i_{env}} = \lim_{\Delta p^i_{env} \to 0} \left(\frac{\mathbf{S}(\mathbf{p_{env}} + \Delta p^i_{env}) - \mathbf{S}(\mathbf{p_{env}})}{\Delta p^i_{env}} \right) \tag{5.8}$$

The overall jacobian matrix is given by:

$$\frac{\partial \mathbf{S}}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial \mathbf{S}}{\partial \mathbf{p_{obj}}} \\ \frac{\partial \mathbf{S}}{\partial \mathbf{p_{env}}} \end{pmatrix} \tag{5.9}$$

6 Results

Our system has three main modules: *Preprocessor*, *Rigid body simulator* and *Optimizer*. We first preprocess the video sequence to compute the silhouettes and bounding boxes of the rigid object. We build a background model for the scene and use autoregressive filters [2] to segment the moving object from the background. We then compute the bounding box and silhouette metrics from the segmented image at each frame. Our tumbling video sequences are typically 35-40 frames long when captured with a digital camera operating at 30 Hz. The optimizer typically takes a couple of minutes to compute the parameters from the sequence on a SGI R12000 processor.

Experiment 1: The goal of this example is to match the motion of a simulation with a complex tumbling motion of a T shaped object (Figure 3). Our user interface lets the user specify an approximate value for the initial positions and velocities of the body. The algorithm robustly estimates the initial position and linear velocity, even with a poor initial guess. However, it is very sensitive to the initial estimate of the orientation and angular velocities. From numerous experiments, we have found the error space of the silhouette metric to be very noisy, containing many local minima. However, with a reasonable initialization for the orientation and angular velocity parameters, we find that our algorithm converges to a reasonable local minima. This convergence is seen in Figure 3(a), where the overall motion of the simulation closely matches the motion in video. We superimpose the bounding boxes obtained from simulation onto the frames from video (the white boxes in the first row) to show the match. We also show the match between the trajectory of a corner point in video with the corresponding trajectory in simulation. The small square boxes show the trajectory of a corner point, identified by hand, from the video sequence. These trajectories are shown for visualization purposes only and are not used in the optimization. As the algorithm proceeds, the trajectory of the 3D corner point in simulation (black line) overlaps with these boxes. This sequence also highlights some limitations with our optimization framework and metrics. Row (c) shows an example where the simulated and the real object have totally different orientations but have silhouettes that look very similar. Finally, we note that our algorithm generates a 3D reconstruction of the object's trajectory that matches the given video sequence.

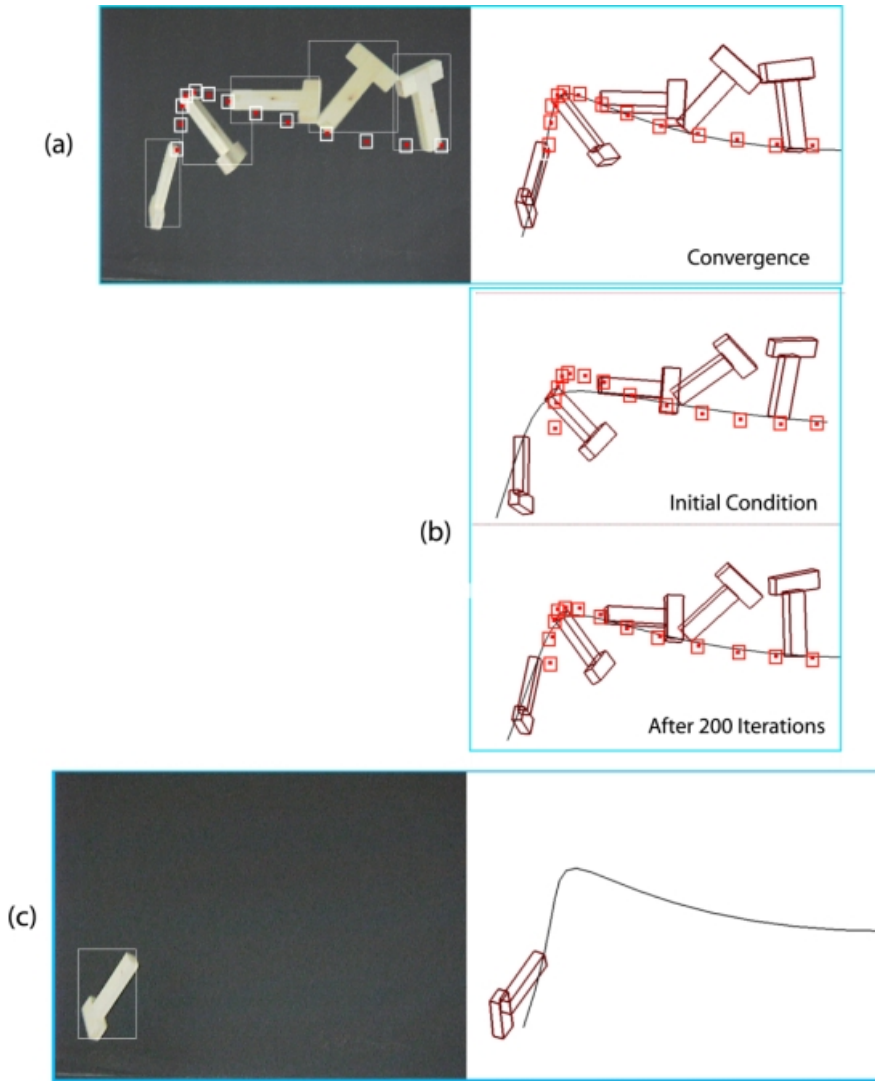

Fig. 3. Computing the parameters of a T-shaped object thrown in the air. The first row (a) shows a few frames from the video (right to left) and the corresponding physical simulation generated by our algorithm. The bounding boxes from simulation are superimposed over the video to show the match. The small square boxes indicate the trajectory of a corner point in video. The thin line indicates the motion of the corresponding corner in simulation. Row (b) shows the results of the algorithm at different stages of optimization. Notice that the match improves as the number of iterations increases. Row (c) highlights a limitation of the silhouette based metric. Note that although the orientation of the simulated object is flipped relative to the real object, they both have similar silhouettes

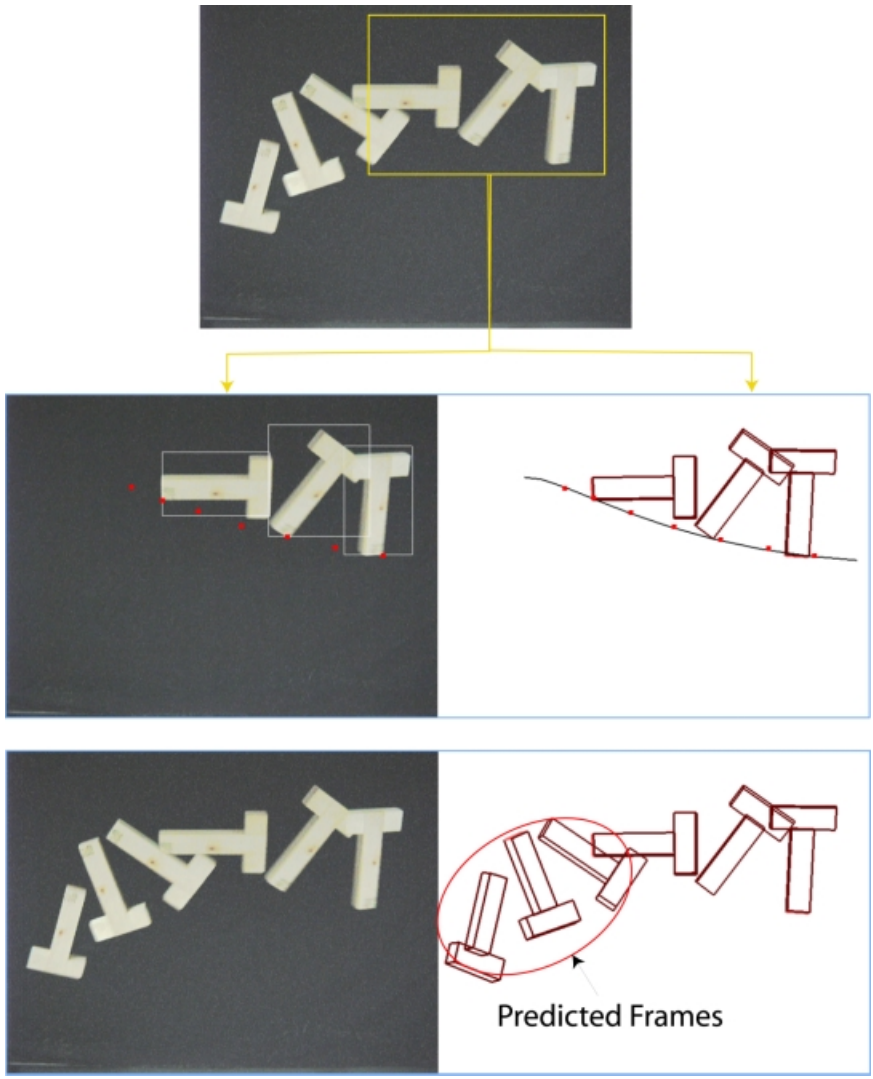

Fig. 4. Predicting the motion of tumbling bodies in video. The first row shows a long sequence of a tumbling object thrown from right to left. We select a portion of this sequence and match its motion in simulation. The second row shows a portion of the original clip contained inside the yellow box, and the corresponding frames of a simulation. We use these parameters to predict the motion of the tumbling object across the whole sequence.

Experiment 2: The objective of this experiment is to predict the motion of a rigid-body in a long video sequence of free flight from a subset of frames of the same sequence. Figures 4 and 5 show two different video clips of a rigid body in free flight, with different

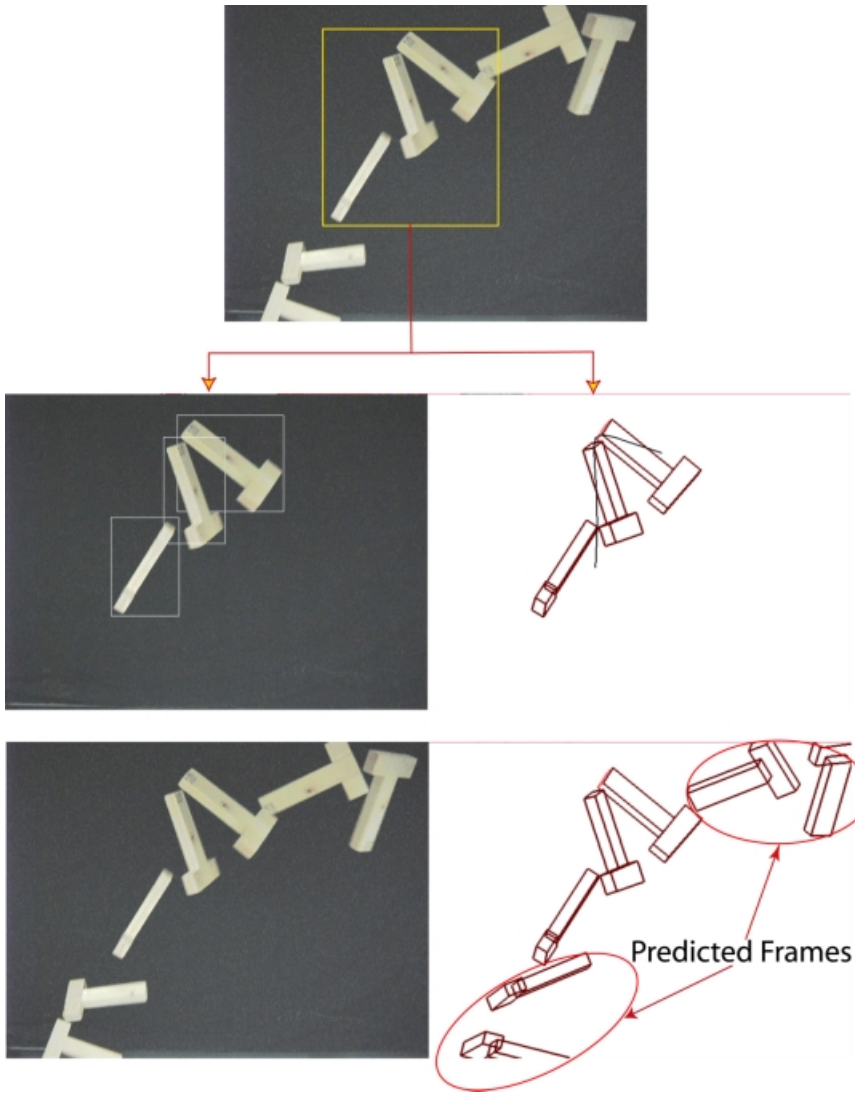

Fig. 5. Predicting the motion of a complicated tumbling object. The second row shows the match between a small portion of the video clip and a simulation. The simulation matches the video quite well for the frames on which it optimized, but the small errors propogates to larger error in the predicted frames.

motion trajectories. We match the motion of the shorter clip to a simulation and use the simulation parameters to predict the motion of the longer clip.

In Figure 4, the object tumbles about its major axis, and the essence of this motion is captured in the shorter sequence. Hence the simulation parameters computed by matching this small sequence correctly predicts the motion in the overall clip. However, the

motion of the object in Figure 5 is about the body's intermediate axis, and is much more complicated. Small errors in the estimated values of simulation parameters results in large orientation error in the predicted frames, as time increases. We see this effect in the results obtained in Figure 5.

Experiment 3: Our algorithm optimizes the direction of the gravity vector along with the motion parameters from a video sequence. Figure 6 shows results of roll correction performed using the parameters obtained from our optimization algorithm. The first row shows four frames of a video sequence captured from a camera with significant roll distortion. We compute the camera pitch and roll parameters which minimize the silhouette error. These parameters are used to warp the original sequence such that the camera vertical axis aligns with the gravity vector. The last row shows the result of this rectification. Notice that the people in the first row are in a slanted pose, whereas they are upright in the last row.

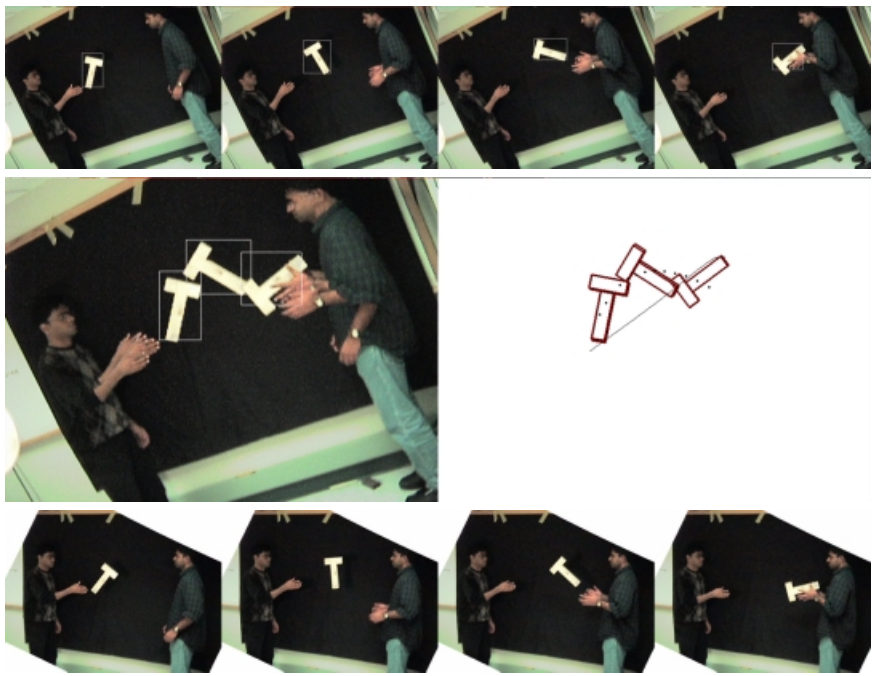

Fig. 6. Using rigid-body motion estimation to correct for camera roll. An object is thrown in the air from left to right (top row) and captured from a video camera with significant roll. The video is difficult to watch on account of the image rotation. The motion of the object is estimated (middle row), as well as the camera roll (defined by the orientation with respect to the gravity direction). The video frames are automatically rectified to correct for the camera roll.

7 Conclusion

This paper describes an optimization framework to extract the motion and underlying physical parameters of a rigid body from video. The algorithm minimizes the least square error between the silhouettes obtained in simulation and silhouettes from video at each frame. The paper presents a gradient based approach to perform this minimization.

The error space of the silhouettes is very noisy with many local minima. From our experiments, we have noticed several different combinations of initial orientations and angular velocities which result in silhouette sequences that look *similar*. This is especially true for shorter sequences, where there is not enough information (from video) to uniquely identify the true control parameters. Moreover, the performance of our simple gradient descent algorithm is sensitive to the initial guess for the control parameters. We are working on implementing better gradient based optimization algorithms and using a mixture of discrete-continuous techniques for multiple initializations. Silhouette metric has difficulties handling motions of symmetric objects, especially when multiple poses of the object project to similar silhouettes. We are investigating the use of simple color based techniques to alleviate this problem. We are interested in optimizing for the object intrinsic parameters like the location of the center of mass and the inertia matrix using our framework. We are also looking at applying our framework to other domains like cloth and articulated bodies.

Acknowledgements. The support of NSF under ITR grant IIS-0113007 is gratefully acknowledged. The authors would also like to thank Alfred Rizzi and Jessica Hodgins for their useful suggestions and comments.

References

1. D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. *In Computer Graphics, Proceedings of SIGGRAPH 94*, pages 23–34, 1994.
2. K.S. Bhat, M. Saptharishi, and P. K. Khosla. Motion detection and segmentation using image mosaics. *IEEE International Conference on Multimedia and Expo.*, 2000.
3. B.E. Bishop and M.W. Spong. Vision-based control of an air hockey playing robot. *IEEE Control Systems*, pages 23–32, June 1999.
4. C. Bregler. Learning and recognizing human dynamics in video sequences. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 8–15, June 1997.
5. S. Chandrashekhar and R. Chellappa. Passive navigation in a partially known environment. *Proc. IEEE Workshop on Visual Motion*, pages 2–7, 1991.
6. S. Chenney and D.A. Forsyth. Sampling plausible solutions to multi-body constraint problems. *In Computer Graphics, Proceedings of SIGGRAPH 00*, pages 219–228, 2000.
7. Q. Delamarre and O. Faugeras. 3d articulated models and multi-view tracking with silhouettes. *In Proc. of the Seventh International Conference on Computer Vision, IEEE*, pages 716–721, 1999.
8. O. Faugeras, Q.T Luong, and T. Papadopoulo. *The Geometry of Multiple Images*. MIT Press, 2001.
9. B.K. Ghosh and E.P. Loucks. A perspective theory for motion and shape estimation in machine vision. *SIAM Journal of Control and Optimization*, 33(5):1530–1559, 1995.

10. W.E. Grimson. *Object recognition by computer: the role of geometric constraints*. MIT Press, 1990.
11. C. Harris. Tracking with rigid models. In A. Blake and A. Yuille, editors, *Active Vision*, chapter 4, pages 59–73. The MIT Press, 1992.
12. R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.
13. D. Jacobs and R. Basri. 3-d to 2-d pose determination with regions. *International Journal of Computer Vision*, 34(2/3):123–145, 1999.
14. P. K. Khosla. Estimation of robot dynamics parameters: Theory and application. *International Journal of Robotics and Automation*, 3(1):35–41, 1988.
15. Y. Masutani, T. Iwatsu, and F. Miyazaki. Motion estimation of unknown rigid body under no external forces and moments. *IEEE International Conference on Robotics and Automation*, 2:1066–1072, 1994.
16. D. Metaxas and D. Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1993.
17. A. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):730–742, July 1991.
18. J. Popovic. *Interactive Design of Rigid-Body Simulation for Computer Animation*. Ph.D Thesis, CMU-CS-01-140, Carnegie Mellon University, July 2001.
19. J. Popovic, S.M. Seitz, M. Erdmann, Z. Popovic, and A. Witkin. Interactive manipulation of rigid body simulations. *In Computer Graphics, Proceedings of SIGGRAPH 00*, pages 209–218, 2000.
20. A.A. Rizzi and D.E. Koditschek. An active visual estimator for dexterous manipulation. *IEEE Transactions on Robotics and Automation*, 12(5):697–713, 1996.
21. J. Schick and E.D. Dickmanns. Simultaneous estimation of 3d shape and motion of objects by computer vision. *Proc. IEEE Workshop on Visual Motion*, pages 256–261, 1991.
22. K. Symon. *Mechanics, Third Edition*. Addison-Wesley Publishing Company, Reading, Massachussetts, 1971.
23. C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.
24. C. Wren. *Understanding Expressive Action*. Ph.D Thesis, Massachusetts Institute of Technology, March 2000.