DOWNLOAD PDF AUTOMATIC RIGGING AND ANIMATION OF 3D CHARACTERS

Chapter 1 : Automatic Rigging and Animation of 3D Characters

Automatic Rigging and Animation of 3D Characters Ilya Baran & Jovan PopoviÄ‡ Figure 1: Our method takes a static character mesh and an input skeleton and rigs the character so that it can be animated by controlling the skeleton.

Comment rigging to specify its internal skeletal structure and to define how the input For example, Autodesk Maya 7 assigns A Tutorial on Support Vector Machines for. We present a method for animating characters automatically. Given a static character mesh and a generic skeleton, our method adapts the skeleton to the character and attaches it to the surface, allowing skeletal motion data to animate the character. Because a single skeleton can be used with a wide range of characters, our method, in conjunction with a library of motions for a few skeletons, enables a user-friendly animation system for novices and children. Our prototype implementation, called Pinocchio, typically takes under a minute to rig a character on a modern midrange PC. Animation, Deformations, Geometric Modeling 1 Figure 1: The automatic rigging method presented in this paper allowed us to implement an easy-to-use animation system, which we called Pinocchio. In this example, the triangle mesh of a jolly cartoon character is brought to life by embedding a skeleton inside it and applying a walking motion to the initially static shape. Introduction Modeling in 3D is becoming much easier than before. User-friendly systems such as Teddy [Igarashi et al. Bringing these static shapes to life, however, is still not easy. In a conventional skeletal animation package, the user must rig the character manually. This requires placing the skeleton joints inside the character and specifying which parts of the surface are attached to which bone. The tedium of this process makes simple character animation more difficult than it could be. We envision a system that eliminates this tedium to make animation more accessible for children, educators, researchers, and other non-expert animators. To support this functionality, we need a method as shown in Figure 1 that takes a character, a skeleton, and a motion of that skeleton as input, and outputs the moving character. The missing portion is the rigging: Our algorithm consists of two main steps: Our design decisions relied on three criteria, which we also used to evaluate our system: A single skeleton is applicable to a wide variety of characters: The resulting animation quality is comparable to that of modern video games. The automatic rigging usually takes under one minute on an everyday PC. For this, we designed a maximum-margin supervised learning method to combine a set of hand-constructed penalty functions. To ensure an honest evaluation and avoid overfitting, we tested our algorithm on 16 characters that we did not see or use during development. Our algorithm computed a good rig for all but 3 of these characters. For each of the remaining cases, one joint placement hint corrected the problem. We simplify the problem by making the following assumptions. The character mesh must be the boundary of a connected volume. Lastly, the character must be proportioned roughly like the given skeleton. We introduce several new techniques to solve the automatic rigging problem: Animating user-provided data by fitting a template has been successful in cases when the model is fairly similar to the template. Most of the work has been focused on human models, making use of human anatomy specifics, e. For segmenting and animating simple 3D models of characters and inanimate objects, Anderson et al. Almost any system for mesh deformation whether surface based [Lipman et al. Teichmann and Teller [] propose a spring-based method. Unfortunately, at present, these methods are unsuitable for real-time animation of even moderate size meshes. Because of its simplicity and efficiency and simple GPU implementation , and despite its quality shortcomings, linear blend skinning LBS , also known as skeleton subspace deformation, remains the most popular method used in practice. Most real-time skinning work, e. However, such techniques are unsuitable for our problem because we only have a single mesh. Instead, we must infer articulation by using the given skeleton as an encoding of the likely modes of deformation, not just as an animation control structure. To our knowledge, the problem of finding bone weights for LBS from a single mesh and a skeleton has not been sufficiently addressed in the literature. Previous methods are either mesh resolution dependent [Katz and Tal] or the weights do not vary smoothly along the surface [Wade], causing artifacts on highresolution meshes. Some commercial packages use

proprietary methods to assign default weights. For example, Autodesk Maya 7 assigns weights based solely on the vertex proximity to the bone, ignoring the mesh structure, which results in serious artifacts when the mesh intersects the Voronoi diagram faces between logically distant bones. This method could also be useful in existing 3D packages. Our prototype system, called Pinocchio, rigs the given character using our algorithm. It then transfers a motion to the character using online motion retargetting [Choi and Ko] to eliminate footskate by constraining the feet trajectories of the character to the feet trajectories of the given motion. Recent exceptions include Motion Doodles [Thorne et al. These approaches focus on simplifying animation control, rather than simplifying the definition of the articulation of the character. In particular, a spatial keyframing system expects an articulated character as input, and as-rigid-as-possible shape manipulation, besides being 2D, relies on the constraints to provide articulation information. The Motion Doodles system has the ability to infer the articulation of a 2D character, but their approach relies on very strong assumptions about how the character is presented. Character Animation Although most skeleton-based prior work on automatic rigging focused on skeleton extraction, for our problem, we advocate skeleton embedding. A few approaches to the skeleton extraction problem are representative. Teichmann and Teller [] extract a skeleton by simplifying the Voronoi skeleton with a small amount of user assistance. In their paper, Katz and Tal [] describe a surface partitioning algorithm and suggest skeleton extraction as an application. The technique in Wade [] is most similar to our own: For the purpose of automatically animating a character, however, skeleton embedding is much more suitable than extraction. For example, the user may have motion data for a quadruped skeleton, but for a complicated quadruped character, the extracted skeleton is likely to have a different topology. The anatomically appropriate skeleton generation by Wade [] ameliorates this problem by techniques such as identifying appendages and fitting appendage templates, but the overall topology of the resulting skeleton may still vary. For example, for the character in Figure 1, ears may be mistaken for arms. Another advantage of embedding over extraction is that the given skeleton provides information about the expected structure of the character, which may be difficult to obtain from just the geometry. So although we could use an existing skeleton extraction algorithm and embed our skeleton into the extracted one, the results would likely be undesirable. For example, Skeleton Extraction 3 Skeleton Embedding Skeleton embedding resizes and positions the given skeleton to fit inside the character. This can be formulated as an optimization problem: Solving such a problem directly using continuous optimization is infeasible. Pinocchio therefore discretizes the problem by constructing a graph whose vertices represent potential joint positions and whose edges are potential bone segments. This is challenging because the graph must have few vertices and edges, and yet capture all potential bone paths within the character. The graph is constructed by packing spheres centered on the approximate medial surface into the character and by connecting sphere centers with graph edges. Pinocchio then finds the optimal embedding of the skeleton into this graph with respect to a discrete penalty function. It uses the discrete solution as a starting point for continuous optimization. To help with optimization, the given skeleton can have a little extra information in the form of joint attributes: Approximate Medial Surface Figure 3: Packed Spheres Figure 4: These attributes are specific to the skeleton but are independent of the character shape and do not reduce the generality of the skeletons. The original and reduced quadruped skeleton The final discretization step constructs the edges of the graph by connecting some pairs of sphere centers Figure 4. Pinocchio adds an edge between two sphere centers if the spheres intersect. The precise condition Pinocchio uses is that the distance from any point of the edge to the surface must be at least half of the radius of the smaller sphere, and the closest sphere centers to the midpoint of the edge must be the edge endpoints. The latter condition is equivalent to the requirement that additional edges must be in the Gabriel graph of the sphere centers see e. While other conditions can be formulated, we found that the Gabriel graph provides a good balance between sparsity and connectedness. Pinocchio precomputes the shortest paths between all pairs of vertices in this graph to speed up penalty function evaluation. Graph Construction Discretization Before any other computation, Pinocchio rescales the character to fit inside an axis-aligned unit cube. As a result, all of the tolerances are relative to the size of the character. To approximate the medial

surface and to facilitate other computations, Pinocchio computes a trilinearly interpolated adaptively sampled signed distance field on an octree [Frisken et al. It constructs a kd-tree to evaluate the exact signed distance to the surface from an arbitrary point. Because only negative distances i. The medial surface is the set of C 1 discontinuities of the distance field. Within a single cell of our octree, the interpolated distance field is guaranteed to be C 1 , so it is necessary to look at only the cell boundaries. Wade discusses a similar condition in Chapter 4 of his thesis []. The skeleton is given as a rooted tree on s joints. To reduce the degrees of freedom, for the discrete embedding, Pinocchio works with a reduced skeleton, in which all bone chains have been merged all degree two joints, such as knees, eliminated , as shown in Figure 5. The reduced skeleton thus has only r joints. This works because once Pinocchio knows where the endpoints of a bone chain are in V , it can compute the intermediate joints by taking the shortest path between the endpoints and splitting it in accordance with the proportions of the unreduced skeleton. Sphere Packing To pick out the graph vertices from the medial surface, Pinocchio packs spheres into the character as follows: Then it processes these points in order and if a point is outside all previously added spheres, adds the sphere centered at that point whose radius is the distance to the surface. In other words, the largest spheres are added first, and no sphere contains the center of another sphere Figure 3. Although the procedure described above takes O nb time in the worst case where n is the number of points, and b is the final number of spheres inserted , worst case behavior is rarely seen because most points are processed while there is a small number of large 3. A good embedding should have the proportions, bone orientations, and size similar to the given skeleton.

DOWNLOAD PDF AUTOMATIC RIGGING AND ANIMATION OF 3D CHARACTERS

Chapter 2 : SmartBody : Automatic Rigging

To appear in the ACM SIGGRAPH conference proceedings Automatic Rigging and Animation of 3D Characters Ilya Baranâˆ— Jovan PopovicÂ´â€ Computer Science and Artiï¬•cial Intelligence Laboratory.

Create a character name. Select the Save button. A progress bar will load as your file is prepared to be uploaded. A new browser window will open once the upload is complete. By default, characters will use a standard skeleton and have facial blendshapes enabled which allow facial animation in other programs. Once the automated rigging is complete, you can choose to change the default settings and re-rig with the new settings if you like. Otherwise, you can confirm the auto-rigging settings and proceed. Upload and rig a custom character Upload a custom character If you have your own 3D character you can upload it for automatic rigging. From the editor panel, click the Upload Character button to browse, select and upload a character. Mixamo supports 3 file formats for upload: Make sure "embed media" is turned on for FBX files to upload your textures. To show textures for an. Rig a custom character Your character will be displayed with markers once the upload is complete. Place the markers on key points wrists, elbows, knees, and groin by following the onscreen instructions. Confirm your marker placement to continue. The rigging process will begin once you confirm. It usually takes a few minutes to complete. You can download your rigged character by selecting the Download button from the editor panel or apply animations by selecting the Find Animations button. Your file must be in. Make sure that the Embed media option is turned on when creating the. Map a rigged character Rigged characters will have their skeleton automatically mapped to the Mixamo system, allowing animations to be applied to your custom skeleton.

DOWNLOAD PDF AUTOMATIC RIGGING AND ANIMATION OF 3D CHARACTERS

Chapter 3 : CiteSeerX â€" Citation Query Automatic rigging and animation of 3d characters

Animating an articulated 3D character currently requires manual rigging to specify its internal skeletal structure and to define how the input motion deforms its surface.

For example, Autodesk Maya 7 assigns weights based A Tutorial on Support Vector Machines for. We present a method for animating characters automatically. Given a static character mesh and a generic skeleton, our method adapts the skeleton to the character and attaches it to the surface, allowing skeletal motion data to animate the character. Because a single skeleton can be used with a wide range of characters, our method, in conjunction with a library of motions for a few skeletons, enables a user-friendly animation system for novices and children. Our prototype implementation, called Pinocchio, typically takes under a minute to rig a character on a modern midrange PC. User-friendly systems such as Teddy [Igarashi et al. Bringing these static shapes to life, however, is still not easy. In a conventional skeletal animation package, the user must rig the character manually. This requires placing the skeleton joints inside the character and specifying which parts of the surface are attached to which bone. The tedium of this process makes simple character animation more difficult than it could be. We envision a system that eliminates this tedium to make animation more accessible for children, educators, researchers, and other non-expert animators. To support this functionality, we need a method as shown in Figure 1 that takes a character, a skeleton, and a motion of that skeleton as input, and outputs the moving character. The missing portion is the rigging: Our algorithm consists of two main steps: The automatic rigging method presented in this paper allowed us to implement an easy-to-use animation system, which we called Pinocchio. In this example, the triangle mesh of a jolly cartoon character is brought to life by embedding a skeleton inside it and applying a walking motion to the initially static shape. Our design decisions relied on three criteria, which we also used to evaluate our system: A single skeleton is applicable to a wide variety of characters: The resulting animation quality is comparable to that of modern video games. The automatic rigging usually takes under one minute on an everyday PC. Automatic Rigging and Animation of 3D Characters. Copyright Notice Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. Permissions may be requested from Publications Dept. For this, we designed a maximum-margin supervised learning method to combine a set of hand-constructed penalty functions. To ensure an honest evaluation and avoid overfitting, we tested our algorithm on 16 characters that we did not see or use during development. Our algorithm computed a good rig for all but 3 of these characters. For each of the remaining cases, one joint placement hint corrected the problem. We simplify the problem by making the following assumptions. The character mesh must be the boundary of a connected volume. The character must be given in approximately the same orientation and pose as the skeleton. Lastly, the character must be proportioned roughly like the given skeleton. We introduce several new techniques to solve the automatic rigging problem: This method could also be useful in existing 3D packages. Our prototype system, called Pinocchio, rigs the given character using our algorithm. It then transfers a motion to the character using online motion retargetting [Choi and Ko] to eliminate footskate by constraining the feet trajectories of the character to the feet trajectories of the given motion. Recent exceptions include Motion Doodles [Thorne et al. These approaches focus on simplifying animation control, rather than simplifying the definition of the articulation of the character. In particular, a spatial keyframing system expects an articulated character as input, and as-rigid-as-possible shape manipulation, besides being 2D, relies on the constraints to provide articulation information. The Motion Doodles system has the ability to infer the articulation of a 2D character, but their approach relies on very strong assumptions about how the character is presented. Character Animation Although most skeleton-based prior work on automatic rigging focused on skeleton extraction, for

our problem, we advocate skeleton embedding. A few approaches to the skeleton extraction problem are representative. Teichmann and Teller [] extract a skeleton by simplifying the Voronoi skeleton with a small amount of user assistance. In their paper, Katz and Tal [] describe a surface partitioning algorithm and suggest skeleton extraction as an application. The technique in Wade [] is most similar to our own: For the purpose of automatically animating a character, however, skeleton embedding is much more suitable than extraction. For example, the user may have motion data for a quadruped skeleton, but for a complicated quadruped character, the extracted skeleton is likely to have a different topology. The anatomically appropriate skeleton generation by Wade [] ameliorates this problem by techniques such as identifying appendages and fitting appendage templates, but the overall topology of the resulting skeleton may still vary. For example, for the character in Figure 1, ears may be mistaken for arms. Another advantage of embedding over extraction is that the given skeleton provides information about the expected structure of the character, which may be difficult to obtain from just the geometry. So although we could use an existing skeleton extraction algorithm and embed our skeleton into the extracted one, the results would likely be undesirable. For example, Skeleton Extraction the legs of the character in Figure 1 would be too short if a skeleton extraction algorithm were used. Animating user-provided data by fitting a template has been successful in cases when the model is fairly similar to the template. Most of the work has been focused on human models, making use of human anatomy specifics, e. For segmenting and animating simple 3D models of characters and inanimate objects, Anderson et al. Template Fitting Almost any system for mesh deformation whether surface based [Lipman et al. Teichmann and Teller [] propose a spring-based method. Unfortunately, at present, these methods are unsuitable for real-time animation of even moderate size meshes. Because of its simplicity and efficiency and simple GPU implementation , and despite its quality shortcomings, linear blend skinning LBS , also known as skeleton subspace deformation, remains the most popular method used in practice. Most real-time skinning work, e. However, such techniques are unsuitable for our problem because we only have a single mesh. Instead, we must infer articulation by using the given skeleton as an encoding of the likely modes of deformation, not just as an animation control structure. To our knowledge, the problem of finding bone weights for LBS from a single mesh and a skeleton has not been sufficiently addressed in the literature. Previous methods are either mesh resolution dependent [Katz and Tal] or the weights do not vary smoothly along the surface [Wade], causing artifacts on highresolution meshes. Some commercial packages use proprietary methods to assign default weights. For example, Autodesk Maya 7 assigns weights based solely on the vertex proximity to the bone, ignoring the mesh structure, which results in serious artifacts when the mesh intersects the Voronoi diagram faces between logically distant bones. Skinning 3 Skeleton Embedding Skeleton embedding resizes and positions the given skeleton to fit inside the character. This can be formulated as an optimization problem: Solving such a problem directly using continuous optimization is infeasible. Pinocchio therefore discretizes the problem by constructing a graph whose vertices represent potential joint positions and whose edges are potential bone segments. This is challenging because the graph must have few vertices and edges, and yet capture all potential bone paths within the character. The graph is constructed by packing spheres centered on the approximate medial surface into the character and by connecting sphere centers with graph edges. Pinocchio then finds the optimal embedding of the skeleton into this graph with respect to a discrete penalty function. It uses the discrete solution as a starting point for continuous optimization. To help with optimization, the given skeleton can have a little extra information in the form of joint attributes: Approximate Medial Surface Figure 3: These attributes are specific to the skeleton but are independent of the character shape and do not reduce the generality of the skeletons. The original and reduced quadruped skeleton spheres. The final discretization step constructs the edges of the graph by connecting some pairs of sphere centers Figure 4. Pinocchio adds an edge between two sphere centers if the spheres intersect. The precise condition Pinocchio uses is that the distance from any point of the edge to the surface must be at least half of the radius of the smaller sphere, and the closest sphere centers to the midpoint of the edge must be the edge endpoints. The latter condition is equivalent to the requirement that additional edges must be in the Gabriel graph of the

sphere centers see e. While other conditions can be formulated, we found that the Gabriel graph provides a good balance between sparsity and connectedness. Pinocchio precomputes the shortest paths between all pairs of vertices in this graph to speed up penalty function evaluation. Graph Construction Discretization Before any other computation, Pinocchio rescales the character to fit inside an axis-aligned unit cube. As a result, all of the tolerances are relative to the size of the character. To approximate the medial surface and to facilitate other computations, Pinocchio computes a trilinearly interpolated adaptively sampled signed distance field on an octree [Frisken et al. It constructs a kd-tree to evaluate the exact signed distance to the surface from an arbitrary point. Because only negative distances i. Distance Field Pinocchio uses the adaptive distance field to compute a sample of points approximately on the medial surface Figure 2. The medial surface is the set of C 1 discontinuities of the distance field. Within a single cell of our octree, the interpolated distance field is guaranteed to be C 1 , so it is necessary to look at only the cell boundaries. Wade discusses a similar condition in Chapter 4 of his thesis []. Approximate Medial Surface To pick out the graph vertices from the medial surface, Pinocchio packs spheres into the character as follows: Then it processes these points in order and if a point is outside all previously added spheres, adds the sphere centered at that point whose radius is the distance to the surface. In other words, the largest spheres are added first, and no sphere contains the center of another sphere Figure 3. Although the procedure described above takes O nb time in the worst case where n is the number of points, and b is the final number of spheres inserted , worst case behavior is rarely seen because most points are processed while there is a small number of large Sphere Packing Figure 4:

DOWNLOAD PDF AUTOMATIC RIGGING AND ANIMATION OF 3D CHARACTERS

Chapter 4 : CiteSeerX â€" Automatic Rigging and Animation of 3D Characters

The Pinocchio systems automatically places skeletal joints inside 3D characters. Published in SIGGRAPH Automatic Rigging and Animation of 3D Characters Making the 3D Printed.

Each segment is a portion of the mesh that is interrelated. For example, segments and are ear segments; and segment is a nose segment. A process for determining the segments of the representative mesh in accordance with embodiments of this invention is described below with reference to FIG. After the segments of the representative mesh are identified, key points of the representative mesh are identified Key points are points at the borders of segments that can be utilized in the animation of the mesh. An example of key points in a representative mesh is shown in FIG. Key points are points such as but not limited to the edges of the mouth, the center of the mouth, the tip of the nose and the corners of the eyes that need to be identified to provide animation of the facial mesh A process for identifying the key points of a segmented representative mesh in accordance with embodiments of this invention is described below with reference to FIG. The bone set is a set of bones that can be utilized to rig the facial mesh. Each bone is a line that establishes a relationship between segments of the mesh and that is used to manipulate the related segments of the mesh during animation. An example of a bone set in accordance with embodiments of this invention is shown in FIG. The bone set includes bones that each establish a relationship between specific portions of representative mesh A process for generating the bone set from the segmentation and key points of the representative mesh in accordance with embodiments of this invention is described below with reference to FIG. The joint centers may be identified using a Support Vector Machine using the location of key points as an input in accordance with embodiments of this invention. In other embodiments, any of a variety of techniques can be utilized to identify joint centers as appropriate to the requirements of a specific application. Once the joint centers are determined, the bones of the bone set are placed in the representative mesh. The orientation of bone local coordinate frames is defined in order to allow for an optimal ability to edit the animation curves. The representative mesh with the bones placed in the mesh can be used to determine the skinning weights for the segments of the representative. The skinning weights define the relation between the motion of the 3D representation polygons segments and the underlying bone set. An example of skinning weights in accordance with embodiments of this invention is shown in FIG. A process for determining the skinning weights of the representative mesh in accordance with embodiments of this invention is described below with reference to FIG. In accordance with some embodiments of this invention, the translation of the bones and skinning weights may be performed using a mean value coordinate system. In accordance with other embodiments of this invention, the translation may be performed using other processes including but not limited to a point to polygon nearest neighbor search process. In the point to polygon process, each vertex in the original mesh is associated with a closest polygon segment on the representative mesh used for skinning processing. Local interpolation is then used to derive skinning weights of the original mesh vertices from those of the respective closest polygons segments vertices on the representative mesh. After the bone set and skinning weights are translated to the original mesh to create the rigged mesh, user inputs may be used to improve the topology of the rigged mesh. The improved topology may allow the rigging mesh to deform better when the bones are animated. Although a specific process for automatically rigging a facial mesh is discussed above with reference to FIG. Various processes that can be utilized to identify representative meshes, identify segments of representative meshes corresponding to portions of a face, identify a bone set and skinning weights for the representative mesh, and for rigging the original facial mesh based upon the rigging of the representative mesh in accordance with embodiments of the invention are discussed further below. Identifying Segments of a Facial Mesh Corresponding to Portions of a Face When a representative mesh has been generated to deal with problematic areas of an original facial mesh of a 3D character, segments of the representative mesh corresponding to portions of a face can be identified. In many embodiments, artificial intelligence is used to identify the

segments. In particular, machine learning processes can be used to identify various segments of the representative mesh corresponding to different portions of a face. A process that identifies segments of a representative mesh corresponding to portions of a face in accordance with embodiments of this invention is shown in FIG. The process utilizes a representative mesh generated from an original facial mesh of a 3D character. The process also utilizes a training set of facial meshes in which segments corresponding to portions of a face or other appropriate portions are identified. The training set may include a training set of meshes, at least one of a testing set of meshes, and a set of gold standard segmentations. The testing set of meshes includes meshes which the machine learning process can use to learn to identify the segments of a facial mesh before generating the segments for the representative mesh. The gold standard segments are a set of facial meshes in which all of the segments are clearly identified to aid in learning for the identification process. The process can perform feature extraction on the representative mesh and the meshes in the training set. The extracted features are then provided to a learning process to perform the identification of the segments in the representative mesh. The JointBoost process learns probabilistically to estimate the segments of the representative mesh The CRF process is a learning process that receives the probabilities of the segments from the JointBoost process and determines a final segmentation of the representative mesh. Identification of Key Points within a Representative Mesh Once the segments of the representative mesh are identified, the key points of the representative mesh can be determined. Key points are typically points at borders between segments in a representative mesh. A machine learning process may be used to identify the key points. To aid in identification of key points, a user interface for providing data about key points is provided in accordance with some embodiments of this invention. A process for identifying key points from a representative mesh in which segments corresponding to portions of a face have been identified in accordance with an embodiment of this invention is shown in FIG. The process begins by receiving inputs of data from a user that helps in identifying key points through a user interface or via metadata associated with the representative mesh. This data may include, but is not limited to, the corner of the eyes; the tip of the nose; the corners of the mouth; and the inner and outer points on the eyebrows. The input data is then provided to one or more machine learning processes, such as the JointBoost and CRF process combination that identifies the key points which uses the input data to better position the key points. In other embodiments, key points are identified manually. In several embodiments, any of a variety of processes trained using any of a variety of machine learning techniques can be utilized to identify key points. Based on the key points, a bone set can be generated. The generation of a bone set in accordance with embodiments of the invention is discussed further below. Generating a Bone Set A bone set may be generated for the representative mesh after the segments and key points of the representative mesh are identified. Each bone is a line that connects two or more segments of the mesh that can be manipulated to animate the mesh. The generation of the bone set includes determining the number of bones needed, size of each bone, the placement of each bone, and the interconnection of bones. The generation of the bone set is typically performed by a machine learning process. A user interface that allows a user to enter specific parameters such as, but not limited to, the number of bones; and maximum and minimum size of the bones may be provided to allow a user to customize the bone set. A process for generating the bone set for a representative mesh using a machine learning process in accordance with an embodiment of this invention is shown in FIG. The process begins by receiving the representative mesh with identified segments and key points. A training set of meshes with bone sets is received User input parameters can also be received A Support Vector Machine can be used to determine the bone set for the representative mesh. The Support Vector Machine receives the training set of meshes with bone sets to learn bone placement. The Support Vector Machine then receives the user input and representative mesh and generates the bone set for the representative mesh based upon the training set and user inputs. Although a Support Vector Machine is referenced above, any of a variety of machine learning techniques can be utilized to automate bone placement as appropriate to the requirements of a specific application in accordance with embodiments of the invention. Once the bones of the bone set have been placed and the joint centers of the bones have been identified, the

skinning weights for the representative mesh may be determined. Processes for determining skinning weights in accordance with embodiments of the invention are discussed further below. Determining Skinning Weights A set of skinning weights defines the relationships between the motion of the segments of the mesh and underlying bone set. In accordance with embodiments of this invention, one of many methods such as, but not limited to, diffusion, proximity maps, and templates may be used to determine the skinning weights. User inputs may also be used in these methods to determine skinning weights. A process for determining skinning weights for a representative mesh based and bone set using information concerning segments, and key points within the representative mesh in accordance with an embodiment of this invention is shown in FIG. The process receives the representative mesh with segments, key points, and a bone set. The parameters of the skinning weights are determined The parameters may be determined by a machine learning process, manually input by a user, or a combination of these processes. Poisson problems are then generated for each segment. The Poisson problems may be generated by modeling each segment as a heat emitting body with the surrounding segments considered heat absorbing bodies. These parameters may be estimated using machine learning processes or input by a user. The results of the Poisson problems provide skinning template profiles. Each skinning template profile is specific for a given joint and represents a typical spatial distribution of the skinning weights on a mesh. The skinning templates are then used to determine the skinning weights of the segments using a machine learning process such as, but not limited to, Bayesian networks and in general Bayesian estimation. The above is description of embodiments of systems and methods in accordance with the present invention. It is foreseen that other skilled in the art will design alternative systems that infringe on this invention as set forth in the following claims either literally or through the Doctrine of Equivalents. A method for automatic rigging of a three dimensional character by a comprising: The method of claim 1 wherein the generating of the representative mesh is completed by performing a volumetric method on the original mesh. The method of wherein the volumetric method includes generating a visual hull. The method of claim 1 wherein the determining the plurality of segments is done by performing a machine learning process to assign the segments. The method of wherein the machine learning process is a JointBoost process. The method of wherein the performing of the machine learning process comprises: The method of claim 6 wherein the determining of the plurality of segments further comprises: The method of claim 7 wherein the determining of the of the plurality of segments from the estimated plurality of segments is performed by applying a Conditional Random Fields CFR process to the estimated plurality of segments using the training set of segments. The method of claim 6 wherein the training set of a plurality of meshes includes a training set of a plurality of meshes at least one of a test set of a plurality of meshes and a gold standard of a plurality of segments. The method of claim 1 wherein the key points are identified by applying a machine learning process to the segments of the representative mesh. The method of claim 10 wherein the identifying of the key points includes receiving user inputs of points on the representative mesh that are provided to the machine learning process. The method of claim 1 wherein the generating of a bone set comprises: The method of wherein the machine learning process is a Support Vector Machine. The method of wherein the generating of the bone set further comprises: The method of wherein the determining of the skinning weights is determined using a diffusion process.

DOWNLOAD PDF AUTOMATIC RIGGING AND ANIMATION OF 3D CHARACTERS

Chapter 5 : Automatic Rigging and Animation of 3D Characters - calendrierdelascience.com

Automatic rigging for animation characters with 3D silhouette By JunJun Pan, Xiaosong Yang, Xin Xie, Philip Willis and Jian J Zhang.*

Created by feng, last modified by shapiro on Oct 31, Introduction Rigging is the process of aligning a skeleton structure into a 3D model and assigning the suitable skinning weights for each vertex. It is a time consuming process even by an experienced animator. Although sometimes it is possible to obtain a rigged model to use in SmartBody, most of the time only 3D models could be obtained either from the web or via 3D scans. The user can input a human-like 3D model and have it immediately animated in SmartBody environment. How to Use First, we need to create a pawn with 3D mesh in the scene. Then in Resource Panel, select the newly created pawn and set its attribute "mesh" to the correct mesh asset in the system. The model then will be shown on the screen. However, since this is only a static mesh without any skeleton or skinning weight information, we need to apply auto-rigging process to synthesize its skeleton and skin bindings. It will shows a list of pawns on the upper-right. Select the desired pawn and choose AutoRigging to start the rigging process. Once it is done, a new character will be created with newly synthesized skeleton in the viewer. The original Pinocchio algorithm works only for water-tight and single component mesh. However, a lot of 3D models used in production may contain either holes or additional mesh components for hairs or props. Thus it is difficult to produce a model that can work out-of-box in Pinocchio. To alleviate this problem, SmartBody provides a voxel-rigging option to pre-process the input 3D model into voxel representation. The voxel model is used for auto-rigging in place of the original mesh to ensure a watertight and connected model is used in Pinocchio. The rigging results from Pinocchio are then mapped back to the origianl model to complete the auto-rigging. This option is more time consuming but can works for most production 3D models. Thus it is recommended when the original method fails. It can be enabled by selecting "VoxelRigging" checkbox. Limitation Although the auto-rigging can be applied on most 3D models, there are a few cases it may fail or produce undesired results. First, it only works for human-like models with limbs proportion similar to a human. Thus for characters with very short or long limbs, the method may produce a bad skeleton fitting. Also, the method can not deal the hollow or empty space inside the model correct. The method will have difficulty inferring the arm joints in this situations. This situation is rare but can happen in some mechanical or cartoon characters. Rigging with Scripts The autorigging capability can be used without a user interface by calling the Python commands to load the asset, associate the model with a pawn, then save out the rig to a. Here is an example:

Chapter 6 : DSpace@MIT: Automatic rigging and animation of 3D characters

Automatic Rigging and Animation of 3D Characters rigging to specify its internal skeletal structure and to define how the input . For example, Autodesk Maya 7 assigns . a bone chain are in V, it can compute the intermediate joints by taking the.

Details in mesh animations are difficult to generate but they have great impact on visual quality. In this work, we demonstrate a practical software system for capturing such details from multi-view video recordings. Given a stream of synchronized video images that record a human performance from mu Given a stream of synchronized video images that record a human performance from multiple viewpoints and an articulated template of the performer, our system captures the motion of both the skeleton and the shape. The output mesh animation is enhanced with the details observed in the image silhouettes. For example, a performance in casual loose-fitting clothes will generate mesh animations with flowing garment motions. We accomplish this with a fast pose tracking method followed by nonrigid deformation of the template to fit the silhouettes. The entire process takes less than sixteen seconds per frame and requires no markers or texture cues. Captured meshes are in full correspondence making them readily usable for editing operations including texturing, deformation transfer, and deformation model learning. This paper proposes a method for capturing the performance of a human or an animal from a multi-view video sequence. Given an articulated template model and silhouettes from a multi-view image sequence, our approach recovers not only the movement of the skeleton, but also the possibly non-rigid temp Given an articulated template model and silhouettes from a multi-view image sequence, our approach recovers not only the movement of the skeleton, but also the possibly non-rigid temporal deformation of the 3D surface. While large scale deformations or fast movements are captured by the skeleton pose and approximate surface skinning, true small scale deformations or non-rigid garment motion are captured by fitting the surface to the silhouette. We show on various sequences that our approach can capture the 3D motion of animals and humans accurately even in the case of rapid movements and wide apparel like skirts. Show Context Citation Context The weights allow us to do skinning, i. Weighted skinning is used to interpolate the joint transformations on a A statistical model of human pose and body shape. Computer Graphics Forum28 by N. Seidel, Mpi Informatik , " Especially, the games and special effects industry heavily depend on realistic human animation. In this work a unified model that describes both, human pose and body shape is introduced whic In this work a unified model that describes both, human pose and body shape is introduced which allows us to accurately model muscle deformations not only as a function of pose but also dependent on the physique of the subject. A learning based approach is trained on approximately full body 3D laser scans taken of subjects. Scan registration is performed using a non-rigid deformation technique. Then, a rotation invariant encoding of the acquired exemplars permits the computation of a statistical model that simultaneously encodes pose and body shape. Finally, morphing or generating meshes according to several constraints simultaneously can be achieved by training semantically meaningful regressors. We create a symmetric template by triangulating a 3D scan, enforcing symmetry by hand and manually fitting a skeleton into the model. Scan Database The project is based on a data Skinning of skeletally deformable models is extensively used for real-time animation of characters, creatures and similar objects. The standard solution, linear blend skinning, has some serious drawbacks that require artist intervention. Therefore, a number of alternatives have been proposed in re Therefore, a number of alternatives have been proposed in recent years. All of them successfully combat some of the artifacts, but none challenge the simplicity and efficiency of linear blend skinning. As a result, linear blend skinning is still the number one choice for the majority of developers. In this paper, we present a novel skinning algorithm based on linear combination of dual quaternions. Even though our proposed method is approximate, it does not exhibit any of the artifacts inherent in previous methods and still permits an efficient GPU implementation. Upgrading an existing animation system from linear to dual quaternion skinning is very easy and has a relatively minor

impact on run-time performance. Traditional motion-capture systems excel at recording motions within lab-like environments but struggle with recording outdoor activities such as skiing, biking, and driving. This limitation led us to design a wearable motion-capture system that records human activity in both indoor and ou This limitation led us to design a wearable motion-capture system that records human activity in both indoor and outdoor environments. Commercial motion-capture systems produce excellent in-studio reconstructions, but offer no comparable solution for acquisition in everyday environments. We present a system for acquiring motions almost anywhere. This wearable system gathers ultrasonic time-of-flight and inertial measurements with a set of inexpensive miniature sensors worn on the garment. After recording, the information is combined using an Extended Kalman Filter to reconstruct joint configurations of a body. Experimental results show that even motions that are traditionally difficult to acquire are recorded with ease within their natural settings. Although our prototype does not reliably recover the global transformation, we show that the resulting motions are visually similar to the original ones, and that the combined acoustic and inertial system reduces the drift commonly observed in purely inertial systems. Our final results suggest that this system could become a versatile input device for a variety of augmented-reality applications. Two representative models that users can interactively manipulate within our deformation system. A desk lamp connected by revolute joints, and its color-coded components. The lampshade is manipulated with the same handle trajectory for three cases: We propose a joint-aware deformation framework that supports the direct manipulation of an arbitrary mix of rigid and deformable components. First we apply slippable motion analysis to automatically detect multiple types of joint constraints that are implicit in model geometry. For single-component geometry or models with disconnected components, we support user-defined virtual joints. Then we integrate manipulation handle constraints, multiple components, joint constraints, joint limits, and deformation energies into a single volumetric-cell-based space deformation problem. An iterative, parallelized Gauss-Newton solver is used to solve the resulting nonlinear optimization. Interactive deformable manipulation is demonstrated on a variety of geometric models while automatically respecting their multi-component nature and the natural behavior of their joints. An animation of an actor created with our method from a multi-view video database. The motion was designed by an animator and the camera was tracked from the background with a commercial camera tracker. In the composited scene of animation and background, the synthesized character and her In the composited scene of animation and background, the synthesized character and her spatio-temporal appearance look close to lifelike. We present a method to synthesize plausible video sequences of humans according to user-defined body motions and viewpoints. We first capture a small database of multi-view video sequences of an actor performing various basic motions. This database needs to be captured only once and serves as the input to our synthesis algorithm. We then apply a marker-less model-based performance capture approach to the entire database to obtain pose and geometry of the actor in each database frame. To create novel video sequences of the actor from the database, a user animates a 3D human skeleton with novel motion and viewpoints. Our technique then synthesizes a realistic video sequence of the actor performing the specified motion based only on the initial database. The first key component of our approach is a new efficient retrieval strategy to find appropriate spatio-temporally coherent database frames from which to synthesize target video frames. The second key component is a warping-based texture synthesis approach that uses the retrieved most-similar database frames to synthesize spatio-temporally coherent target video frames. We show through a variety of result videos and a user study that we can synthesize realistic videos of people, even if the target motions and camera views are different from the database content. Range scan registration using reduced deformable models by W. Zwicker - EG , " We present an unsupervised method for registering range scans of deforming, articulated shapes. The key idea is to model the motion of the underlying object using a reduced deformable model. We use a linear skinning model for its simplicity and represent the weight functions on a regular grid locali We use a linear skinning model for its simplicity and represent the weight functions on a regular grid localized to the surface geometry. This decouples the deformation model from the surface representation and allows us to deal with the severe occlusion and missing data that is inherent in range scan

data. We formulate the registration problem using an objective function that enforces close alignment of the 3D data and includes an intuitive notion of joints. This leads to an optimization problem that we solve using an efficient EM-type algorithm. With our algorithm we obtain smooth deformations that accurately register pairs of range scans with significant motion and occlusion. The main advantages of our approach are that it does not require user specified markers, a template, nor manual segmentation of the surface geometry into rigid parts. Experimental Results We tested our algorithm with five datasets: Each dataset consists of a sequence of depth scans of a moving object. Example-based facial rigging allows transferring expressions from a generic prior to create a blendshape model of a virtual character. This blendshape model c This blendshape model can be successively fine-tuned toward the specific geometry and motion characteristics of the character by providing more training data in the form of additional expression poses. We introduce a method for generating facial blendshape rigs from a set of example poses of a CG character. Our system transfers controller semantics and expression dynamics from a generic template to the target blendshape model, while solving for an optimal reproduction of the training poses. This enables a scalable design process, where the user can iteratively add more training poses to refine the blendshape expression space. However, plausible animations can be obtained even with a single training pose. We show how formulating the optimization in gradient space yields superior results as compared to a direct optimization on blendshape vertices. We provide examples for both hand-crafted characters and 3D scans of a real actor and demonstrate the performance of our system in the context of markerless art-directable facial tracking. A result of our method: The resulting motion is physi The resulting motion is physically plausible, maintains the original artistic intent, and is easily editable. We present a method that brings the benefits of physics-based sim-ulations to traditional animation pipelines. Our framework fits seamlessly into the workflow typically employed by artists, as our output consists of animation curves that are identical in nature to the result of manual keyfram-ing. Artists can therefore explore the full spectrum between hand-crafted animation and unrestricted physical simulation. In ad-dition, we use automatically extracted high-level rig parameters to intuitively edit the results of our simulations, and also to speed up computation.

Chapter 7 : Upload and rig 3D characters with Mixamo online services

Automatic rigging, Generation and placement of an animation skeleton, is developed to avoid tedious and time-consuming manual rigging. Here, an animation skeleton typically consisting of bones and.