Eurographics Workshop on Intelligent Cinematography and Editing (2015)
W. Bares, M. Christie, and R. Ronfard (Editors)

Film Ties: An Architecture for Collaborative Data-driven Cinematography

W. Bares[1] D. Schwartz[2] C. Segundo[1] S. Nitya[1] S. Aiken[1] and C. Medbery[1]

[1]College of Charleston, Charleston SC, USA
[2]Marist College, Poughkeepsie NY, USA

Abstract

The ability to store, share, and re-use digital assets is of primary importance in the film production pipeline. Digital assets typically include texture images, three-dimensional models, and scripts for creating special effects such as water or explosions. Despite the growing use of virtual cinematography in the film production pipeline, existing tools to manage digital assets are unable to harness the knowledge inherent in the creative composition and editing decisions made by cinematographers. This work introduces Film Ties, a new form of visual communication in which a first artist creates a virtual camera composition whose visual composition properties are stored into a database allowing other artists to adopt that composition for use in their own different virtual scenes. On adopting a composition, the system computes a comparable composition of subjects situated in a second virtual environment. Artists can post comments on shared compositions using text or propose additional compositions that adapt or improve upon the original composition. The stored compositions and suggested edits also serve as an educational resource for junior filmmakers and film students.

Categories and Subject Descriptors (according to ACM CCS): H.2.4 [Systems]: Multimedia databases—; Group and Organization Interfaces H.5.3 Computer-supported cooperative work

1. Introduction

Virtual cinematography is now established as part of the film production pipeline and finds extensive use in planning and filming compelling shots as evidenced by recent Academy Awards for Best Cinematography, namely Emmanuel Lubezki for "Gravity" (2013), Claudio Miranda for "Life of Pi" (2012), Robert Richardson for "Hugo" (2011), and Mauro Fiore for "Avatar" (2009). Production studios are especially concerned with storing, sharing, and re-using digital assets which typically consist of images, music, three-dimensional virtual prop models, rigged three-dimensional characters, and scripts used to generate special effects such as realistic water [Sof15]. Despite the importance of virtual cinematography in the production pipeline, there exists no method for harnessing the knowledge of composed camera shots as a digital asset that can likewise be stored, shared, and re-used.

This paper proposes a new mode of communication in which the essence of a first composition is shared, enabling others to instantly obtain, within their own distinct scenes, new virtual camera compositions which incorporate the composition features of the selected shared image. This workflow enables studios to create databases of camera compositions, which can be re-used in other productions. In re-using compositions, the system invokes a smart camera solver which computes a comparable shot in the virtual 3D scene of the new production. The Film Ties workflow also includes a social mode in which artists can choose to browse compositions of selected favorite artists. Artists can post text comments on one another's compositions. Comments posted on a first composition may include a second composition that serves as a suggested improvement or alternate shot.

2. Film production pipeline

In the modern film production pipeline, storing, sharing, critiquing, editing, searching, and browsing digital assets is essential. Digital assets are stored in networked systems to enable sharing and collaboration. The pipeline's asset browser enables the artists to locate and preview digital assets. Comments and suggested changes are recorded so they can be directed to the appropriate persons and to maintain a record of the edits made to the work. Archived assets have value

DOI: 10.2312/wiced.20151081

for re-use in sequels or as a learning resource or point of reference for artists working on other projects [Dun14].

This synopsis of the film production pipeline suggests that a system that aims to store camera compositions must provide the following features:

- Networked to allow sharing of compositions
- Persistent data storage of compositions
- Browse and search compositions
- Record comments and suggestions for review
- Tag content by author and time stamp
- Re-use of compositions

2.1. Related work

Prior works have proposed automated methods to assist users in composing and editing shots in computer-generated environments and in sharing and editing digital images and videos in social spaces.

2.2. Virtual camera model

In Computer Graphics practice, a camera is typically represented by a set of properties that define its position (x,y,z), orientation, aspect ratio, and lens field of view. The location, size, and orientation of target objects may also be represented to allow a system to compute camera properties to view desired targets [GA88]. The virtual camera and target objects may exist purely in a computer-generated virtual scene or a real-world scene whose properties have been digitized by some means.

2.3. Social and database systems

Many prior works dealt with database systems to create, share, and comment upon artifacts of digital media.

FotoFile provides an interface to organize and efficiently browse and retrieve photographs and audio according to metadata tags. Users can tag content according to the content of the image or tag as favorites. It includes facial recognition to help automate creation of tags to identify persons appearing in a photograph [KPC*99]. The Social Camera leverages metadata tags including GPS, gyroscope orientation, time, and a mobile device's camera settings to retrieve and propose highly-rated pictures that a photographer could consider when located in a similar position, facing direction, and time of day [BMS11]. The OSCAR system aids photographers by retrieving exemplary images from its database by searching for exemplars having similar subject matter and composition features [YSQ*12]. The Family Video Archive enables family members to share, browse, and annotate videos and includes facilities to automate annotations using metadata tags that list members of a family and dates [AGL03].

Like these prior works, our system incorporates a social component that allows users to share and comment on content. Film Ties differs from these prior works in that the content being shared represents the composition properties that can be automatically adapted to different 3D scenes.

2.4. Intelligent virtual cameras

Research in intelligent virtual cinematography aims to compute virtual camera configurations by position, orientation, and lens properties to visualize objects or events that exist in computer-generated virtual environments [CO09]. Early works computed virtual camera configurations using mathematical functions of the position, size, and orientation of a subject in addition to the desired relative view angle and distance by adding an appropriate displacement vector to a point in the subject [Fei85]. Blinn extended this approach to compute camera placements of one or two subjects so that the subjects appeared at desired points on-screen [GA88]. The Toric Manifold solution proposed by Lino and Christie generalizes Blinn's method to efficiently computes camera properties that will result in compositions having one or two specified target objects appearing at the desired on-screen locations [LC12]. The Director's Lens computes and proposes a gallery of suggested cameras whose visual composition properties reflect previously recorded shots and editing patterns [LCRB11].

Our work is most similar to Director's Lens [LCRB11] in that an intelligent camera system computes and proposes a gallery of suggested cameras using the visual composition properties of previously saved shots. Film Ties introduces collaborative, social, and database functionality.

2.5. Production pipeline tools

Existing tools to support the film production pipeline can store, search, browse, and re-use digital assets, including texture images, 3D models, and procedural scripts for executing special effects or camera behaviors such as panning to following a target [Sof15]. Other toolkits offer procedures to simulate commonly used camera behaviors such as orbiting a target or simulating the jitter of a camera mounted to a vehicle [Ani15]. Existing industry solutions provide no means for users to create and share knowledge of camera compositions.

Presently, no prior work address all of the requirements for the proposed asset management of camera compositions in the film production pipeline.

3. Example interaction

This first example illustrates how users can share compositions and post and view comments to shared compositions. Mark logs into the Film Ties web page, creates an account,

then opens an animated scene of two birds playing volleyball. He presses the take a picture button (top-left and top-right sides of the GUI). The client transmits the visual properties of his composition to the server. Mark then sees a thumbnail image of his composition in the My Shots Gallery in the lowermost part of the GUI (Figure 1).

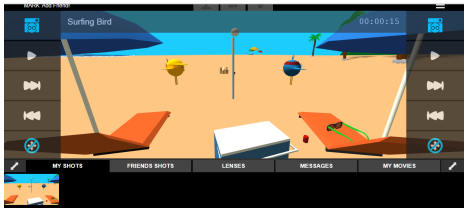

Figure 1: *Mark saves a composition.*

Jane logs into Film Ties and selects Mark as a user whom she wishes to follow. When Jane clicks on the Friend's Shots Tab, she sees Mark's composition in her gallery. Jane likes the way Mark frames the action between the umbrellas, but thinks that the direct side angle makes the ball appear to stick to the top of the net post. She increases the camera angle to keep the framing and show the ball and net from a better angle (Figure 2). She taps the Messages Gallery tab, then saves her composition as a comment upon Mark's composition along with a text message.

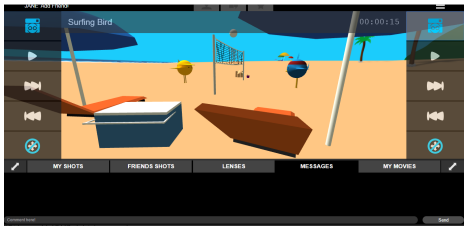

Figure 2: *Jane suggests an alternate composition to Mark.*

Mark reviews his compositions by touching the My Shots Gallery tab. He notices that one of his compositions has a comment so he touches that composition's thumbnail image, then touches on the Messages tab. His Messages Gallery shows Jane's alternate composition and text note (Figure 3).

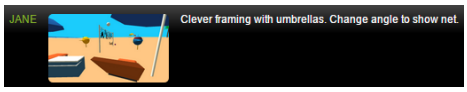

Figure 3: *Mark views Jane's suggested composition and comment.*

In a second scenario, a film instructor has created a composition set in the volleyball scene to share with the film class (Figure 4).

The film students elect to follow the instructor so that they

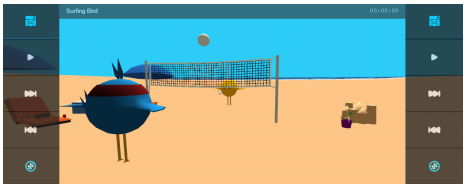

Figure 4: *Film instructor shares a composition.*

can view the shared composition. In this example, the students are working in a different virtual 3D scene consisting of flying blimps. Film Ties invokes its smart camera solver which analyzes the staging of the blimp scene to compute a composition of the two blimps (Figure 5) that preserves the composition properties of the instructor's example (Figure 4).

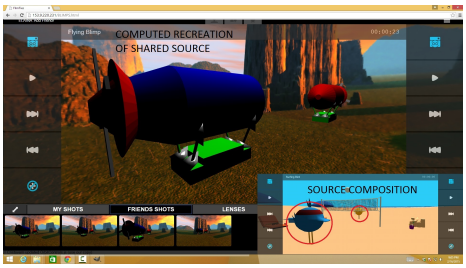

Figure 5: *Solver computes a similar composition.*

4. System overview

The Film Ties system is a client-server architecture with a lightweight client managing the GUI and scene rendering and sending and receiving messages via a socket connection with the server. The server processes client messages, accesses the database, computes virtual camera viewpoints, and transmits result messages to clients (Figure 6).

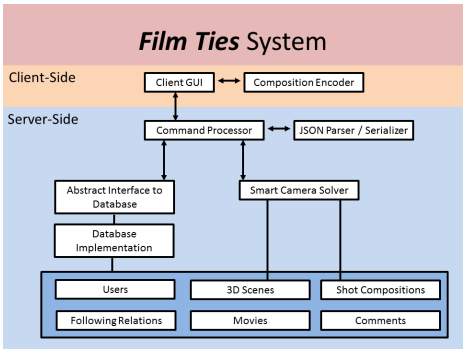

Figure 6: *Film Ties Architecture*

5. Client-side system

The Film Ties client user interface enables its users to register, login, logout, access VCR-style animation record and playback controls, and utilize multiple galleries to browse and search compositions. Each gallery displays a different set of content ranging from static shot compositions (organized by current user or other "friend" users), messages and suggested alternate compositions, and frames recorded for the movie. VCR buttons provide controls to play/pause the animation, record a snapshot, and step ahead or back in animation time. All galleries include buttons to select the desired sorting attribute such as by user who created the content or time created.

5.1. Encoding compositions

When a user saves a snapshot, the client encodes, for each user-designated subject, a description of how it appears in a normalized picture frame with bottom-left (-1.0, -1.0) and top-right at (1.0, 1.0).

- Location of the subject's center
- Bounding rectangle of the projection of the subject's bounding box
- Subject's front-facing vector in local camera coordinates
- Fraction of the subject's bounding rectangle lying in the frame
- Fraction of the subject's bounding box that is occluded

The staging encodes the three-dimensional layout of the subjects as they are arranged in the virtual set and includes:

- Position of the subject's center
- Front-facing direction vector of subject
- Orientation transform
- Axis-aligned bounding box of the subject

5.2. Client messages

The client encodes outgoing messages as JavaScript Object Notation (JSON) text where all messages include a header that identifies the command type, user making the request, state of the GUI selections of current gallery, scene, movie, selected snapshot, and current movie frame number. Commands that involve saving a snapshot give the encoded 2D composition properties and 3D staging layout. The listing in Figure 7 shows an abbreviated version of the JSON message that Jane's client sends when she suggests the alternate composition from Figure 4.

5.3. Refresh on results received from server

On receiving a response message from the server, the client updates its display to reflect the result of the request. Figure 8 illustrates the JSON message that Mark's client receives when Mark requests to see Jane's suggested alternate camera

```
{"header":{"command":"CMD_SAVE_SNAPSHOT_COMMENT",
 "userContext":{"userId":"JANE", "sceneId":1,
 "movieId":"Volleyball",
 "movieFrameNumber":0,
 "selectedSnapshotId":1 },
 "playerCompositions":
  [{"name":"Blue_bird",
    "position":{"x":0.3,"y":0.2},
    "rectangle":
      {"min":{"x":0.2,"y":-0.1},
       "max":{"x":0.3,"y":0.5}},
    "frontDirection":
      {"x":0.56,"y":0.83,"z":-0.04},
    "inFrame":1.0,
    "occlusion":0.0 }, ...
  "playerStagings":
   [{"name":"Blue_bird",
     "position":
        {"x":-7.5,"y":2.0,"z":0.0},
     "frontDirection":
        {"x":-7.4,"y":2.2,"z":0},
     "orientation":
        {"x":0,"y":-0.71,"z":0,"w":0.71},
     "boundBox":
        {"min":{"x":-8.8,"y":-0.0,"z":-2.2},
         "max":{"x":-6.3,"y":4.0,"z":2.2}},
     ... }],
   "commentText":
      "Good framing between umbrellas.
      Change angle to better show net."}
```

Figure 7: *Example client message to save a snapshot comment*

angle. The gallery entry identifies the user, Jane, the timestamp when Jane created the suggestion, and the unique snapshotID of Jane's new composition. It also gives the client the camera properties that were computed by the camera solver which the client will use to render a thumbnail image into Mark's Messages gallery. The field commentAppliesToSnapshotId specifies the unique ID of Mark's original snapshot. Lastly, we see Jane's comment that accompanies her suggested alternate composition.

6. Server-side system

The server is implemented as multi-threaded C++ code that maintains a queue of active client socket connections and a queue of received messages. On receipt of a message, the server converts the incoming JSON message into an equivalent C++ structure. It inspects the command type attribute and dispatches the appropriate command handler routine, which accesses the database, computes camera solutions, and packages the result object. Lastly, the server serializes the result object into JSON text and sends the message to the client which initiated the incoming message.

```
{"entryType":"SNAPSHOT_COMMENT",
 "userId":"JANE",
 "socialScore":1,
 "createdDate":"2015-03-15T18:25:43-05:00",
 "snapshotId":2,
 "camera":{
   "position":{ "x":-4.4,"y":2.6,"z":-16.4},
   "quaternion":{ "x":0.0097,"y":0.9887,
                  "z":0.0742,"w":-0.1295},
   "fov":60, "aspect":3.323,
   "near":1,"far":1000},
 "commentId":1,
 "commentAppliesToSnapshotId":1,
 "commentText":
   "Good framing between umbrellas.
   Change angle to show more of the net."}
```

Figure 8: *Example server response*

6.1. Database

The database is organized in a two-layer design consisting of an abstract interface and a Microsoft SQL Server 2014 implementation. It also contains a baseline simulated database built upon the C++ standard collection library. The database contains twelve principal tables, with the ones most relevant to the previous example interactions explained below.

The Users table holds descriptive information about the users of our application. This information includes the UserID, password and affiliation of the user, along with the user's ImpactScore, which is the calculated social ranking of this user's contributions to the community, and the date the user was added.

Since our application is a social application, the database includes a Following table, which keeps track of "who is following whom". This table consists of two fields, a UserID and a FollowerID. In this table, the record (X, Y) reflects the fact that user Y is a follower of user X. Of course, this table could also include the record (Y, X), which would mean that user X is also following user Y.

The Movies table describes each movie to which our users have access. Movies are identified by a MovieID and are given a MovieName and MovieGenre. We also track the UserID of the user who created the movie, the frames-per-second rate of the movie (MovieFPS), the calculated SocialScore for the movie, the level of MovieAccess, and the CreatedDate.

The table used to hold Compositions stores a description of what we want to see in the 2-D picture frame. This includes the CompositionID and number of subjects, along with a JSON string that describes how subjects appear in the 2-D picture frame by giving the position, bounding rectangle, facing direction, fraction in frame, and fraction occluded (FrameComposition). The table also contains enumerated values for describing the cutting height (close-up, medium shot, extreme long shot, etc.) of the subjects (CompositionType), the relative height of the camera to the subjects (CompositionHeight), and how the composition is lit (CompositionLighting).

The Snapshots table describes a single, non-moving camera image or frame that represents a description of how select characters and props are artfully composed for a given staging and composition. Uniquely identified by a SnapshotID, each record in the table also contains a Description and the UserID of the user who saved the snapshot. An optional field, OriginatingSnapshotID, identifies the existing snapshot (if any) that was adopted by this user and was the basis for this snapshot. The SceneContentID, StagingID, CompositionID, LensID, LensFocalLength, LensFocalRatio, FilmFormatID, SocialScore, CreatedDate, and SnapshotAccess that are associated with this snapshot are also stored.

A MovieFrame represents the use of a specified snapshot to film a given moment of animation time in a movie. This table contains a MovieFrameID, the SnapshotID that was used to film this frame, the UserID of the person who made this frame, the MovieID to which this frame belongs and the CreatedDate. It also includes the AnimationTime, which captures the moment in time (in frames counted from the start of animation) when this snapshot occurs in the animation.

Comments and feedback are collected in the SnapshotComments table. Each record in the table contains a unique CommentID, the SnapshotID to which the comment applies, the UserID of the person making the comment, the complete CommentText and its SnapshotCommentType such as suggest alternate, improvement, or other, the CreatedDate and the SocialScore of the comment, which is initialized to 1 for new comments. It also contains an optional CommentSnapshotID, which is used whenever an alternate snapshot has been created to show how the given snapshot might be reshot.

The MovieFrameComments table is analogous to the SnapshotComments table, with "Snapshot" being substituted with "MovieFrame" throughout. An additional field, AnimationTime, is included to represent the moment of the animation at which the comment is to apply. This enables users to suggest one or more alternate shots or edits for a given moment in time in a movie.

The system defines a set of transactions which involve reading or writing data from the tables. The available transactions are expressed as an abstract C++ class or interface that defines the method names, parameters, and return values. This abstraction layer enables implementers to deploy the system using any suitable database engine (such as Microsoft SQL Server as in our prototype).

6.2. Virtual camera solver

When the client requests one or more stored snapshots, the server must determine whether it can re-use the camera prop-

erties stored with a snapshot or if it must compute camera properties that re-create that snapshot's composition in the user's current 3D scene. For snapshots in the Messages Gallery and My Movie Gallery, the system responds with the stored camera properties. For snapshots in the My Shots or Friend's Shot's Galleries, the system invokes the smart camera solver to analyze the given staging layout of the user's current 3D scene and the visual composition properties of a given snapshot to compute a suitable comparable composition. The Smart Camera solver consists of an updated version of the Toric Manifold solution proposed by Lino and Christie, which computes camera parameters that will result in compositions having one or two specified target objects appearing at the desired on-screen locations [LC12]. Lino and Christie have provided an updated version of the solver that can also compute solutions in which one principle target either appears at a specified height on the screen or is viewed from a specified relative view angle between the target's "front face" and the camera. The client specifies which set of virtual actors in the current scene will play the roles for the subjects in the source composition that is to be re-created.

6.3. Transmit results to client

The server creates a return message that includes a header that identifies the client command request to which this server-generated message responds. The message also includes a status indicator (success or failure) along with optional error message text. A message may include a payload that provides additional information to the client such as what is shown in Figure 8 that specifies how to display a suggested composition and its associated text comment.

7. Conclusion

Film Ties introduces an architecture which satisfies the required functionalites of storing, browsing, searching, and reuse of camera compositions. It also provides a mechanism for commentary and suggestions and tagging content by author and timestamp. The content can be explored to faciliate learning from past movie projects. Film Ties also introduces a new mode of social communication, in which the essence of camera compositions can be shared, thus enabling powerful new creative workflows for film production studios and film educators.

8. Future work

We are planning a large-scale study of the system involving users with professional film experience, film educators and students. We also intend to implement the continuity assist features found in Director's Lens, whereby computed cameras conform to cinematic continuity relative to preceding shots [LCRB11]. The client-server design makes it attractive to implement clients for content-creation packages and game engines. Aspects of this work are protected by US and European patents.

9. Acknowledgements

We thank Elaina Cole for video editing and video narration, Clinton Medbery for video editing and coding the animated blimp scene, Malik Moore and Marcos Senna for acting, and Ian Dilling for assisting in configuring the wireless network. Thanks also to Chris Benson and William Blanchett for contributing 3D modeling and animation. Thanks to Christophe Lino and Marc Christie for sharing their code and expertise of the Toric Manifold camera solver.

References

[AGL03] ABOWD G. D., GAUGER M., LACHENMANN A.: The family video archive: An annotation and browsing environment for home movies. In *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval* (New York, NY, USA, 2003), MIR '03, ACM, pp. 1–8. 2

[Ani15] ANIMATIONS C.: A bibliography of literate programming. http://www.craftanimations.com, 2015. 2

[BMS11] BOURKE S., MCCARTHY K., SMYTH B.: The social camera: A case-study in contextual image recommendation. In *Proceedings of the 16th International Conference on Intelligent User Interfaces* (New York, NY, USA, 2011), IUI '11, ACM, pp. 13–22. 2

[CO09] CHRISTIE M., OLIVIER P.: Camera control in computer graphics: Models, techniques and applications. In *ACM SIGGRAPH ASIA 2009 Courses* (New York, NY, USA, 2009), SIGGRAPH ASIA '09, ACM, pp. 3:1–3:197. 2

[Dun14] DUNLOP R.: *Production Pipeline Fundamentals for Film and Games*. Focal Press, February 2014. 2

[Fei85] FEINER S.: Apex: An experiment in the automated creation of pictorial explanations. *IEEE Comput. Graph. Appl. 5*, 11 (Nov. 1985), 29–37. 2

[GA88] GRAPHICS I. C., APPLICATIONS: Where am i? what am i looking at? *IEEE Comput. Graph. Appl. 8*, 4 (July 1988), 76–81. 2

[KPC*99] KUCHINSKY A., PERING C., CREECH M. L., FREEZE D., SERRA B., GWIZDKA J.: Fotofile: A consumer multimedia organization and retrieval system. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 1999), CHI '99, ACM, pp. 496–503. 2

[LC12] LINO C., CHRISTIE M.: Efficient composition for virtual camera control. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2012), SCA '12, Eurographics Association, pp. 65–70. 2, 6

[LCRB11] LINO C., CHRISTIE M., RANON R., BARES W.: The director's lens: An intelligent assistant for virtual cinematography. In *Proceedings of the 19th ACM International Conference on Multimedia* (New York, NY, USA, 2011), MM '11, ACM, pp. 323–332. 2, 6

[Sof15] SOFTWARE S. E.: Side effects software. http://www.sidefx.com, 2015. 1, 2

[YSQ*12] YAO L., SURYANARAYAN P., QIAO M., WANG J. Z., LI J.: Oscar: On-site composition and aesthetics feedback through exemplars for photographers. *Int. J. Comput. Vision 96*, 3 (Feb. 2012), 353–383. 2