PROCEEDINGS

IS&T / SPIE
Electronic
Imaging
SCIENCE AND TECHNOLOGY

The Engineering Reality of Virtual Reality 2014

Margaret Dolinsky
Ian E. McDowall
Editors

3–4 February 2014
San Francisco, California, United States

Sponsored by
IS&T—The Society for Imaging Science and Technology
SPIE

Volume 9012

IS&T
imaging.org

SPIE

PROCEEDINGS
IS&T / SPIE
Electronic Imaging
SCIENCE AND TECHNOLOGY

The Engineering Reality of Virtual Reality 2014

Margaret Dolinsky
Ian E. McDowall
Editors

3–4 February 2014
San Francisco, California, United States

Sponsored by
IS&T—The Society for Imaging Science and Technology
SPIE

Published by
SPIE

Volume 9012

Proceedings of SPIE, 0277-786X, v. 9012

The papers included in this volume were part of the technical conference cited on the cover and title page. Papers were selected and subject to review by the editors and conference program committee. Some conference presentations may not be available for publication. The papers published in these proceedings reflect the work and thoughts of the authors and are published herein as submitted. The publishers are not responsible for the validity of the information or for any outcomes resulting from reliance thereon.

Please use the following format to cite material from this book:
Author(s), "Title of Paper," in *The Engineering Reality of Virtual Reality 2014*, edited by Margaret Dolinsky, Ian E. McDowall, Proceedings of SPIE-IS&T Electronic Imaging, SPIE Vol. 9012, Article CID Number (2014)

ISSN: 0277-786X
ISBN: 9780819499295

Copublished by
SPIE
P.O. Box 10, Bellingham, Washington 98227-0010 USA
Telephone +1 360 676 3290 (Pacific Time) · Fax +1 360 647 1445
SPIE.org
and
IS&T—The Society for Imaging Science and Technology
7003 Kilworth Lane, Springfield, Virginia, 22151 USA
Telephone +1 703 642 9090 (Eastern Time) · Fax +1 703 642 9094
imaging.org

Printed in the United States of America.

Paper Numbering: Proceedings of SPIE follow an e-First publication model, with papers published first online and then in print and on CD-ROM. Papers are published as they are submitted and meet publication criteria. A unique, consistent, permanent citation identifier (CID) number is assigned to each article at the time of the first publication. Utilization of CIDs allows articles to be fully citable as soon as they are published online, and connects the same identifier to all online, print, and electronic versions of the publication. SPIE uses a six-digit CID article numbering system in which:

- The first four digits correspond to the SPIE volume number.
- The last two digits indicate publication order within the volume using a Base 36 numbering system employing both numerals and letters. These two-number sets start with 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B … 0Z, followed by 10-1Z, 20-2Z, etc.

The CID Number appears on each page of the manuscript. The complete citation is used on the first page, and an abbreviated version on subsequent pages. Numbers in the index correspond to the last two digits of the six-digit CID Number.

Conference Committee

Symposium Chair

Sergio R. Goma, Qualcomm Inc. (United States)

Symposium Co-chair

Sheila S. Hemami, Northeastern University (United States)

Conference Chairs

Margaret Dolinsky, Indiana University (United States)
Ian E. McDowall, Fakespace Labs, Inc. (United States)

Session Chairs

1 Smart Phones, Smart Immersion, and Creative Thinking
Ian E. McDowall, Fakespace Labs, Inc. (United States)

2 Seriously Speaking: Navigation, Immersion, and Architectural Design
Margaret Dolinsky, Indiana University (United States)

3 Exploring Space, HMDs, and Audiovisual Integration
Ian E. McDowall, Fakespace Labs, Inc. (United States)

4 Move!: Affectively Speaking About Immersion and Aesthetics
Margaret Dolinsky, Indiana University (United States)

Introduction

On behalf of the Engineering Reality of Virtual Reality (ERVR) program committee, welcome to the proceedings of the conference held 3-4 February 2014 in San Francisco, California, United States. ERVR is held annually as a part of IS&T/SPIE Electronic Imaging. It is the premier international imaging symposium and is at the forefront of research and innovation in digital imaging systems, 3D display, image quality, optimization, and more.

This year's conference included a panel moderated by Jaqueline Ford Morie with the panelists Brenda Laurel of UC Santa Cruz and Margaret Dolinsky Indiana University. The discussion and audience input examined the current state of virtual reality, augmented reality, online realities, and alternate realities along with related devices that are making development more accessible.

Margaret Dolinsky
Ian E. McDowall

Contents

SESSION 1 SMART PHONES, SMART IMMERSION, AND CREATIVE THINKING

SESSION 2 SERIOUSLY SPEAKING: NAVIGATION, IMMERSION, AND ARCHITECTURAL DESIGN

Interactive projection for aerial dance using depth sensing camera

Tammuz Dubnov[a], Zachary Seldess[b] and Shlomo Dubnov[b]

[a]UC Berkeley, CA, USA
[b]Calit2, UCSD, La Jolla, CA, USA

ABSTRACT

This paper describes an interactive performance system for floor and Aerial Dance that controls visual and sonic aspects of the presentation via a depth sensing camera (MS Kinect). In order to detect, measure and track free movement in space, 3 degree of freedom (3-DOF) tracking in space (on the ground and in the air) is performed using IR markers. Gesture tracking and recognition is performed using a simplified HMM model that allows robust mapping of the actor's actions to graphics and sound. Additional visual effects are achieved by segmentation of the actor body based on depth information, allowing projection of separate imagery on the performer and the backdrop. Artistic use of augmented reality performance relative to more traditional concepts of stage design and dramaturgy are discussed.

Keywords: Augmented Reality, Projection Design, RGB-D Camera, Kinect, Gesture Tracking, Aerial Dance, Circus Aerials

1. INTRODUCTION

The paper describes technology for an interactive multimedia composition for dancer or aerial performer and a computer where the performance utilizes audiovisual materials that are controlled by the movement of the performer. These movements are detected using a RGBD camera that tracks infra-red (IR) markers that are placed on dancer‘s body. The 3D information about the position of the markers in space are input into a Hidden Markov Model (HMM) that recognizes and tracks these gestures relative to similar gestures that were recorded into the system at a training phase. During the performance the dancer's movements influence the audiovisual and musical elements by controlling the speed of the playback of the audiovisual materials, as well as generating additional graphics and transforming the pre-designed video materials and sound playback in response to the dancers movements.

In the paper we describe the system's architecture and provide the technical details of the IR markers tracking method and provide evaluation of the HMM gesture recognition algorithm for a set of aerial and ground moves. In the paper we describe the strategies of mapping the gestures to graphics and discuss the considerations behind composition in an augmented reality setting. These artistic possibilities are further demonstrated in the context of a production of a new multimedia work that combines aerial dance with interactive graphics using the Kinect camera.

One of the challenges of the project is the scale, both in time and space, and the open nature of aerial dance which requires system design that supports a large range of sensing scenarios and offers a palette of creative mappings from human actions to animated projection, lighting and sound, based on the actors positions in space, their movement and gestures. Besides the technical challenge, maybe the most interesting aspect of this project is the attempt of coming up with a new language for stage, lighting and sound design that utilizes technological opportunities that transform a traditional stage into an augmented reality reactive space that allows producing novel means for artistic expression. Since this type of art combines dance movement with circus-like aerial acrobatics on apparatuses such as the silks, lyra and more, the system is required to detect, measure and track free movement in space. Segmentation of an actor's body in space can be achieved based on depth information,

Further author information: (Send correspondence to T.Dubnov)
E-mail: tdubnov@berkeley.edu, Telephone: 1 619 895 9950

The Engineering Reality of Virtual Reality 2014, edited by Margaret Dolinsky, Ian E. McDowall,
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 9012, 901202 · © 2014 SPIE-IS&T
CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2041905

which in turn allows projection of separate imagery on the performer and the backdrop. Such projection can serve different artistic roles, from augmented costume design through body mapping, to virtualization and abstraction of the dancer body immersed in a digital narrative. Other design considerations include establishing causality relations between the human actors and the virtual objects. This transforms the role of what traditionally would be considered as inanimate props into active players entailed with distinct behavior and even a character.

The work demonstrate how traditional concepts of stage design and dramaturgy have to be reconsidered in augmented reality performance situations. In terms of existing genres, the work combines aspects of scripted and choreographed mastery with a thrill of a gamer experience that often interferes with the former. Conceptual focus rather than material limits define the system. Initial results and videos of rehearsals with the system, which we named Zuzor, can be found at http://shlomodubnov.wikidot.com/kinect-aerial-dance or https://www.facebook.com/ZuzorKinect.

2. AERIAL DANCE AND AUGMENTED REALITY

Aerial Dance is a contemporary dance genre with integrated aerial apparatuses such as silks (also known as chiffon and fabrics), lyra, trapeze and more. Despite the close proximity to aerial circus, the two genres arise from separate lineages with different aesthetic approaches. Aerial circus generally consists of distinct acts, composed of a sequence of prescribed tricks with a duration of up to ten minutes, with emphasis on physicality and acrobatic skills. The moves are linked together with relatively free choreography set to music.[1] Aerial dance makes more free use of ground and suspended apparatuses, with emphasis on transitional movement and a corresponding lack of emphasis on any specific trick as well as the choreographer's intention and dance-like crafting of the work.[2] Accordingly, the commonalities and the differences in the aesthetics of the two art forms also imply different approaches to the use of Augmented Reality (AR) technology.

In circus arts there is more room for modular and improvised actions. In fact, some of the common theatrical tricks employed to increase the tension in the audience is to simulate consecutive failures of a particularly challenging move until some physical difficulty is overcome to the cheers and amazement of the crowd. Accordingly, circus like approach to AR allows more game like situations, where some of the graphics and interaction might malfunction or create a more direct and evident dialog between the technology and the human performer. In such cases, part of the movement control of effects such as sound or video is destined to comply with the failures and repeated attempts, and if the difficulty of control is evident enough it may in fact add to the overall artistic effect. In Aerial Dance, the smoothness of movement and invisibility of effort allows less room for unpredictability and uncertainty in the presentation. Accordingly, the amount of variations that AR technology allows must be limited to expressive inflections, well within the range of acceptable and expected interpretative variations for a highly stylized dance choreography.

The boundary between aerial circus and aerial dance begins to blur, as modern dance emphasizes the physicality of the performance with potential use of props and playful (even potentially intrusive) relations with the audience. In parallel as circus acts become increasingly stylized with tight integration of music and visuals it is easy to see where large overlaps between the two genres begin to take form. For the purpose of this work, the different possibilities of using the body, the aerial apparatuses and their relation to the music and visual effects had to be addressed in several possible types of interaction between the dancer and the computer. Accordingly, we distinguish between several types of interaction and AR design in terms of mapping gestures to sound and video:

- Direct Mapping: establishing the causal relations between the performer's movement and the graphic elements. Graphics include effects such as flocking, moving and floating objects, particle filter and any kind of videography element. A common movement style that we used in our artistic projects is syncing the tracking of the performer's position (in terms of the x and y coordinates) and mapping the video projection in correct relations to the performer. In our case we often mapped the performer's exact position to that of the graphic object in order to create a spotlight effect using whatever videography element we chose. Of course it is possible to create many types of mapping between the xyz coordinates of the performer and the object, such as mapping the depth of the performer (the z coordinate) to the size of the videography element, or reflecting the x coordinate to create a reflection for a duet like performance between the performer and its accompanying projection.

- Gesture Following: controlling the speed of a prepared audiovisual playback relative to a rehearsed gesture. The system provides overall precision of live versus recorded data that can be used creatively in a game like performance, as the performer consecutively attempts to perform the gesture accurately enough for the software to detect it to completion and display the corresponding full videography projection. The gesture recognition software has the capabilities to both recognize a gesture as well as to track its speed relative to the gesture recorded.
- Event Detection: setting markers along a rehearsed gesture, as well as specific detectors for abrupt changes enables the software to trigger events in specific points along the gesture. Abrupt changes in speed can be detected using the program's preprogrammed speed detection. It is possible to incorporate a graphic element that only displays when a marker surpasses a certain threshold of speed. The end of the drop can be detected by recording the drop as a gesture in a rehearsal and then placing a marker at the end of the gesture that would start the graphic effect. Another possibility is to keep track of the performer's speed and when it passes a certain threshold (which would only happen in a big drop) start the graphic display. Many artistic possibilities exists with variations on the marker placement within the recorded gestures and variations on the speed of movements choreographed into the piece.
- Actor Immersion: using depth based masking it is possible to differentiate between the performer and the backdrops. It could be desirable to emphasize the performer by projecting on the backdrop the silhouette of the performer. In our experiments we had an aerial performer in the front with a backdrop of vertical lines and performer's silhouette in horizontal lines. The actor immersion created a new setting where the image of the actor and the background video were integrated into one dynamic visual element.

3. KINECT CAMERA TRACKING

3.1 Method Description

For the description of our tracking algorithm we assume we have two pieces of information from the depth camera: an IR image and a depth image. Our method uses a right-handed coordinate system with z pointing directly out from the camera, y pointing out of the top of the camera, and x pointing to the right of the camera, when facing it.

We perform analysis on a given marker constellation in two main phases:

1. Pre-processing of IR and depth images.
2. Derivation of 3D coordinates of potential markers.

Before deriving the 3D coordinates of markers within the camera's viewport, we first need to prepare the IR and depth images. We convert the Kinect's raw depth values $r \in (0, 1027)$ into meters using the following formula[3] $d = 0.1236 \tan((r/2842.5) + 1.1863)$ where d is the distance in meters. We also convert the IR image to binary based on a user-defined brightness threshold, and then perform a single-pass morphological dilation on the binary IR image, marking a pixel ON if any of its neighbors is ON. We discuss the justification for this last process bellow.

We then perform 2D blob analysis on the dilated IR binary image to extract the bounds of all visible IR markers in the scene and use those bounds to analyze corresponding regions in the Kinect's depth image. To accurately map the IR image onto the depth image we shift the IR image by two pixels along its x axis. Before we analyze the IR binary image for blobs, however, we need to account for an idiosyncrasy in the Kinect's depth reporting behavior.

IR markers have a tendency to show up as "holes"in the Kinect's depth map. After the conversion of the IR image to binary, the bounds resulting from our blob analysis would occasionally surround pixels containing non-valid values in the depth image and hence be ignored by the tracker method. We avoid this potential problem by including the above-mentioned dilation on the image, which expands the bounds of each blob slightly, allowing the depth values surrounding the IR markers in the depth image to be included as part of the marker depth.

We then estimate the actual depth (i.e. location on the z axis) of each detected blob by calculating the average of all valid depth values in the image corresponding to the blob's bounding box in the IR image. We assume this average depth describes the depth of the blob's centroid. Given the horizontal and vertical lens angles of the IR camera, we can now derive the real-space x and y coordinates of the centroids of all found markers (blobs) as follows:

- Map the (x, y) pixel coordinates of the depth image to the range $(-1.0, 1.0)$ (left to right, bottom to top).
- Calculate the azimuth (az) and elevation (el) of each blob using the following formulas:

$$az = \arctan(blob\_pixelx(1.0/\tan(horiz\_lens\_angle))) \tag{1}$$

$$el = \arctan(blob\_pixely(1.0/\tan(vert\_lens\_angle))) \tag{2}$$

- Finally, calculate the real-space cartesian (x, y) values of each blob using the following formulas:

$$x = \tan(az)z \tag{3}$$

$$y = \tan(el)(-z) \tag{4}$$

3.2 Method Implementation

We have designed a Kinect tracking server that implements the above-described tracking method using Max/MSP/Jitter,[4] a graphical and script-based programming environment for real-time audio, graphics, and data processing. The server is set up via a simple configuration file using the JSON data-interchange format, and provides an intuitive GUI for real-time control over a variety of application parameters, depth and IR image visualization, tracker audition and debugging, and network I/O connectivity. It receives remote commands via UDP or Open Sound Control (OSC) at user-defined ports and serves the tracking results as a dynamically sized array of real-space $x/y/z$ blob centroid coordinates in meters (length equal to number of found blobs times 3) to an arbitrary number of listening clients.

Kinect IR and raw Depth images are acquired using jit.freenect.grab,[5] an open-source 3rd party object using the OpenKinect project's libfreenect library,[6] that retrieves the Kinect's RBG, IR, and depth images, and accelerometer readings, allows control over the camera's tilt motor and allows for the use of multiple Kinect cameras simultaneously. We have decided to use a libfreenect-based tool because it provides all the raw data from the Kinect that we require, and it allows our implementation to work out of the box, without driver installation requirements or other library dependencies (compared to OpenNI/NITE installation, for example).

As discussed above, we first convert the raw values of the depth image into meters. In our implementation, the user defines a range of allowable depth values to be passed to the tracker, defaulting to the Kinect's entire depth range (ca. 0.45 - 13 meters). Depth values that fall outside of this range will be set to a depth of 0.0, and ignored by the tracker. We also allow the user to optionally define up to 6 clipping planes in the 3D space described by the depth image, in order to sculpt an optimal tracking space for the environment. These clipping planes can be useful in eliminating undesirable IR and depth noise in the data, occurring as a result of IR-reflective surfaces, unused IR markers, or other IR and depth data falling within the Kinect's viewport such as other Kinect tracking regions elsewhere in the space. Each clipping plane is defined via three $x/y/z$ points describing the plane's surface. Here we transform the values of the depth image into real-space 3D coordinates using the same method as is described in Section 3.1. If a pixel in the depth image falls behind a clipping plane, its depth value will be set to 0.0, and ignored by the tracker. Otherwise, the depth value remains unchanged.

Also as discussed in Section 3.1, we perform 2D blob analysis on the dilated IR binary image in order to extract the bounds of all visible IR markers in the scene. In our implementation, blob tracking and processing is done using cv.jit,[7] an open-source 3rd party collection of tools for computer vision tasks in Max/MSP/Jitter. As tracked regions and IR markers vary in size, our implementation allows the user to define a minimum acceptable blob area (measured in pixels of the IR image) for a blob to be considered valid. Additionally, in order to place an upper limit on the CPU load of the tracker method, which increases as the blob count increases, we allow the user to define a maximum number of allowable blobs to pass to the tracking method. After the IR and Depth images are processed, we pass them to the tracker portion of the software for marker analysis, an implementation of the methods described in Section 3.1.

In order to support off-site analysis, algorithm development, and improvement, our implementation also allows the user to record the Kinect camera's raw depth and IR matrices to disk. The user has the option to import the captured data for use as an alternative live data stream, when either a Kinect camera, tracked subject, or an ideal tracking and performance environment is unavailable.

4. GESTURE RECOGNITION AND FOLLOWING

Once the 3D coordinates of the markers are available, we use the sequence of these positions as a representation of a gesture. In our implementation we use a modified Hidden Markov Model (HMM) by Bevliacqua et al.[8] that allows recognition and following of a gesture in real time based on a single example. An extensive work on Gesture-Based Communication in Human-Computer Interaction can be found in a recent book by Annelies Braffort, et al.[9] A more recent study focusing on hand gesture recognition[10] provides a taxonomy of modeling approaches that also applies to our case. According to this study, the modeling approaches can be broadly divided into temporal and spatial models, where temporal models include Kalman Filter, HMM, Neural Networks and Finite State Machine (FSM). The spatial aspects of gesture recognition is divided between 2D and 3D modeling approaches, which are further classified into template based approaches, shape features and use of markers. Moreover, geometric considerations can be included to isolate specific attributes, such as fingertips versus palm in the hand recognition task.

In our application it seems very difficult to assume an a-priory model of the dancer's body, not only due to the large variety of body shapes in ground dance movement, but also due to the free position in space when considering horizontal movement, drops and spins that are part of an aerial dance act. Accordingly, our choice of the tracking method was limited to use of markers, as explained in the previous section. Moreover, we limited the use of templates to a single recording for the training phase. In a classical HMM, there are three problems that it can solve: (1)The Recognition Problem: Given an HMM and a sequence of observations, what is the probability that the observations are generated by the model?, (2)The Tracking Problem: Given a model and a sequence of observations, what is the most likely state sequence in the model that produced the observations?, and (3)The Learning Problem: Given a model and a sequence of observations, how should we adjust the model parameters in order to maximize the probability of the observations given the model?

In our modified approach, problem (3) is only partially addressed by assuming a fixed left-to-right architecture with transition probabilities set by the user and a Gaussian observation probability with fixed variance, whose mean is set to be the coordinates of a sub-sampling of the training gesture curve. The following explanation is based on the description from Bevilacqua.[8] The HMM model fits the learned gesture by directly associating each sampled points in time to a state in a left-to-right Markov chain. The relation between a reference gesture and the HMM is shown in Figure 1. At time i the corresponding state emits a k-dimensional observable O with a probability b_i according to a normal distribution, which creates an allowed spread around the nominal gesture sample points, according to equation

$$b_i(O) = \frac{1}{\sqrt{(2\pi\sigma)^k}} exp - \frac{||O - \mu_i||^2}{2\sigma^2} \tag{5}$$

where the multi-variate covariance matrix of the sensors Σ^2 is replaced by a diagonal constant matrix equal to $\Sigma^2 = I\sigma^2$ whose values are set externally either from prior knowledge or estimated from prior experiments. Moreover, the model defines a limited number of permitted transitions by setting the transition probabilities for self and near neighboring states. Self transitions are used for remaining in a state in the case of a slower performance, moving forward is used for tracking gesture at an approximately original speed, and skipping a state occurs in case of a faster movement (these can be seen by the arrows between states in Figure 1). Skips larger then one state are not permitted, as well as backward transitions are excluded for cases when the gesture

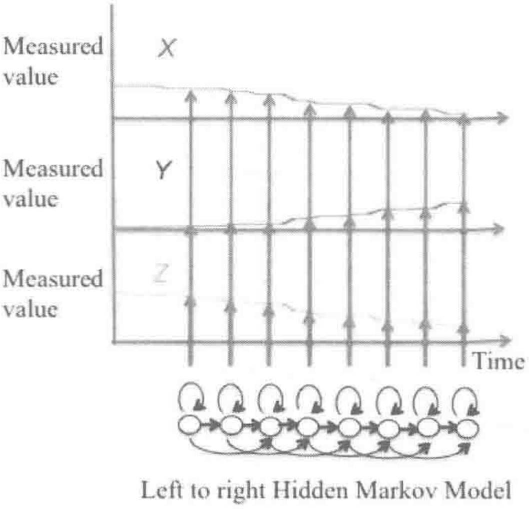

Figure 1. Learning procedure: a left-to-right HMM used to model the xyz path for a recorded gesture.

is transversed in reverse, and so on. The time progression related to the recorded gesture is solved using the HMM tracking solution that selects the state index that gives the maximal forward probability for input up to the current time. Denoted by $\alpha_t(i)$, the forward probability of HMM for state i given and input sequence of observations $O_1, O_2, ..., O_t$, the time progression is estimated either using the Viterbi algorithm or as the index $argmax(\alpha(i))$ allowing arbitrary jumps between states during tracking. Moreover, the overall likelihood of recognizing the gesture up to time t is given by

$$Likelihood(t) = \sum_{i=1}^{N} \alpha_t(i), \tag{6}$$

where N is the size of the gesture or of a gesture segment using a sliding window approach. The sliding windowing is needed in order to allow real-time computation of HMM in the case of long phrases that result in a large number of states of the HMM. The size of the window is set manually by the user.

5. SYSTEM OVERVIEW

To summarize the overall structure of the performance system, the following elements are required: Kinect Camera, Performer wearing IR markers, Kinect tracking system, MAX/MSP Mubu Gesture Follower, Quartz Composer Graphics and Video playback, Max/MSP audio playback control, Video Projection and Sound System. The communication between the KVL tracking and the Gesture Follower and QC Graphics system is established using Open Sound Control (OSC) format. Furthermore, to allow switching between gesture vocabularies and different sound and graphic designs and interactions, a simple queuing system was programmed that allowed switching between scenes by a human operator or a preprogrammed timeline. The system was mounted compactly on top of the projector that was placed in the front of the stage between the audience and the dancer.

The IR marker were attached to the performer on arms or legs (or both), according to artistic decisions that would best detect the gestures (with the IR marker often attached on the costume sleeve). In the direct mapping case, the 3D coordinates were mapped to the graphics directly with a minor resizing depending on the performance space.

In the case of a gesture mediated audiovisual interaction, the choreographer and video artist decide about the relation of the movement and visuals. The initial training phase in the production process is necessary for the program to be able to recognize the gesture in future performances. With the intended gestures recorded, the videographer can design video effects to the choreographer's discretion with the captured gesture data. Some of the videographer's choices were outlined in Section 2.

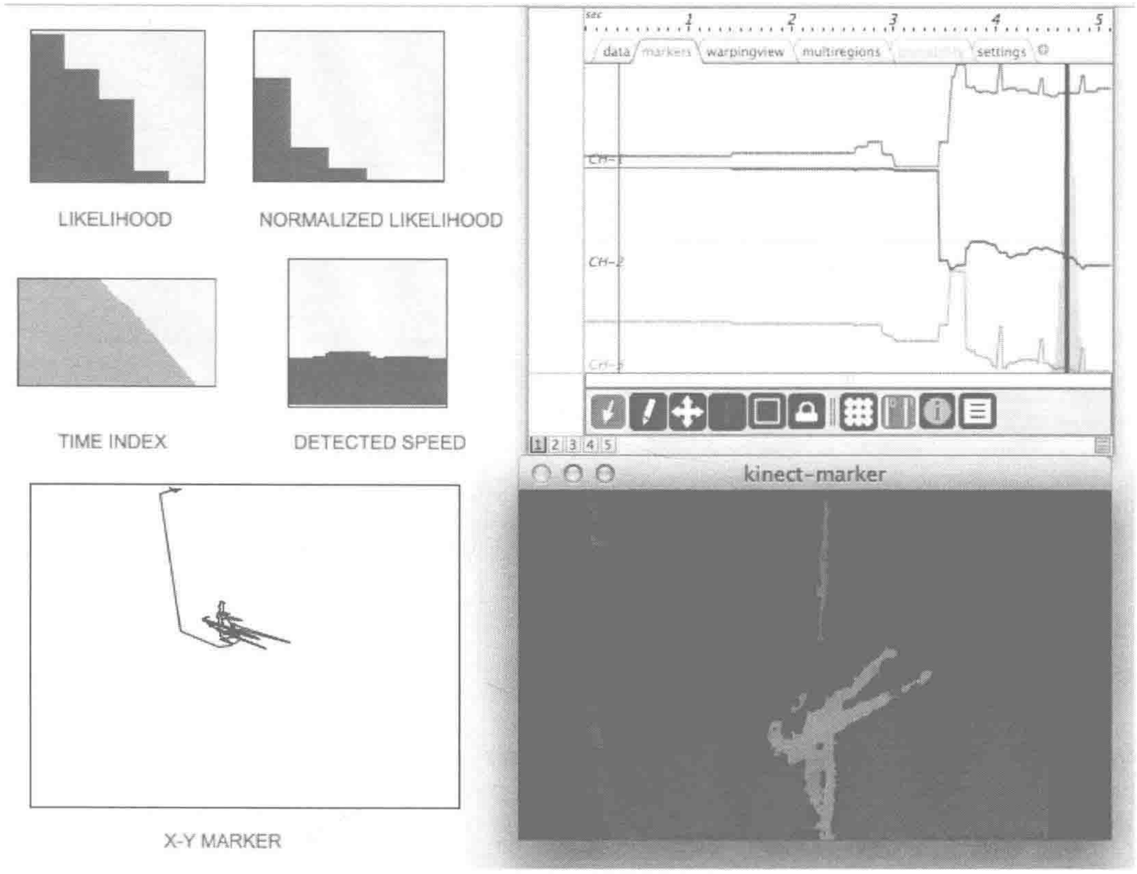

Figure 2. Screen show of the IR image, the reference gesture and the tracking and classification results that the system outputs.

When the piece is ready to be performed including the audiovisual elements interwoven into the piece, the setup for an actual performance is not complicated. With the animations and gestures ready and the IR marker present on the performer in some way, the Kinect must be placed in its intended location (for us this was always downstage center) and connected to the computer with the audiovisual designs which in turn should be connected to either a projector or a the lighting system.

6. EXPERIMENTAL EVALUATION

The proposed system was evaluated using rather complex real-life gestures taken from a dance and ground acrobatic repertoire as well as several typical sequences of movements (routines) on aerial fabric (also known as silks and chiffon). Here we describe the analysis of three gesture categories - an aerial drop, lyra mounting and spinning, and a ground back walkover. The aerial drops were further comprising of three types of drops, commonly known as Waterfall, Straddle Slide and a Star Drop. Each drop was performed by three different aerialists.

Figure 2 shows snapshot from the gesture recognition and following system based on the marker movement that was recorded by Kinect's infrared camera. The bottom right image shows the IR image with the square on the leg corresponding to blob detection of the IR marker worn by the aerialist. The graph in the center shows the coordinate profiles of the most likely reference gesture that were recognized by the system and the tracking location on the gesture timeline. The left columns shows the likelihood and the normalized likelihood for the five gesture categories, with the three leftmost bars corresponding to the likelihood of three drops in the drops category. Moreover the time warping and the estimated tempo / speed are shown. Evaluation of the system was done by recording a single gesture as a reference and testing it against the performance of similar move by the other two aerialists. In the case of the Lyra and the ground back walkover, the training and the testing gestures were recorded by the same performer in repeated trials.

Table 1. Confusion Matrix of Drops versus Spins on Lyra and Ground Back walkover

	Silk Drops	Lyra Spin	Back Walkover
Silk Drops	1.0	0	0
Lyra Spin	0	0.83	0.17
Back Walkover	0	0	1.0

Table 2. Confusion Matrix of Various Aerial Drops

	Star Drop	Waterfall Drop	Straddle Drop
Star Drop	0.66	0.17	0.17
Waterfall Drop	0.11	0.61	0.28
Straddle Drop	0	0.5	0.5

These preliminary results show that the system is capable of distinguishing between three large categories of moves: Silk Drops that undergo large changes in their y coordinate, Lyra Spins that undergoes elliptic like movement in the x coordinate and Back Walkovers that create a half circle shape in the xy plane. As Table 1 shows, these gestures were for the most part successfully recognized except for one instance in the Lyra Spin which had strong similarities to Back Walkover in its x coordinate trajectory.

We conducted another experiment between aerial drops to see if each drop could be uniquely recognized. As shown in Table 2, this was less successful due to similarities in the drops and various other limiting technical aspects, such as the different orientations of the drop per performer, as further discussed in Section 8.

7. DISCUSSION

7.1 Computer Interaction and the Arts

A large variety of research tools and results are available on the Internet about computer interaction for performance arts. For instance, the Dance and Technology Zone (DTZ)[11] maintains an annotated bibliographic source relating to dance and technology and supplies links to a wide range of publications on the issues of interactive media for production of dance and other live performances.

Professional dance groups and companies have been drawn to the possibilities of controlling computer-generated material through movement. These groups and companies utilize a variety of gesture capture paradigms to create interactive compositions. We compiled a small collection of exemplary projects on a Zuzor page,[12] with many of the examples dealing with video mapping and projection design that provide new tools for combining video art with live performance. Other resources include Digital Projection Design blog,[13] and the EyesWeb Project[14] that provides gesture capture using RGB camera for artistic and educational applications.

Other related experimental works draw their inspiration from analogue videos of the 1970s and 1980s where video synthesizers and multiplexers were used to process real-time analogue video and produce effects such as blue screening, chroma-keying, vision mixing and more. Today similar technology is ubiquitously available in software and is used by Video Jockeys (VJ) and visual multimedia performers like Robert LePage and Laurie Anderson. Software applications like Quartz Composer, Isadora, TouchDesigner or OpenEndedGroups's experimental Field platform,[15] have brought both the design and generation of real-time 3D graphics and video processing into the hands of the broad digital artistic community.

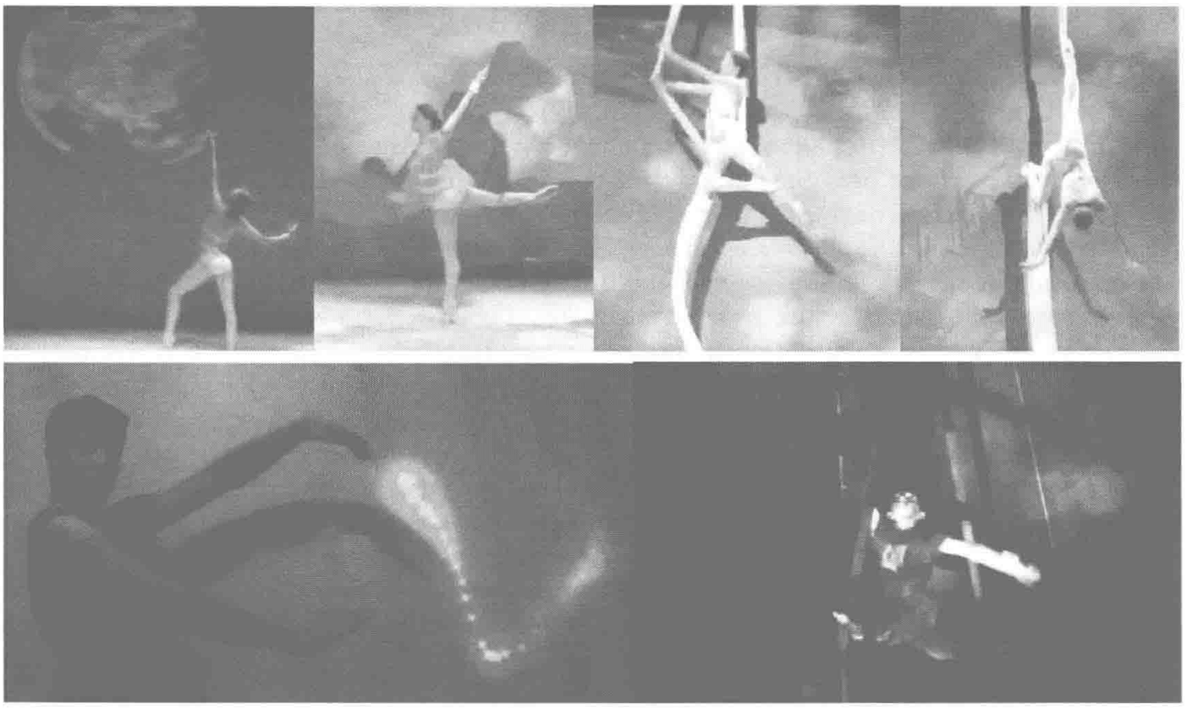

Figure 3. Excerpts from a video recording showing selected ground and aerial moves and their generated graphics. See text for more detail.

Among these explorations, a particular role is taken by technologies of Augmented Reality that deal with the combination of real world and computer generated data in real time. Introduction of totally open or improvisational situation into a professional and more traditionally trained performance disciplines creates several difficulties. For one, working with game-like science not only introduces uncertainty and requires special training by the artists, but it also complicates the technical aspect of the performance such as issues of timing and synchronization between different audiovisual elements such as music, lighting and projection, as well as queuing of scene transitions and precise timing of dramatic effects. Moreover, due to anticipated causality and close synchronization between visuals, music and the choreography, it is less significant for the audience to note that the computer generated audiovisual materials are strictly controlled by the performer, versus a more common situation where the dancers learn to synchronize themselves to music and even visual effects. Accordingly, in this project we are exploring the area between the improvisational and the composed (i.e. planned or sequenced) performance where the effect of the tracking technology is geared towards added expressiveness and flexibility in performance by altering the timing and some expressive qualities of what otherwise would be a "canned" or pre-recorded accompanying audiovisual media. Accordingly, the criteria for success of such system would be its responsiveness and ease of use that would not throw off a traditionally trained performer into totally experimental and unexpected realms.

7.2 Graphics Examples

The set of images in Figure 3 is from various performances using the system. The top set of images is showing graphic effect generated by moves of an aerial dancer in a small showing. The graphics effect include hovering an image of a Globe across the screen, fanning images of fire on the ground, generating images of fire from silks and generating flying stars while spinning on the silks. The bottom left image is a picture from a performance using the system on grounded dance, the effects used where a trailing ball of changing lights tracking the IR marker on the performers left wrist. The bottom right is from a Matrix movie themed circus act utilizing the system, where the effects used included a rotating globe made of green 0's and 1's and particle filter emitting yellow 0's and 1's tracking the IR marker on the aerialist.