Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

PRACTICAL 3D ANIMATION OF MULTIPLY ARTICULATED CONSTRUCTION EQUIPMENT

Vineet R. Kamat

Department of Civil and Environmental Engineering
University of Michigan
Ann Arbor, MI 48109-2125, U.S.A.

Julio C. Martinez

Department of Civil and Environmental Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0105, U.S.A.

ABSTRACT

This paper presents research that led to the design and implementation of practical 3D animation methods to visualize multiply-articulated construction equipment in 3D animations of discrete-event construction process models. Using principles of forward and inverse kinematics, we designed and implemented generic pieces of multiply-articulated virtual construction equipment that accept task-level instructions from external software processes. Discrete event simulation models can configure and instantiate specific pieces of such equipment and instruct them to perform common construction tasks using simple, parametric statements of text. Once instructed to perform specific tasks (e.g. Load dirt), these "smart" pieces of equipment (e.g. Backhoes) automatically decipher the sequence and amplitudes of the elemental motions their components (e.g. Boom) must undergo to accomplish those tasks. The animation methods are implemented in a software tool called KineMach that integrates as an add-on with the VITASCOPE visualization system.

1 INTRODUCTION

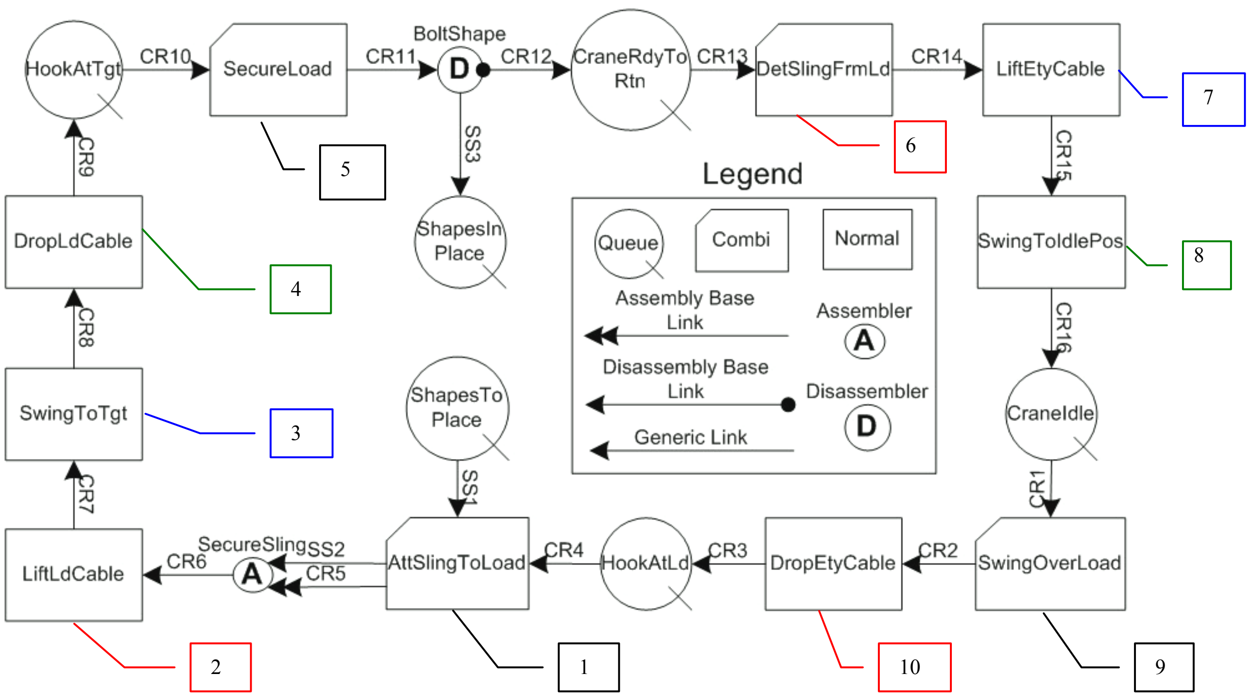
VITASCOPE is an acronym for VIsualizaTion of Simulated Construction OPErations. VITASCOPE (Kamat 2003) is a general-purpose, user-extensible 3D animation language especially suited for visualizing simulated construction operations in smooth, continuous, dynamic 3D virtual worlds. VITASCOPE animates a modeled construction process by converting discrete pieces of quantitative information communicated at uneven time instants by a running simulation model into smooth continuous motion. In order to animate a modeled operation, a simulation model is instrumented to generate a time-stamped animation trace of events that occur during a simulation run using the syntax of VITASCOPE's parametric language. VITASCOPE then post-processes the generated animation trace and depicts the modeled processes in 3D by instantiating and manipulating pre-created CAD models of involved simulation resources (e.g. equipment, materials, terrain) in a virtual environment.

VITASCOPE's parametric animation statements allow a simulation model to communicate the elemental motions involved in performing construction in a geometric transformation level syntax. Work involved in designing the VITASCOPE language identified geometric transformation-based elemental motions as the only appropriate level in the hierarchical taxonomy of construction processes at which a discrete-event simulation model can generally communicate a dynamic operation in 3D (Kamat and Martinez 2003). VITASCOPE's primary motion-describing statements (e.g. MOVE, ROTATE, SCALE) each describe a single elemental motion that a construction resource undergoes as it performs work (e.g. A truck MOVEs along a haul road, the cab of a backhoe ROTATEs as it swings, a crane's cable SCALEs as the hook is dropped or raised). A time-stamped sequence of an arbitrary number of such elemental motions describes a smooth, continuous 3D rendition of the construction operation being modeled.

1.1 Problem Description

In discrete event simulation, the state of a model changes only at discrete, but possibly random sets of simulated time points (Schriber and Brunner 2001). These time points are typically the start or end of the model's activities, and it is only then that a running discrete-event simulation model can communicate with other processes, or perform other actions such as perform output to an animation trace. The consequence of this inherent discrete-event simulation characteristic and VITASCOPE's elemental motion-describing animation vocabulary is that a simulation model intending to communicate an animation in 3D must consist of individual activities that represent each elemental motion of the resources involved in the operation being modeled. This is necessary so that time-stamped animation statements describing the involved elemental motions can be communicated at the discrete time instants at which each of the corresponding individual activities start or end during a simulation run. To illustrate this point, let us consider the example of a simulated structural steel frame erection operation using a crane (Figure 1).

The top half of Figure 1 presents a discrete-event simulation model created in STROBOSCOPE (Martinez 1996) that models the processes involved in erecting the



```
          …
          TIME 6770;
 1        ATTACH Shape65 TheHook (0,-0.5,0);
          TIME 6770;
 2        SCALE TheCable (0,-30,0) 15;
          SLIDE TheHook (0,30,0) 15;
          TIME 6785;
 3        TGTROTATE TheBoom HOR 151.93 20;
          TGTSLIDE TheTrolley (17.00,0,0) 20;
          TIME 6805;
          TGTSCALE TheCable (1,16.30,1) 15;
 4        TGTSLIDE TheHook (0,-16.30,0) 15;
          TIME 6805;
 5        ROTATE Shape65 HOR 28.07 15;
          TIME 6830;
 6        DETACH Shape65;
          PLACE Shape65 AT (0,18,0);
          HORIZORIENT Shape65 0;
          TIME 6830;
 7        TGTSCALE TheCable (1,14,1) 10;
          TGTSLIDE TheHook (0,-14,0) 10;
          TIME 6840;
 8        TGTROTATE TheBoom HOR 135 5;
          TIME 6845;
 9        TGTROTATE TheBoom HOR 180 10;
          TGTSLIDE TheTrolley (20,0,0) 10;
          TIME 6855;
10        TGTSCALE TheCable (1,34.55,1) 10;
          TGTSLIDE TheHook (0,-34.55,0) 10;
          …
```

Figure 1: Communicating a Steel Erection Operation using Elemental Geometric Transformations

steel frame in significant sub-task level detail. The lower half of the figure presents a VITASCOPE animation trace segment that when processed will depict the erection of a near column on the second tier in the frame. The numbers in the callout boxes pointing to the model's activities and the trace indicate which animation statements are communicated at the time instants at which instances of the respective activities start in a simulation run. In an animation trace that describes the construction of the entire steel frame, each of these statements will be communicated to the trace several times (depending on how many steel shapes are placed) albeit at different simulation time instants and with different numerical parameters for the statements' arguments (depending on the particular steel shape being placed).

The primary objective in using simulation to model construction processes is to evaluate and compare the outcome of alternative decisions (e.g. choosing the best construction technique and method, choosing the appropriate construction equipment, identifying the optimum allocation of labor at different levels, establishing operating logic strategies) to make the best choice. The level of detail at which a construction process is modeled depends on the specific purpose of the model and the amount and validity of the data available for the analysis. For instance, in the case of the steel erection operation presented above, an engineer may choose to model the processes at a significantly lower level of detail if s/he deems that the insights provided by the simulation results are sufficient to make the decision being contemplated and/or if detailed statistical input data is not available for the analysis. Such a possible simulation model is presented in Figure 2.

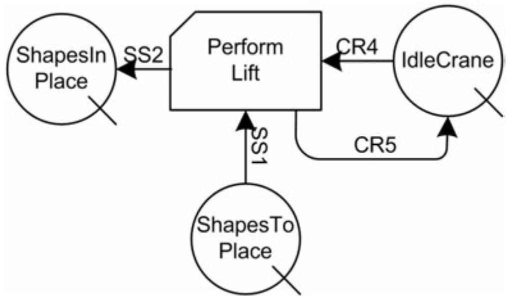


Figure 2: Modeling the Steel Erection Operation at Basic Task Level Detail

This model simulates the same steel erection operation presented above albeit at a much lower level of detail by encapsulating all the involved sub-tasks into a single large activity. A simulation model at this typical low level of detail (relatively speaking) is however unable to generate a 3D animation trace for lack of sufficient discrete time instants in a simulation run at which each of the involved tasks (e.g. Lower hook, Attach sling) could be communicated using elemental motion statements (e.g. SCALE cable, ATTACH shape). In order to be time stamped and communicated to an animation trace, each elemental motion-describing animation statement requires the occurrence of a separate discrete event during a simulation run. A simulation model thus needs separate activities for each elemental motion so that discrete events are generated when those activities start or end at which time instants the running model can time stamp and communicate the respective animation statements. In the case of the simulation model presented in Figure 2, the only discrete events that can occur during a simulation run are the start and end of the main activity "Perform Lift". The model generates no events and encapsulates no data about the sub-tasks (e.g. Lower hook, Attach sling, Swing to target) involved in performing each lift. At this typical level of modeling detail, there are not enough discrete events to generate animation statements that can describe the modeled processes in a smooth, continuous manner in 3D.

The fallout of this situation is that in order to be animated in 3D, simulation models must compulsorily be created at a significantly high-level of detail regardless of whether the detail is essential to the analysis from the decision-making point of view. In an earthmoving operation, for instance, a loading site must be modeled using separate activities for each elemental motion of the loader (Swing empty, Lower boom, Scoop dirt, Lift boom etc.) when a single activity (e.g. Load Dirt) in most cases is sufficient to model a loader otherwise (Kamat and Martinez 2001). Simulation models are often created at high levels of abstraction where each building block (i.e. activity) in the model represents a basic construction task (e.g. Load Dirt) and not elemental motions (Swing empty, Lower boom, Scoop dirt, Lift boom etc.).

2 RESEARCH INITIATIVE

In attempting to address this issue, we found promise in a technology based on the concept of Inverse Kinematics that is widely used for controlling industrial robotic manipulators. Engineers who design and implement computer programs to control robots used in the manufacturing industry are concerned with moving a robot's gripper (i.e. the tool at the end of the robot's arm) between specific desired positions and orientations (Korein and Badler 1982). A robotic manipulator is an articulated structure composed of a kinematic chain of linkages. The joints in a robot's arm are controlled by servomotors that change the length of its various links or the orientation between them so that each link (particularly the end tool) is positioned as required defining a specific articulation for a particular task. Robotics engineers can greatly benefit by being able to simply say where a particular robot's tool must work and have some procedure compute how each link and joint in the robot's arm is to be arranged to achieve that goal (Madhavapeddy and Ferguson 1998). To study that problem, researchers in robotics have devoted a significant amount of work in investigating the problem kinematically.

2.1 Kinematics

Kinematics studies the geometric properties of the motion of points without regard to their masses or to the forces acting on them (McCarthy 1990). A set of points in which the distance between any pair never varies is called a rigid body or a rigid link. The position of such a rigid link in Euclidian space is fully qualified by six dimensions; three translations (i.e. the X, Y, and Z co-ordinates) and three rotations (i.e. the Roll, Pitch, and Yaw). A set of links connected by joints that constrain their relative movement is called a kinematic chain. Common industrial robots that are comprised of a set of links with one end attached to a rigid base are defined as open kinematic chains (Craig 1989). A closed kinematic chain (also called a mechanism), on the other hand, may be attached to a rigid base in more than one place.

2.1.1 Forward Kinematics

A kinematic problem can be studied and analyzed in one of two ways. The case where all joint angles of a particular chain articulation are manually specified is termed Forward Kinematics (FK) (Watt and Watt 1992). In FK, the final shape of the articulated chain is dependent on the angles explicitly specified for each of its links. This can be mathematically stated as $X = f(\Theta)$, where X is the resultant position and orientation vector of the free end of the chain (called the end effector) and $\Theta = (\theta_1, \theta_2, ..., \theta n)$ is the state vector that describes the configuration of the chain by encapsulating the position and orientation vectors of each constituent link. In FK, the configuration of all link joints is specified explicitly. The configuration of the end effector is determined indirectly by descending the tree of the structure and accumulating all leading link transformations to arrive at the only possible solution for the end effector.

2.1.2 Inverse Kinematics

Inverse Kinematics (IK), on the other hand, is the reverse of FK and is of significantly more interest to robotics engineers. Given the desired position and orientation of the end effector in a kinematic chain, IK is concerned with computing what angles and/or lengths each of the link joints need to be in to achieve that target (Zhao and Badler 1994). Using the same terminology described above, the IK problem can be mathematically stated as $\Theta = f^{-1}(X)$. Due to its non-linear nature, IK is significantly more complex than FK because there are usually multiple or infinitely many solutions to the problem especially when the number of links in a chain is large. In many cases, there are no feasible solutions at all when it is impossible for the linked chain to reach the target. This occurs when the sum of the lengths of a chain's links is less than the distance between the target position and the rigid base of the chain (called the terminator).

2.2 Applicability of IK Techniques

VITASCOPE's methods of animating simulated construction processes are based on FK technologies. At the start of each modeled elemental task (e.g. Lower boom), a simulation model must explicitly communicate the elemental motion (i.e. the target position and/or orientation) that a component (e.g. Boom) on the corresponding virtual piece of equipment (e.g. Backhoe) must undergo to visually depict the elemental task. Therefore, in cases where simulation models have no separate activities to represent elemental tasks and have no spatial information about the corresponding virtual equipment's geometry, it is impossible to communicate a 3D animation trace.

In order to animate construction processes modeled at higher levels of abstraction (i.e. basic tasks), we need animation methods that accept instructions at the basic task level and then automatically figure out the elemental motions of the corresponding virtual pieces of equipment to visually depict those tasks. In other words, we need animation methods that can allow a simulation model to simply say (i.e. communicate) what basic task (e.g. Load dirt, Perform lift) a particular virtual piece of equipment must perform and have some procedures automatically compute how each link (i.e. component) in the equipment's kinematic chain is to be arranged at each pose (i.e. articulation) necessary to accomplish that task.

IK is directly relevant to our problem and presents an ingenious technique of addressing it. For instance, by simply knowing where in the virtual world to dig (i.e. the target position), an IK procedure can compute the relative angles at which a virtual backhoe's components (i.e. Cab, Boom, Stick, Bucket) should be such that the virtual bucket (the end effector in the kinematic chain) reaches the target position. After automatically computing the specifics of a particular required articulation, existing FK based animation methods can be invoked to explicitly move the virtual equipment's components to that articulation.

3 TECHNICAL APPROACH

IK is a goal-directed process. When an end-effector is assigned a desired goal, it must travel towards the goal (assuming it is reachable) through a series of one or more iterative steps. The adopted IK procedure is then responsible for computing the transformations of the remaining kinematic chain joints. Most pieces of articulated construction equipment can be viewed as open kinematic chains comprised of a finite number of links that are joined together with simple prismatic or revolute joints, each of which typically allows only one degree of freedom to the next link down the hierarchy.

A prismatic joint is a sliding connection between two links that exhibit one or more degrees of translational freedom (i.e. the child link can slide with respect to the parent link in one or more dimensions). An automobile shock absorber, for instance, is a simple prismatic joint exhibiting only one degree of translational freedom. A revolute joint between two links, on the other hand, allows a child link to rotate around one or more axes with respect to its parent link. A door hinge, for instance, is a simple revolute joint that describes only one degree of rotational freedom.

The process of decomposing and viewing pieces of construction equipment as kinematic chains is straightforward. A backhoe, for instance, can be described as a kinematic chain where pairs of components (i.e. cabin and the crawlers, boom and cabin, stick and boom, bucket and stick) are joined together by simple revolute joints, each describing only one degree of rotational freedom. When the crawlers are stationary, a backhoe's links can be analyzed as an open kinematic chain i.e. the crawlers (the base link) can be assumed to be attached to a fixed base (the earth in this case). The leading edge of the backhoe's bucket is the end effector of the kinematic chain that travels to certain points where the excavator digs (or dumps).

Components in a tower crane can similarly be viewed as a collection of links joined together by revolute as well as prismatic joints. While the connection between the crane's swinging boom and the tower is simple revolute, the joint between the boom's arm and the trolley that suspends the crane's hook is a simple prismatic joint. In addition, even though the crane's hook is freely suspended from the trolley with cables, it can (for IK purposes) be viewed as having been joined to the trolley using another simple prismatic joint. In a crane's case, the hook is the end effector while the fixed tower base is the terminator.

In this work, we adopted a general, computationally efficient, and relatively easy-to-implement IK algorithm based on non-linear optimization. This algorithm (called the Cyclic-Coordinate Descent) attempts to minimize the distance between a specified target and the end effector of the specified kinematic chain by iteratively adjusting the chain's joints in a way that minimizes the distance.

3.1 Cyclic-Coordinate Descent Method

The Cyclic Coordinate Descent (CCD) method is an iterative heuristic search technique which attempts to minimize position and orientation errors between an end effector and its desired position by varying one joint variable at a time (Wang and Chen 1991). Each iteration involves a single traversal of the kinematic chain from the most distal link inward towards the chain terminator. Each joint variable is modified in turn to minimize an objective function. As a solution is obtained at each joint, the position and the orientation of the end effector are immediately updated to reflect the change. Thus, the minimization problem solved at any particular joint in the chain incorporates the changes made to more distal joints during iteration.

The mechanics of the CCD IK method are best explained with a practical example which we present next. The mathematical formulation of the IK problem using the CCD non-linear optimization method, however, makes some interesting reading. We refer the mathematically inquisitive reader to Welman (1993) who presents a concise mathematical formulation of the minimization problem encountered in the CCD IK method.

3.1.1 Basic Computation Scheme

Consider a virtual hydraulic backhoe excavating dirt from a pit. A cross-sectional side view of such a configuration is presented at the top of Figure 3. Let us assume that the point marked "X" is the location where the virtual backhoe must "dig" in its next pass. For simplicity in introducing the algorithm, let us also assume that the backhoe's cabin is already aligned to dig at point "X". In other words, as the lower half of Figure 3 presents, point "X" lies on the straight line that connects the backhoe's boom, stick, and bucket when viewed from the top. In order to dig at point "X" then, only the backhoe's boom, stick, and/or bucket need be manipulated. The goal of the CCD IK algorithm, then, is to minimize the distance between the leading edge of the bucket and point "X" by rotating the virtual backhoe's bucket, stick, and boom iteratively.

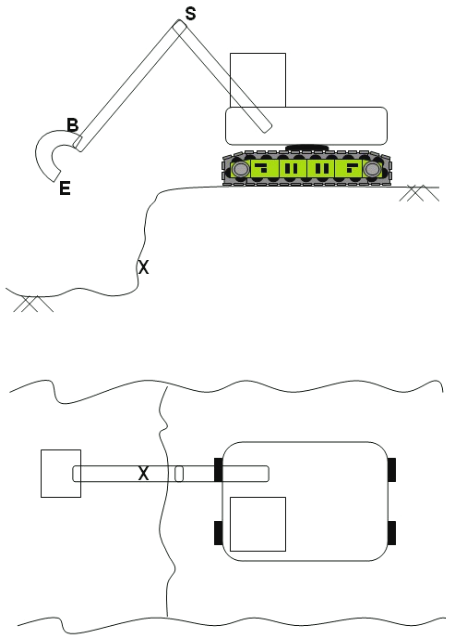


Figure 3: Configuration of a Backhoe's IK Problem

In order to solve this minimization problem using the CCD algorithm, we start with the last link in the kinematic chain (i.e. the bucket itself). First, we create a vector from the root of the in-context link (B) to the current end effector position (E). We then create another vector from (B) to the desired target position (X). This is graphically presented in Figure 4 (a). In this iteration, the aim of the algorithm is to compute the angle (A) by which the vector (BE) must be rotated in order to coincide with vector (BX).

The angle (A) can be computed using a standard result from elementary vector algebra. In particular, the dot product for the two vectors (BE and BX) can be defined by $BE \bullet BX = |BE||BX|\cos A$. By computing the inverse cosine of the vector dot product, we obtain the magnitude of the angle (A) between the vectors.

In order to coincide vector (BE) with (BX), however, we need another piece of information in addition to the magnitude of angle (A). In particular, we need to know the direction of rotation about point (B). We compute that by using another elemental result from vector algebra. The cross product of the two vectors (BE and BX) is a third vector that is always perpendicular to both (BE) and (BX). The cross product vector thus represents the axis in 3D space about which to rotate the source vector (BE). Knowing the amount of rotation and the direction in which to apply it, we modify the in-context link. This is presented graphically in Figure 4 (b).

The algorithm then moves one link up the kinematic chain (the backhoe's stick in this case) and repeats the procedure. The result of applying this next iteration at joint (S) is presented graphically in Figures 4 (c) and (d). This procedure continues up the chain until the base link in the chain is reached, after which the whole process is repeated, starting at the last link again. The iterations continue until either the end effector (E) is close enough to the desired position (X) or a specified number of iterations have been performed.

Specifying a limit for the maximum number of iterations is necessary to stop the computation in cases where the target position is unreachable. Note that in this particular configuration, the algorithm will not attempt to rotate the cabin as it is already assumed to be aligned with the target position. In addition, the crawlers represent the fixed terminator link in the chain and are not manipulated during the iterations. In cases when the kinematic chain consists of prismatic joints (e.g. in a virtual tower crane), the algorithm proceeds in a very similar way except that a link is attempted to be translated (and not rotated) to coincide the end-effector with the required target position.

4 KINEMACH

Our implementation of IK techniques to animate simulated construction processes is a powerful tool that allows discrete event simulation models created at typical high levels of abstraction to communicate a smooth, continuous opera-

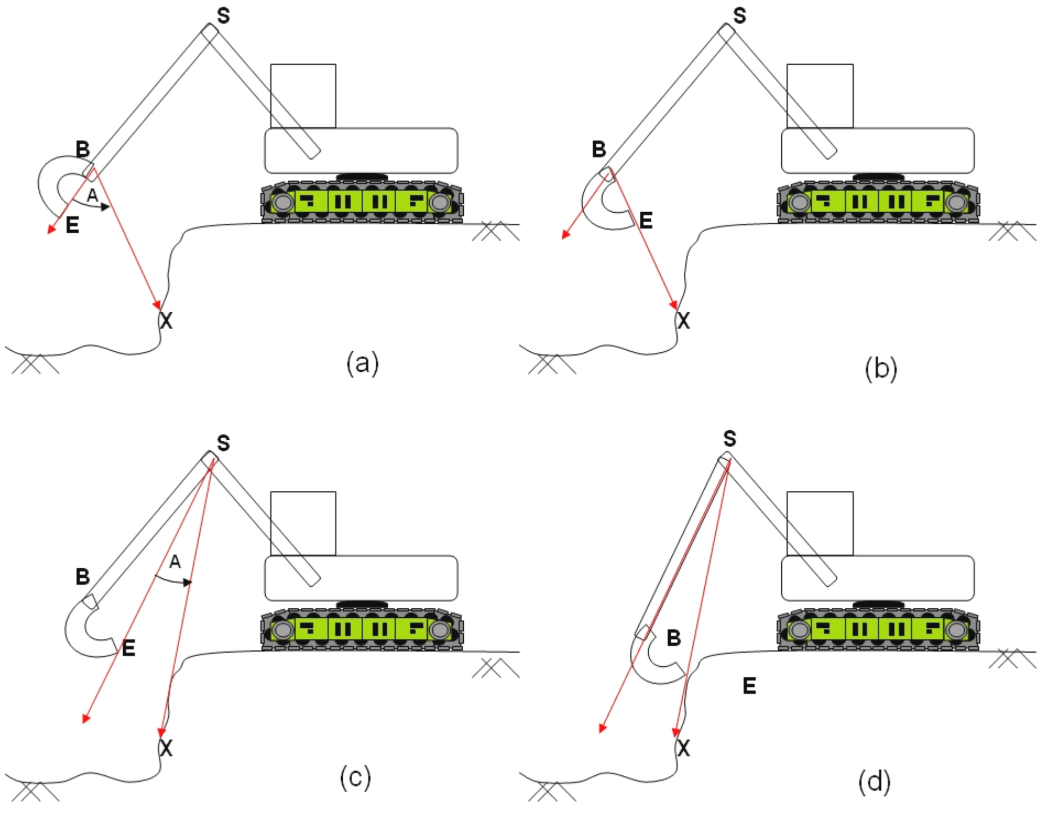


Figure 4: Cyclic Coordinate Descent Iterations

tion in 3D using a correspondingly high, construction task-level vocabulary. This tool, called KineMach, is implemented as an extension (add-on) to VITASCOPE.

4.1 Parametric Animation Methods

KineMach implements "smart", generic pieces of virtual construction equipment and provides simple parametric text statements that can be used to issue task-level instructions to that equipment to visually depict the performance of construction work. Currently implemented generic pieces of equipment include a tower crane, a crawler mounted lattice boom crane, a crawler mounted backhoe, and a highway dump truck. These virtual pieces of equipment are generic in that more than one piece of a particular equipment (having different dimensions, if necessary) can be instantiated and independently manipulated on the same virtual construction site.

Tables 1 and 2 present some of the parametric statements KineMach implements. In particular, only statements used to instantiate and manipulate virtual crawler mounted backhoes (Table 1) and crawler mounted lattice boom cranes (Table 2) are presented. KineMach statements that implement generic tower cranes and dump trucks are similar in spirit and are excluded from this paper due to space constraints. The parametric statements KineMach presents embody a construction work-like terminology making them very readable. Their meaning and significance should be self-explanatory.

KineMach's statements are designed to represent the common construction tasks that real pieces of construction equipment perform on real construction sites. Many statements are designed using standard, documented, and commonly used terminology. For instance, in the case of the implemented cranes, most KineMach statements have a direct one-to-one correspondence with standard crane hand signals used in real crane operations. Few other statements (e.g. PutThatThere) represent an improvised (albeit equally readable) terminology that we expected would be of particular utility in addressing the current problem.

4.2 Animation Schema

KineMach's intention is to be able to animate modeled construction processes in 3D at the same smooth, continuous level detail afforded by existing animation methods. The merit behind KineMach's design, however, is that the same level of detailed animation is achieved with significantly lower operational details being communicated by authoring simulation models themselves. KineMach thus essentially describes an additional layer of abstraction in the interface between discrete event simulation models and existing methods of animating operations in 3D. This layer of abstraction automatically generates the elemental, operation-level details that would otherwise have to be communicated explicitly by simulation models.

Figure 6 presents the relation between discrete event simulation models, existing methods of animating simulated processes, and KineMach. A running simulation model, which would otherwise have to communicate elemental motion-level animation instructions, now communicates with KineMach in a construction task-level vocabulary. KineMach deciphers the instructions and uses IK to compute the elemental motions involved in performing the tasks. KineMach then invokes existing elemental motion animation methods to depict the performance of the reconstruction tasks in 3D. We return to the crane example presented earlier to further elucidate this relationship.

A discrete event simulation model created using high level construction tasks as building blocks (e.g. Figure 2) can now communicate instances of those tasks to KineMach in the pertinent piece of equipment's vocabulary. For instance, each time a steel shape is placed (i.e. the activity "Perform Lift" takes place), the simulation model communicates a KineMach statement [e.g. Crane1.PutThatThere Shape65 (0,18,0) 60;] to the animation trace requesting the instantiated virtual crane to perform the task.

When KineMach processes each such statement, it first computes the elemental crane motions necessary to accomplish the task and apportions the total task time (communicated by the simulation model) to the individual elemental motions using proportions determined empirically. KineMach then generates elementary motion-describing animation statements in the VITASCOPE language and forwards them to the visualization engine to graphically depict the operation. Other pieces of KineMach implemented equipment work in much the same way.

5 FUTURE WORK

A obvious extension to KineMach is the design of animation methods to instantiate and manipulate other types of

Table 1: KineMach Implemented Backhoe Statements

Statement	Usage
BACKHOE [HoeName] [CrawlerWidth];	Instantiate, name, and size a virtual backhoe
[HoeName].DigAt [Source] [Time];	Instruct a backhoe to dig at a specified source in given time
[HoeName].DumpAt [Sink] [Time];	Instruct a backhoe to dump at a specified sink in given time
[HoeName].DigHereDumpThere [Source] [Sink] [Time];	Instruct a backhoe to perform a loading pass in given time
[HoeName].Travel/BackUp [Trajectory] [Time];	Instruct a backhoe to travel forward or reverse on a specified trajectory in given time

Table 2: KineMach Implemented Lattice Boom Crane Statements

Statement	Usage
CRAWLERCRANE [CraneName] [CrawlerWidth] [BoomLength];	Instantiate, name, and size components of a virtual crawler-mounted crane
[CraneName].HookAt [TargetPosition] [Time];	Instruct an instantiated crane to move its hook to a specified position in a given time
[CraneName].PutThatThere [ObjName] [TargetPosition] [Time];	Instruct an instantiated crane to perform a full lift in the given operating time
[CraneName].RaiseBoom/ LowerBoom [Amount] [Time];	Instruct an instantiated crane to raise or lower its boom by a specified amount in a given time
[CraneName].Travel/BackUp [Trajectory] [Time];	Instruct an instantiated crane to travel forward or backward on a specified trajectory in a given time
[CraneName].SwingBy/ SwingToward [Amount] [Time];	Instruct an instantiated to swing by a specified amount or towards a specific object in a given time
[CraneName].HoistBy/LowerBy [Amount] [Time];	Instruct an instantiated to hoist or lower its hook by a specified amount in a given time

equipment in a high level, task-based language. KineMach and the underlying VITASCOPE animation language are both designed to be end-user extensible. Animation methods to manipulate other virtual pieces of construction equipment (e.g. dozers, scrapers, front-end loaders, articulated trucks, draglines etc.) can thus be designed and seamlessly integrated into KineMach's existing vocabulary.

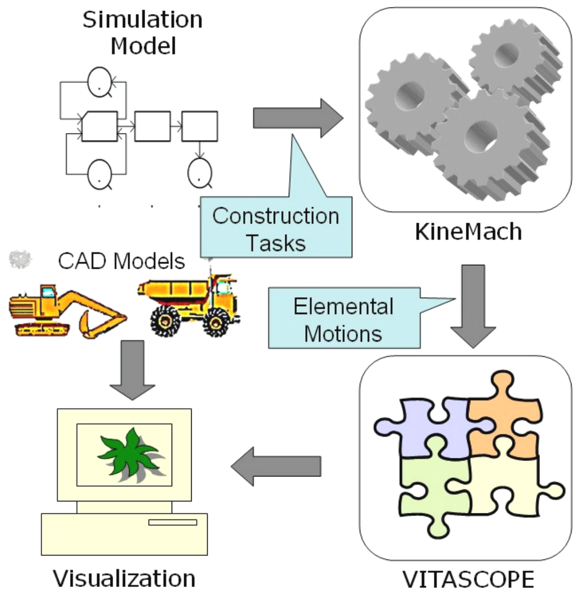


Figure 6: KineMach Animation Schema

Another area of future research is incorporating dynamics into KineMach's computations. The current implementation only analyzes the motions of the equipment pieces kinematically without regard to the forces that cause those motions. A KineMach-instantiated crane, for instance, would never tip over if it is overloaded. In fact, such computation is not considered at all. Introducing the notion of forces and loads into the computation process would cause KineMach's equipment pieces to not only be kinematically intelligent, but dynamically "smart" as well.

6 CONCLUSION

The design of 3D animation methods to visualize simulated construction processes modeled and communicated at high, task level detail requires three key technologies:

1. Simple, parametric text methods (i.e. statements) with semantics rich enough to succinctly represent the description of common construction tasks.
2. Automated dissection of the communicated construction tasks into constituent elemental motions of relevant virtual equipment pieces.
3. Generation of spatial, geometric information to compute the amplitudes and directions of the deciphered elemental motions to visually depict the performance of construction work.

Designing statements to instantiate generic pieces of construction equipment that accept pertinent task-level instructions (in parametric text statements) provides a logical basis of generally organizing 3D animation methods intended to communicate construction in a higher level, domain specific terminology. In addition, the application of IK techniques provides an ingenious framework to automatically compute the elemental motions articulated machines undergo as they perform specific construction tasks. Furthermore, IK techniques (by their very nature) automatically generate the geometric transformations necessary to describe the computed elemental motions thus making the explicit communication of detailed, spatial, geometric information unnecessary.

The designed 3D animation methods allow engineers to visualize discrete event simulation models created at typical, higher, task-level detail. In addition, they relieve engineers from manually encapsulating unnecessary (from simulation point of view) detailed geometric specifics into simulation models. By greatly simplifying the process of animating modeled operations, this work puts in place the computing infrastructure to make simulation-driven construction operations visualization more straightforward, accessible, and practical.

ACKNOWLEDGMENTS

The authors thank the National Science Foundation (NSF) CAREER and ITR programs for supporting the presented work. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

Craig, J. J. 1989. Introduction to Robotics: Mechanics and Control, 2nd Ed., Reading, MA: Addison-Wesley.

Kamat, V. R. 2003. VITASCOPE: Extensible and Scalable 3D Visualization of Simulated Construction Operations. Doctoral dissertation, Department of Civil and Environmental Engineering, Virginia Tech, Blacksburg, VA. Available online via `<http://scholar.lib.vt.edu/theses/available/etd-04142003-222138/>` [accessed April 9, 2004].

Kamat, V. R., and J. C. Martinez. 2001. Visualizing Simulated Construction Operations in 3D. Journal of Computing in Civil Engineering 15 (4): 329-337.

Kamat, V. R., and J. C. Martinez. 2003. Automated Generation of Dynamic, Operations Level Virtual Construction Scenarios. Electronic Journal of Information Technology in Construction (ITcon). 8: 65-84. Available online via `<www.itcon.org>` [accessed April 9, 2004].

Korein, J. U., and N. I. Badler. 1982. Techniques for Generating the Goal-Directed Motion of Articulated Structures. IEEE Computer Graphics and Applications 2 (9): 71-81.

Madhavapeddy, N., and S. Ferguson. 1998. Specialized Constraints for an Inverse Kinematics Animation System Applied to Articulated Figures. In Proceedings of Eurographics 98, 215-223. Geneva, Switzerland: European Association for Computer Graphics.

Martinez, J.C. 1996. STROBOSCOPE: State and Resource Based Simulation of Construction Processes. PhD Dissertation, Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, Michigan.

McCarthy, J. M. 1990. Introduction to Theoretical Kinematics, Cambridge, Massachusetts: MIT Press.

Schriber, T. J., and D. T. Brunner. 2001. Inside Discrete-Event Simulation Software: How it Works and Why it Matters. In Proceedings of the 2001 Winter Simulation Conference, ed. S Andradttir, K.J. Healy, 158 -168. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Wang, L.C.T., and C.C. Chen. 1991. A Combined Optimization Method for Solving the Inverse Kinematics Problem of Mechanical Manipulators. IEEE Transactions on Robotics and Automation 7 (4): 489-499.

Watt, A., and M. Watt. 1992. Advanced Animation and Rendering Techniques - Theory and Practice, New York, New York: ACM Press - Addison Wesley.

Welman, C. 1993. Inverse Kinematics and Geometric Constraints for Articulated Figure Manipulation, MS thesis, Simon Fraser University, Burnaby, British Columbia, Canada.

Zhao, J., and N.I. Badler. 1994. Inverse Kinematics Positioning Using Nonlinear Programming for Highly Articulated Figures. ACM Transactions on Graphics 13 (4): 313-316.

AUTHOR BIOGRAPHIES

VINEET R. KAMAT is an Assistant Professor in the Department of Civil and Environmental Engineering at the University of Michigan. He received a PhD in Civil Engineering at Virginia Tech in 2003; a MS in Civil Engineering at Virginia Tech in 2000; and a BE degree in Civil Engineering from Goa University (Goa, India) in 1998. He designed and implemented the VITASCOPE visualization system with J. Martinez as part of his doctoral research. In addition to visualization, his research interests include discrete event simulation, information technology, and decision support systems for construction engineering. His email and web addresses are `<vkamat@umich.edu>` and `<http://www.engin.umich.edu/~vkamat>`.

JULIO C. MARTINEZ is an Associate Professor at Virginia Tech. He received his PhD in Civil Engineering at the University of Michigan in 1996; a MSE in Construction Engineering and Management from the University of Michigan in 1993; a M.S. in Civil Engineering from the University of Nebraska in 1987; and a Civil Engineer's degree from Universidad Catolica Madre y Maestra (Santiago, Dominican Republic) in 1986. He designed and implemented the STROBOSCOPE simulation language with P. Ioannou and was V. Kamat's research advisor. In addition to discrete event simulation, his research interests include construction process modeling and decision support systems for construction engineering. His email address is `<julio@vt.edu>` and his website may be accessed at `<http://strobos.ce.vt.edu>`.