Physical Rig for First-person, Look-at Cameras in Video Games

Edward T. Lixandru Victor Zordan*
University of California, Riverside

Abstract

This paper proposes a physics based model to simulate a reactive camera that is capable of both high-quality tracking of moving target objects and producing plausible response interactively to a variety of game scenarios. The virtual physical rig consists of a motorized pan-tilt head that is controlled to meet desired target look-at directions as well as an active suspension system that stabilizes the camera assembly against disturbances. To showcase its differences with other camera systems, we contrast our physically based technique with other *direct* (kinematic) computed methods from industry standard techniques.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Gaming

Keywords: camera motion, video games, physics based modeling, physics control

1 Introduction

The camera is paramount for interacting within 3D immersive video games as well as many other virtual-world applications. The camera view contributes a great deal to games because it is the window through which the viewer sees the virtual world. Even with advanced rendering, high quality animation, intuitive interaction, and refined gameplay, a video game will not be enjoyable if the viewer cannot see what is going on or, worse, if the viewer finds the displayed scene nauseating. Of course, the camera view direction, or *look-at*, must be chosen with care to effectively reveal the goings-on within the game world. Yet, in contrast to its importance, automatic camera look-at controllers have received relatively little research, especially in the computer animation domain.

The focus of this paper is physically based camera control that captures the scene from the point of view of the character. Such *first-person* cameras perform as a surrogate eye for the game player as the user's avatar moves about and interacts within the game world. It contrasts the *cinematic* camera which largely places the viewer as an outsider watching the action. In first person, the user experiences the viewpoint of the player character (PC), which allows for easy surveying for the nearby vicinity (to the PC) and can add a dramatic effect to the gaming experience, for example by limiting what is visible. However as the PC moves, the camera must react, responding and following to convey consistency and a clear view of the PC's surroundings. Our goal is to exploit a physical rig model to achieve these aims.

In addition to positioning the camera as the PC moves, first-person cameras separately dictate target look-at direction to allow the PC to move one direction while the player views another. A common example in tactical first-person shooters is *strafing* which has the PC moving left or right while the camera looks forward, e.g. attending to an enemy. Novel to this paper, we discuss the use of a physically controlled camera to synthesize the look-at trajectory based on a desired target. Our virtual physical camera creates a unique camera response function in correspondence with the movement of the PC as well as movement and swapping of target objects and/or viewing directions that dictate the desired look-at orientation. While a handful of other physics cameras have been proposed, for example to track the PC's position with a spring-damper model [Stone 2004], ours is unique in that it controls the motion of the rotation using a physical system.

Further, in addition to rotation, our physically based rig includes a suspension system that controls the position relative to its base, defining the complete rigid-body transform of the camera, i.e. where it is as well as where it is pointed. We propose to drive these six degrees-of-freedom (DOFs) through a straightforward control system that both models an active suspension, to stabilize the first-person camera, and a motorized head that dictates the look-at direction based on desired target input. Our camera's real-world counterpart would appear similar, but more sophisticated than, the amateur/pro-stunt cameras that are gaining popularity today, for example one that is secured to a human, e.g. upon the helmet of a motocross racer.

The benefit of our model is improved camera motion over the state of the art, both in terms of quality and immersion. Through a variety of examples, we show our camera in contrast to industry-standard techniques [Haigh-Hutchinson 2009]. Our camera is able to handle a host of settings including: looking at still and moving targets; swapping between various targets; and following vehicle-based and human motion capture-driven PCs that produces an enriched effect of a real-world 'stunt-like' first-person camera. Our virtual camera rig offers several contributions including improved response, especially in the case of fast moving or switching targets, as well as increased realism when rich first-person movement is to be conveyed through the camera. Further, we show that our physical rig's output mixes well through simple blending with other techniques creating a space of options which support the best both physics and traditional techniques have to offer.

2 Related Work

The majority of research in cameras for games cover topics associated with virtual cinematography [Tomlinson et al. 2000; Kennedy and Mercer 2002; Courty et al. 2003; Elson and Riedl 2007; Jhala and Young 2010; Lino et al. 2010] (among many others) and/or the artificial intelligence of planning for cameras, for example [Christie et al. 2005; Bourne and Sattar 2005; Kwon and Lee 2008; Oskam et al. 2009; Oskam et al. 2011]. The former identifies a host of problems, including shot selection, timely camera switching, and others in order to make the game experience more *filmlike*. The latter addresses a myriad of needs related to issues such as camera placement, framing for multiple targets, occlusion avoidance, and camera-path planning in complex environments. In contrast, our approach aims to improve the physical realism of the camera motion for enriched, immersive first-person games.

Standardization for first-person camera controls have led to general

*e-mail:vbz@cs.ucr.edu

adoption of a few specific techniques. It is most common to use predefined acceleration for first person shooters [Sanchez-Crespo 2004]. The velocity is clamped by a set maximum to prevent the camera from moving too quickly. For rotation, most games blend between the current angle and desired angle using one of a variety of interpolation methods. The interpolated trajectory helps the camera from being too jittery [Sanchez-Crespo 2004]. In comparison, our camera uses the desired angles as guides, but instead of interpolation, we use a controller and the momentum of the physically simulated camera rig t to transition from the current to the desired angle in a physical manner.

A good example reference for the industry practice of first-person camera control for video games is the text by Haigh-Hutchinson [2009]. While it covers a number of direct (kinematic) techniques for first-person and other cameras, less appears for use of physical models for cameras. The text does describe a spring-based technique for positioning similar to [Stone 2004], which we will contrast subsequently, and also highlights methods to resolve intersections between the camera and objects in the scene through collision forces. However, Haigh-Hutchinson warns against "rotational forces, as direct orientation control is usually required to ensure adequate framing of game objects." In contrast, our work uses rotational forces (torques) to both produce high-quality framing (look-at) and to create a sense of immersion through the physical response of the rig due to the movement of the camera's base.

The technique described by Stone [2004] uses a spring controller (servo) to move a camera through space in a continuous fashion. In contrast to our grounded rig, a flying "chase" camera results that is *pulled* behind character. As Stone points out, in a game environment the camera and target are dynamic and simple fixed-time interpolation between current and desired positions would not work, hence the spring-mass solution proposed. In contrast, our camera is attached to a separately controlled dolly or PC which grounds the camera much like its real-world counterpart. Finally, in Stone's solution look-at is computed exactly, while we use angular PD servos to rotate the camera.

Recent efforts in real-world camera control has some notable overlap with our own, namely the recent work of Carr and colleagues [Carr et al. 2013]. They describe the use of a hybrid virtual-robot camera that is employed to produce (look-at) shots for a third-person real-world event. Their goal is to create purposeful but aesthetically pleasing motion and their system uses a virtual physical-like representation to constrain their desired robot-controlled real-world camera from making abrupt changes in shots. In essence, they use a procedure like our own to purposefully constrain their real-world camera through a virtual physical representation for the camera. The effects of limits on their filmed motion parallels our own in terms of motion quality, although their goal is real-world event capture from automated robotic cameras rather than video-game camera motion.

Finally, in contrast to all other approaches, an additional goal for our physical system is to increase realism and immersion through the plausible disturbance response of the camera rig. Existing games purposefully shake the camera (through procedural methods) to create the *feeling* for the player that they are experiencing the outcome of an explosion, or first person running [Haigh-Hutchinson 2009]. However, these effects rely heavily on the skill of the animator and realize limited success due to the mismatch between the phenomena and the camera movement. In our solution, we realize a physically plausible response, that is also easily controlled using the blend camera described in Section 5.

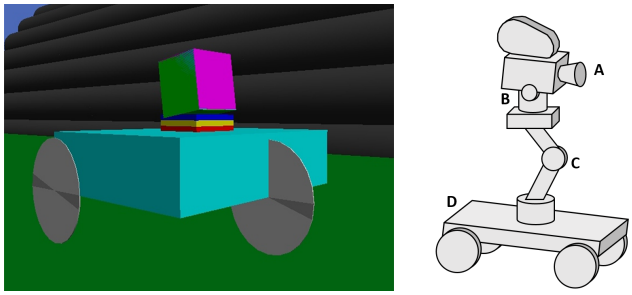

Figure 1: *Physical rig simulation (left) compared to conceptual illustration (right) with: Camera (A); Pan-Tilt (B); Stabilizer (C); and Dolly (D). The simulated unit on the left shows near co-incident green and magenta cubes at the top that are the bodies from which our camera transforms are extracted, for our physical rig (green) and for the direct/kinematic camera(s) used for comparison in our results (magenta). The thin plates below these camera "boxes" act as the simulated stabilizers with three slider joints in series. This assembly sits upon a wheeled dolly cart.*

3 Camera Set-up

Our virtual camera set-up has two configurations, with and without an incorporated dolly cart. The camera with the cart appears in Figure 1. We employ the cart to allow the camera to act in first person for a disembodied PC. We also demonstrate the camera directly attached to a moving character. The effect of the latter provides an *extreme action* feel to the footage rendered through the camera, similar to a stuntman wearing an affixed camera. The rig includes a motorized pan and tilt function to orient the camera relative to its base which we model using a single universal joint that allows rotation about the Z and Y axis (heading and pitch respectively.) Three "shock" plates actively stabilize the camera's position relative to its base allowing simultaneous damping in all three directions. The suspension rig resulting prevents high frequency propagation of motion from the base and mimics the effect of a so-called *steadicam* in the real world.

The entire setup is physically simulated using ODE (Open Dynamics Engine, www.ode.org) with nominal masses (1 kg each) for each body and ODE-computed inertias derived from the geometric shapes. Using the control system described next, we can track moving targets through the pan-tilt activation as well as damp and correct for the movement of the base.

4 Camera Control

Our camera look-at controller is based on a combination of a proportional derivative (PD) servo and a feedforward component computing generalized forces, as follows.

$$\tau = k(\theta_d - \theta_c) + b(\omega_d - \omega_c) + \tau_{ff} \quad (1)$$

where θ_d and θ_c are the desired and current angle of the camera body, ω_d and ω_c are the desired and current angular velocity of the camera, respectively. We set our desired velocity to be zero because we want the camera to come to a static state. τ_{ff} is a feedforward term that prevents the camera from falling behind fast targets (Section 4.2). This control equation is employed on each of the degrees of freedom (DOFs) including the pan-tilt and slider joints. Note, the stabilizing plates have a zero desired value. The heading and pitch desired values are computed based on the target (Section 4.1).

We tune the motion of a PD servo by changing the k and b values to achieve different effects. The value governs the force or torque and their overall and relative strengths lead to a variety of rich visuals. Not surprisingly, settings close to critical damping prove to create a high-quality general tracking, and this is employed for the sliders throughout as well as the pan-tilt DOFs for the majority cases. However, we also show in our results the impact of other choices for the pan-tilt - creating both a *lazy* slow moving camera as well as a more *frantic*, almost nervous-like visual targeting.

4.1 Desired values

We expect that the camera follows a known focus within the scene, whether it is a simple viewing direction, character, stationary object, area/space, or more complicated scenarios [Vo and Lien 2010; Vo et al. 2012]. The choice of focus is outside the scope of this paper, and we refer others to the extant literature with respect to target choice and control [Christie et al. 2008; Haigh-Hutchinson 2009]. However, beyond selecting the target look-at, we do assume there is some tolerance for deviation as the frame of the camera will subsequently be viewed by the game player. That is, the player's eyes are free to scan the viewing window to see the target, so the target need not be exactly in the middle of the frame. In fact, perfect tracking can even be unnatural, because it can seem as though the world (and the PC) is pivoting around the target.

Our system computes the controller's desired inputs, θ_d, as the necessary angles for heading and pitch that put the target in the middle of the viewing window. Neglecting roll (which assumes the up-vector is aligned with the global coordinate frame) calculation of the control inputs is straightforward. The respective heading and pitch are computed from angle between the vector of the camera position to the target and the camera's facing zero-vector, in our case along the global X, each projected onto the XY and XZ planes of the camera base. We use the outcome of these two values as the desired values for the pan-tilt control DOFs. Note, for convenience this calculation can often also be computed against the global coordinate frame, but this approach is foiled if the base rotates grossly from the global vertical axis.

4.2 Feedforward torques

To keep the target in sight when it is moving quickly, we use feedforward control on the heading and pitch torques in addition to the servo control. With τ_{ff}, the camera achieves fast tracking while allowing less stiff control gains. Our simple feedforward activates when the target is too far from the center of the viewing window, measured with error ϵ, according to the following schedule.

$$\tau_{ff} = \begin{cases} 0 & \text{if } \epsilon \leq \theta_{min} \\ \tau_{max} \frac{(\epsilon - \theta_{min})}{(\theta_{max} - \theta_{min})} & \text{if } \theta_{min} < \epsilon < \theta_{max} \\ \tau_{max} & \text{if } \epsilon \geq \theta_{max} \end{cases}$$

We implement feedforward with a linear ramp as such to allow for a smooth acceleration and prevents a jerking motion caused by adding a discontinuous value of torque to the camera. Note, feedforward is zero for the slider DOFs.

5 Blend Cameras

In our results, we compare our camera system to both direct tracking, i.e. using the exact look-at direction, and to a basic interpolation scheme, in the case of fast-moving or switching targets. However, to highlight the full spectrum of benefits from our physical camera, we include discussion of uniformly blending the outcome of our model with these direct camera alternatives. Namely, we point out there is an entire space between the physical outcome of our rig simulation and the simple tracking of exact or interpolated target look-at values. By tuning the blend weight, a game developer can enjoy any degree of physicality from the proposed rig along with some of the perfect solution offered by the direct techniques. In this section, we first discuss the direct alternatives which we use to contrast the physical camera, and then propose a simple method to blend the results to easily access solutions in the aggregate space.

A *naive* look-at camera can instantaneously change orientation and position to keep targets in the middle of the viewing window. While this simple camera will always guarantee that a target is in the exact center of the window, it will create an artificial motion to do so, snapping perfectly to the desired position and orientation for the target. If the target moves slowly and/or in a continuous fashion this camera can appear to be smooth and, at times, realistic. However, this quality is lost once the target starts changing directions and/or position rapidly. A practical example is due to a discrete, player-controlled focus in a game, e.g. where multiple enemies appear simultaneously and the player fixates on a single enemy before focusing on another. If the camera system instantaneously moves to the perfect state, it clearly produces discontinuities whenever the target selection changes during play. The camera snapping to the new target instantly is both jarring and disorienting because the player cannot place the new target in relation to the previous.

Some solutions for such jumping that are commonly used in games include: blending the target look-at directions; interpolating to create a smooth (virtual) target trajectory; and/or controlling smoothness over the camera's angular velocity directly [Haigh-Hutchinson 2009]. We experimented with several of these and in the examples described for this paper we include for comparison an ease-in/ease-out (EIEO) blend for the look-at angles with interpolation values computed using spherical-linear interpolation (slerp). We set a fixed duration of time to move the camera from the start to the finish look-at. As seen in the resulting animations, this EIEO blend responds to the target and produces a smooth animation for the camera. However, it can be difficult to tune for broad set of scenarios and also adds a "floating" quality to the camera motion (see video).

Beyond choosing one technique over another, an interesting solution is the blend of the direct and physical cameras' results. The reason is that it strikes a compromise between the benefits of both when the simple (or EIEO) tracking is too perfect and the physical camera response adds offsets to the camera motion that are beyond those allowed or desired. The latter could also be tuned, but a simple blend between the generic (critically damped) physics and the direct solution leads to a well-behaved and easily tunable space. We respectively use slerp and linear interpolation to average the rotation and position camera values from each source. We show results with an equal weighing in the accompanying animations and result plots, but higher or lower weights are also possible and will result in expected changes in the outcome.

6 Experiments

To showcase the power of our physical rig we devised a series of experiments. The first pair highlight the benefit and outcomes of fast moving targets and discontinuous switching of targets and exercise largely the motorized pan-tilt of the rig. The second pair place the camera in more extreme settings, one with a complex external influence on the camera and the other as if taken from a first-person stunt camera. In the latter pair, the camera motion of the physical rig conveys a richer more textured result that communicates to the player a more immersive experience over the direct methods.

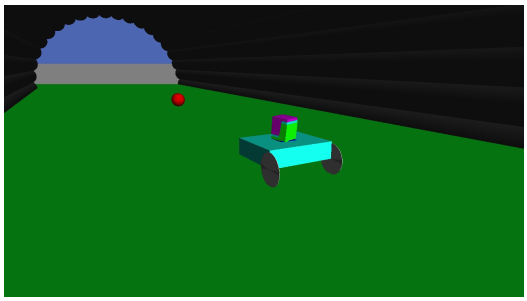

Figure 2: *Following a ball moving with discontinous position.*

6.1 Experiment 1: Target tracking

Our first experiment tracks a random moving "ball" target that randomly goes from not moving to quick jumps. This target provides us with a difficult look-at scenario as it moves. In this experiment, we employ the dolly and set it to move forward slowly in a single direction. The target is followed by a direct look-at camera (computed kinematically) which is placed at the same offset as the physical camera (i.e. moved by the dolly). See Figure 2 where the direct look-at camera is magenta, while the physical camera is green.

Footage taken directly from the simple look-at camera causes undesirable motion, having discontinuous jumps from one orientation to another. Every time the target makes a fast move, the camera snaps orientation to place the ball in the center of the viewing window. The result is jarring in comparison to the physical camera which moves in a smooth and quick path to the new desired position.

As expected, the physical camera performs by tracking the ball but not allowing discontinuous jumps. When the ball moves from one location to another, the system smoothly but quickly torques the camera to the new desired orientation. From the camera's viewpoint (through its footage) the relative path the target takes to get to the new orientation helps inform the viewer about how the target moves. The target always stays in the viewing frame and it is easier for the viewer to understand the path of the target because the target can be seen to move in relation to the background.

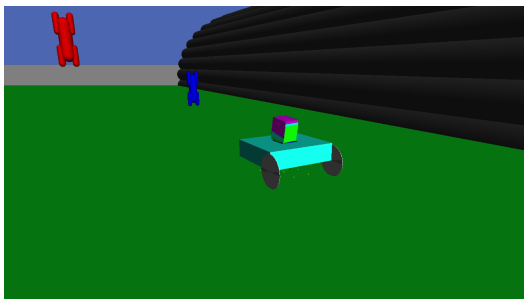

Figure 3: *Changing targets between two "enemies", one stationary, one moving. The red target is the current focus for the camera.*

6.2 Experiment 2: Switching between targets

Next we test the camera's ability to switch discretely between two targets, while we keep the position of the camera stationary. One of the targets is statically placed in the distance, while the other is close to the camera, moving in a circular motion. As one target moves closer (in front of) and farther from the other target, the switch arc length varies in this experiment. We allow the "player" to freely change the target to show how the camera responds.

Because an exact camera would immediately change orientation to the new target, causing disorientation in the viewer, as described in the previous experiment, we instead highlight the effect of the EIEO slerp trajectory in the case of switching. In this experiment, if the time between the targets is long enough, the EIEO trajectory's footage is satisfactory (although it appears to the viewer that the camera is floating.) However, if the time is short, shorter than the interpolation time duration, the EIEO animation being in mid-interpolation lead to undesirable effects with jumpy footage. Although we can select the duration to be any value, or can perhaps compute it automatically based on the arc-length of the error, no easy solution upholds quality blends in the relatively simple space of fast/slow and near/far target switches.

In contrast, the physical camera provides different response based on each switch's characteristics. Regardless of whether the targets were far or close (to each other), the controller responded with the same visual quality. When far away, the motion to the new target was faster, as the torques applied were greater, but the resulting motion remained sensible and the target easily trackable in the resulting footage. Also, if the player switched the target quickly, the PD-servo was able to smooth effect the camera regardless of what state it was in from the previous transition.

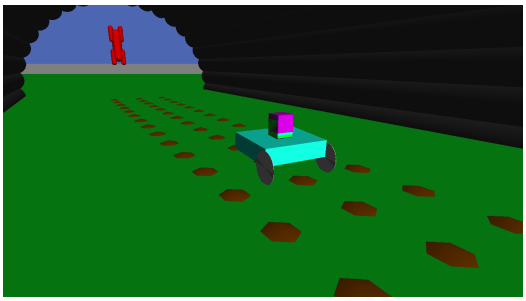

Figure 4: *Tracking a static viewing direction while cart-base traverses a rough terrain.*

6.3 Experiment 3: Tracking over rough terrain

Our next example consists of the camera on the cart moving rapidly over rough terrain. The camera looks at the horizon as the cart traverses the bumps in its path. For reference, we include a visual target some distance in front of the camera in order to show how much the camera transform changes. The point of this camera is to highlight how the moving base affects the rig.

Of course, the exact look-at keeps the camera aimed perfectly as expected. However the camera's tracking nearly smooths out all the effects of the bumps (except for the position displacement of the cart which is embedded into our definition of target angle.) In the resulting footage, it appears as though the camera is moving over a smooth road instead of rough terrain. The target reference shape moves very little, giving the impression that the camera is mounted on a stable body. In fact, if the bumps had been more similar in color to the surroundings, the viewer would be hard-pressed to figure out that there was rough terrain in the scene at all.

In contrast, the physics based camera provides an exciting "ride" for the viewer. The bumpiness of the road is translated to the camera, but it is damped enough to see the scene as it goes by. The target shape in the distance does move up and down in the viewing window due to the cart going over the bumps giving the impression of the rough terrain. The scene goes from boring to exciting because the physics camera responds to the rough terrain. This immerses the viewer in the scene, because the camera motion is believable given

the context. Even if the bumps blended into the surroundings, the camera motion would inform the viewer that the cart was moving over rough terrain.

Finally, as described in Section 5, if the taste of the game maker suggests that the physical camera is "too much" the blend animation shown highlights one of a number of solutions for the space between the simple direct and completely physical response.

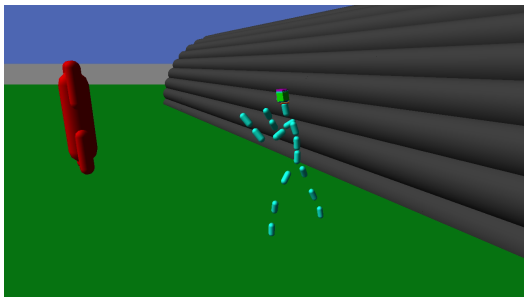

Figure 5: *Tracking a moving target while mounted to a motion capture character.*

6.4 Experiment 4: Virtual stunt camera

For our fourth and final example, we place the camera setup on a motion capture character doing a karate kata, a sequence of fighting moves. We shrank the camera setup in order to fit on the motion capture characters head (see Figure 5). Next, we play the motion capture data and have the camera track a target (supposed enemy) that is moving simply (procedurally) in a semi-circle in front of the character.

The exact look-at camera tracks the target perfectly over the entire motion playback. However, the target appears stationary as the world circles around it. While the target comes closer and farther based on the displacements of the motion capture data, from the footage's point of view, the camera looks like it is moving back and forth on a rod, appearing to the viewer as if the camera is arbitrarily zooming in and out. The exact tracking makes the camera mounting look as though it was not on the motion capture character at all.

In contrast, the physical camera provides an enriched tracking experience. Much of the detail of the karate kata comes through the camera footage. The target is largely tracked well, except when the head whips around very quickly, but its clear that the camera is mounted on the motion capture character, because the target moves tellingly from the center of the frame. The movement is more dramatic, as the camera has to quickly change position when the character does a fast movement. The zooming in and out of the target is gone. While the PC does get closer to the target during kicks, it is clear that it is because the character is moving and not the camera zooming in on the target. The physical rig suspension keeps the motion from being too jittery, but lets some of the characters action come through. In this animation in particular, rapid turns need the feedforward component to keep the target in frame, although it does "escape" briefly from time to time due to the severe action and extreme disconnect between the pre-recorded kata and the targets hemispherical trajectory.

7 Analysis and discussion

Figure 6 demonstrates that the heading response curve of our physics camera remains smooth and continuous, even when the desired orientation is discontinuous. Our physically based camera

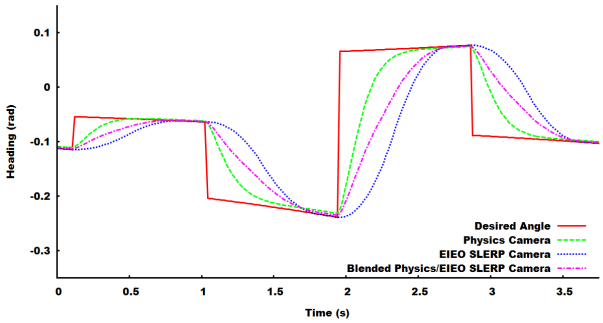

Figure 6: *Comparison of heading response curves for the pure physics, EIEO slerp, and a blended solution.*

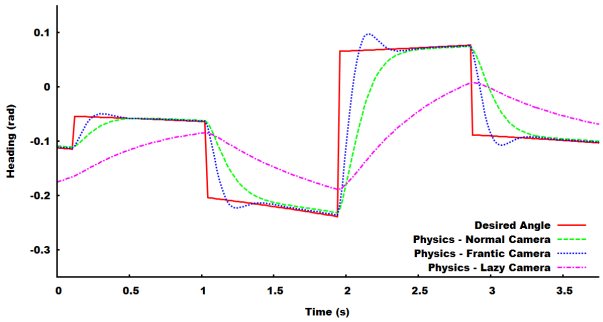

Figure 7: *Comparison of heading response curves for the generic physics in contrast to a lazy and frantic effect.*

rig enforces continuous position and velocity trajectories because it applies force/torque to the camera dynamics. While the EIEO slerp curve also smoothly interpolates between target jumps, its response timing is independent of the discontinuity size. In contrast, the physics camera depends integrally on the discontinuity, which leads to more a dynamic response.

Figure 7 demonstrates the flexibility of the physics camera by showing different response curves that the camera can achieve. The "normal" physical camera ($k = 2$, $b = 1$) is the same as the one in Figure 6. The frantic effect is a physics camera ($k = 5$, $b = 1$) that simulates a "nervous" response to the target changing position. In practice, a nervous PC may keep track of an enemy she is afraid of, but in her frantic effort to keep track of the enemy, the PC may accelerate so quickly she does not have enough time to slow down when she reaches her target, overshooting slightly. The opposite can also be true if we overdamp the camera ($k = 0.5$, $b = 1$) . This simulates a lazy character slow at her job of keeping track of a target. The response curve is sluggish and does not even reach the desired angle by the time the target jumps again. Both types of response curves are continuous and keep the target in the viewing window.

Figure 8 demonstrates that the physics camera is more influenced by the context of the scene than the direct look-at camera that follows the desired angle. This difference is due to the physical rig being influenced by orientation and position of the head of the motion-capture character that acts as its base. The look-at camera following the desired angle shows little influence. Because the physics rig is affected by the motion of the motion capture base along with the camera suspension and pan-tilt setup, its trajectory is non-trivial and cannot be easily represented by a procedural/interpolation function. Finally, while the physics is smoother, it also adds detail from

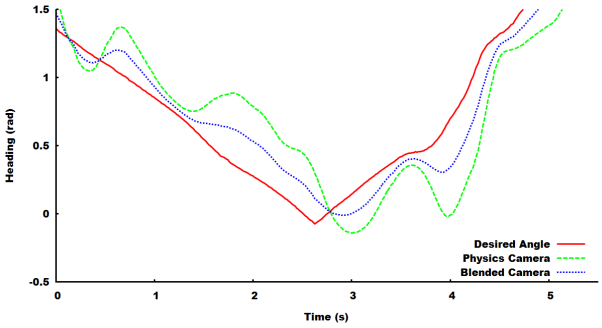

Figure 8: *Comparison values of heading for desired angle, physics, and blended camera in the motion capture example.*

the motion capture missing in the camera footage created from the direct desired-angle look-at.

Of course, our physics camera also has drawbacks. The major one is that the camera requires a physics engine and simulation time. While this may not matter in an offline simulation, processor time is at a premium for real-time video games. Even though our simulation is efficient and runs much faster (3X) than real time, the direct look-at or EIEO cameras described are simple calculations that cost little in comparison. Another drawback is that even with feedforward control, the same limitations that keep the physical system realistic constrain it from always keeping the target in the viewing window, especially if the base or the target move very rapidly. While losing the target may simulate an actual real-world response to such extreme scenarios, it may be more desirable to have the target in the view window. However, the blended camera can offer a solution here as it allows the animator to choose a blend that conveys some physics but also keeps a target in the viewing window. Finally, much of the commentary in this paper would benefit from thorough evidence-based perceptual experimentation and, although it was beyond the scope of this paper, we are encouraged to pursue this effort in future work.

8 Conclusion

In conclusion, we demonstrate a system that controls both position and orientation for a camera using physical simulation. The resultant look-at camera provides a smooth response while tracking moving targets, switching targets, going over rough terrain or tracking motion capture data. Through a number of experiments, we highlighted the ways in which the result is superior to current direct solutions for first-person, look-at cameras. We also show that we can blend the described physics camera motion with other camera motion in order to exploit the benefits of both. Our hope is to inspire efforts that will lead to further exploration in the topic of physically based camera systems.

Acknowledgments

Partial support for the primary author came from UCR graduate division. The authors thank Mika Lai and Tamar Shinar for their input on the project.

References

BOURNE, O., AND SATTAR, A. 2005. Applying constraint weighting to autonomous camera control. *AIIDE 5*, 3–8.

CARR, P., MISTRY, M., AND MATTHEWS, I. 2013. Hybrid robotic/virtual pan-tilt-zom cameras for autonomous event recording. In *Proceedings of the 21st ACM international conference on Multimedia*, 193–202.

CHRISTIE, M., MACHAP, R., NORMAND, J.-M., OLIVIER, P., AND PICKERING, J. 2005. Virtual camera planning: A survey. In *Smart Graphics*, Springer, 40–52.

CHRISTIE, M., OLIVIER, P., AND NORMAND, J.-M. 2008. Camera control in computer graphics. In *Computer Graphics Forum*, vol. 27, 2197–2218.

COURTY, N., LAMARCHE, F., DONIKIAN, S., AND MARCHAND, É. 2003. A cinematography system for virtual storytelling. In *Virtual RealityTechnologies for Storytelling*. Springer, 30–34.

ELSON, D. K., AND RIEDL, M. O. 2007. A lightweight intelligent virtual cinematography system for machinima production. In *AIIDE*, 8–13.

HAIGH-HUTCHINSON, M. 2009. *Real Time Cameras: A Guide for Game Designers and Developers*. Morgan Kaufmann.

JHALA, A., AND YOUNG, R. M. 2010. Cinematic visual discourse: Representation, generation, and evaluation. *Computational Intelligence and AI in Games, IEEE 2*, 2, 69–81.

KENNEDY, K., AND MERCER, R. E. 2002. Planning animation cinematography and shot structure to communicate theme and mood. In *Proceedings of the 2nd international symposium on Smart graphics*, ACM, 1–8.

KWON, J.-Y., AND LEE, I.-K. 2008. Determination of camera parameters for character motions using motion area. *The Visual Computer 24*, 7-9, 475–483.

LINO, C., CHRISTIE, M., LAMARCHE, F., SCHOFIELD, G., AND OLIVIER, P. 2010. A real-time cinematography system for interactive 3d environments. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 139–148.

OSKAM, T., SUMNER, R. W., THUEREY, N., AND GROSS, M. 2009. Visibility transition planning for dynamic camera control. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 55–65.

OSKAM, T., HORNUNG, A., BOWLES, H., MITCHELL, K., AND GROSS, M. H. 2011. Oscam-optimized stereoscopic camera control for interactive 3d. *ACM Trans. Graph. 30*, 6, 189.

SANCHEZ-CRESPO, D. 2004. *Core Techniques and Algorithms in Game Programming*. New Riders Education.

STONE, J. 2004. Third person camera navigation. In *Game Programming Gems 4*, Charles River Media.

TOMLINSON, B., BLUMBERG, B., AND NAIN, D. 2000. Expressive autonomous cinematography for interactive virtual environments. In *Proceedings of the fourth international conference on Autonomous agents*, 317–324.

VO, C., AND LIEN, J.-M. 2010. Following a large unpredictable group of targets among obstacles. In *Proceedings of the International Conference on Motion in Games*, Springer.

VO, C., MCKAY, S., GARG, N., AND LIEN, J.-M. 2012. Follow a group of targets in large environments. In *Proceedings of the International Conference on Motion in Games*, Springer.