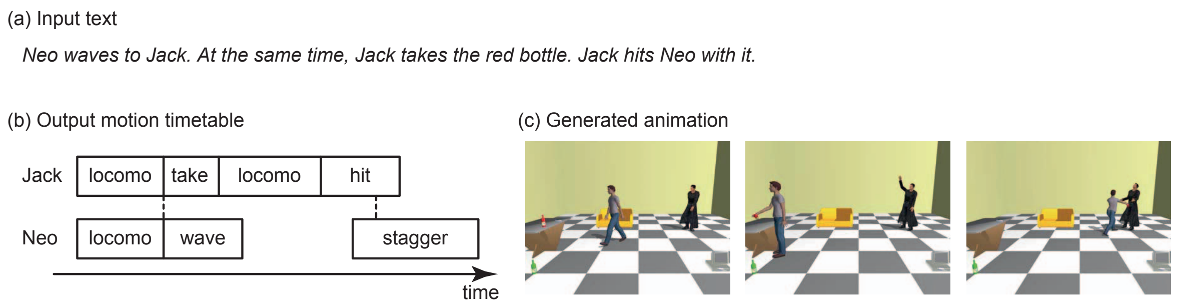
Generating Animation from Natural Language Texts and Framework of Motion Database

Masaki Oshita
Kyushu Institute of Technology
680-4 Kawazu, Iizuka, Fukuoka, 820-8502, Japan
e-mail: oshita@ces.kyutech.ac.jp


(a) Input text

Neo waves to Jack. At the same time, Jack takes the red bottle. Jack hits Neo with it.

Figure 1. Example of our system. (a) Input text. (b) Searched motion clips and their execution timings. (c) Generated animation.

***Abstract*— This paper presents an animation system that generates an animation from natural language texts such as movie scripts or stories. We also propose the framework of a motion database that stores many motion clips for various characters. When an input text is given, the system searches for an appropriate motion clip from the database for each verb in the input text. Temporal constraints between verbs are also extracted from the input text. The searched motion clips are scheduled based on these temporal constraints. In addition, when necessary, some automatic motions such as locomotion, taking an instrument, changing posture, and cooperative motions are searched from the database. An animation is then generated using an external motion synthesis system. Using our system, users can make use of existing motion clips. Moreover, because it takes natural language text as input, even novice users can use our system.**

Keywords-component; computer animation, motion database, natural language processing.

I. INTRODUCTION

Recently, computer animation has been widely used in movies, video games, TV programs, web graphics, etc. Because computer animation is a very powerful tool to present a story, drama, or instruction, there are demands from non-professional people to create computer animation. However, it is a difficult task because of two main issues. The first issue is the difficulty of making and reusing motion data. Currently, motion data are mainly created using motion capture or keyframe techniques. Either way, they are very time consuming and require professional skills. Although there are demands for reusing existing motion data, this is difficult because of the lack of a system for storing and searching large amounts of motion data. Because there can be various motions of various characters, it is difficult to manage them in a standard file system or database. Currently, most motion data are created from scratch for individual scenes and are thrown away without reuse. The second issue is the limitation of current animation systems. A computer animation can be created by combining a number of existing motion clips using animation software such as MotionBuilder, Maya, 3ds Max, etc. However, it is difficult for novice users to utilize such software, because handling motion data is tricky and these systems require training.

To address these issues, we developed an animation system that generates an animation from natural language texts such as movie scripts or stories (Fig. 1). We also developed a motion database that stores many motion clips for different characters. When an input text is given, the system searches for an appropriate motion clip from the database for each verb. Temporal constraints between verbs are also extracted from the input text. The searched motion clips are scheduled based on the temporal constraints. In addition, when necessary, some automatic motions such as locomotion, taking an instrument, changing posture, and cooperative motions are searched from the database. The system outputs a motion timetable which consists of motion clips and their execution timings. An animation is then generated using an external motion synthesis system. Using our system, even novice users can create animation by making use of existing motion clips.

There are many possible applications of our system. Recently, in movie production, simple animations are created before production to check camerawork, screenplay, necessary visual effects, etc. These animations are called “previsualization” or “animatics”. They are also often created for the scenes in which no computer graphics are involved. Using our system, even directors or writers who are not professional animators can create an animation very quickly. Moreover, our system can be used by non-professional people who want to make an animation but do not have professional skills. It can also be used for children to visualize a story to make it interesting and easy to understand. Our system can be used for movie production.

Even though animators want to add more details to the output of our system, our method is much easier than making animations from scratch.

In this paper, we propose a motion frame that contains meta-information about a motion clip, an object-oriented database framework for storing a number of motions of a number of characters in a hierarchical structure, natural language analysis methods that are specialized for extracting motion related descriptions from an input text, and scheduling of multiple motions based on the temporal constraints in an input text. In addition, we have done preliminary experiments which showed that our system generates expected results from various input texts.

The rest of this paper is organized as follows. Section II reviews the related work in the literature. Section III gives an overview of our system. Section IV, V, and VI describe our methods for the framework of the motion database, motion search and motion scheduling, respectively. Section VII shows some experimental results and discussions. Finally, Section VIII concludes the paper.

II. Related Work

Generating animation from natural language texts has been a challenge. Many research groups have tackled this problem. The SHRDLU system, which was developed by Winograd [1], is known as the pioneer. Using SHRDLU, a user can give commands to a robot using English in an interactive manner, and make it arrange objects in a scene. However, the types of commands were very limited.

Badler et al. [2][3] developed virtual agents that follow natural language interactions. They proposed Parameterized Action Representation (PAR), which has a similar purpose to the motion frame in our research. The PAR has more complex information such as pre-condition and achievement. The motion generator of each PAR is programmed using a state machine. It can use motion data or any motion generation methods. However, specifying detailed information and constructing motion generators are very time consuming.

Tokunaga et al. [4] developed the K2 system, which has similar goals to Badler et al. In their system, agents are controlled via spoken language. Their research is rather focused on solving the vagueness of natural language instructions. They use case frames [5] to search for motions. Unlike our work, they use all cases that are used in linguistic analysis. The interpretation of each case is left to the user who adds the case frame handler. The motion generator for each case frame must be manually programmed by the user.

These previous works aim at developing intelligent agents that understand natural language instructions and make plans to execute them. However, the systems are very complex, and many rules are required. On the other hand, our system aims to reuse existing motion data easily and efficiently. The motion frame in our work contains just enough information to search for appropriate motions that match natural language texts and it is easy to describe. We believe that our system is more practical.

Sumi et al. [6] developed a system for visualizing short stories for children. The system extracts keywords from an input text, and chooses an appropriate scene, characters, and motions from a database. It simply plays a motion that matches the keywords. Although a user can add motion data to the system, the system cannot select motions appropriate for the objects or characters and cannot generate interactions between characters and the scene.

There is very little research that deals with motion scheduling from natural language texts. The above systems simply execute motions as instructions are given or events happen, and no scheduling is considered. However, in order to execute multiple motions of multiple characters as instructed by an input text, the execution timing of the motions must be coordinated. Baba et al. [7] developed a system for generating an animation that satisfies temporal and spatial constraints given by natural language texts. The system determines appropriate initial positions of the agents and objects that are specified in the input text. However, the motions of the agents and motion scheduling were not considered.

Coyne and Sproat [8] developed WordsEye, which converts natural language texts to a scene. Because their purpose is to generate a still image, when a character motion is indicated in a given text, the system simply chooses a pose for the action from the database.

There are animation engines that support some script language such as Improv [9] and Alice [10]. However, it is still difficult to program the agents and to make use of a large amount of existing motion data. In addition, markup language formats for describing animation including scenes, characters and actions have been proposed [11][12]. However, they are difficult to describe by hand. The animation files should be created by using specific authoring software. Moreover, it is difficult to add and reuse motion data using such file formats and authoring software.

There are many motion synthesis methods which generate new motions from a small number of motions [13][14]. However, they require a manual setup for each motion module. It is difficult for end users to add new motion modules. Although currently our system selects one motion from the database, it is possible to extend our system to blend a number of selected motions based on quantitative motion query parameters such as contact position.

III. System Overview

In this section, we explain the overview of our system (Fig. 2) and data representation (Fig. 3).

When an input text is given to the system, natural language processes (syntax analysis and semantic analysis) are applied first. The syntax analysis is the process of converting a plain text to a tree structure with phrase tags and dependencies. Fig. 3(b) is an example of the analyzed tree which is computed from an input text (Fig. 3(a)). The type of each phrase and the dependency between phrases are determined. For example, S, NP, VP and PR in Fig. 3(b) represent sentence, noun phrase, verb phrase and preposition, respectively.

The semantic analysis extracts information about motions described in the input text from the tree structure. A query frame contains information for the motion search. One is generated for each verb in the text. The temporal constraints contain information about execution timing between verbs. For example, QF1~QF3 and TC1~TC2 in Fig. 3(c) represent query frames and temporal constraints,

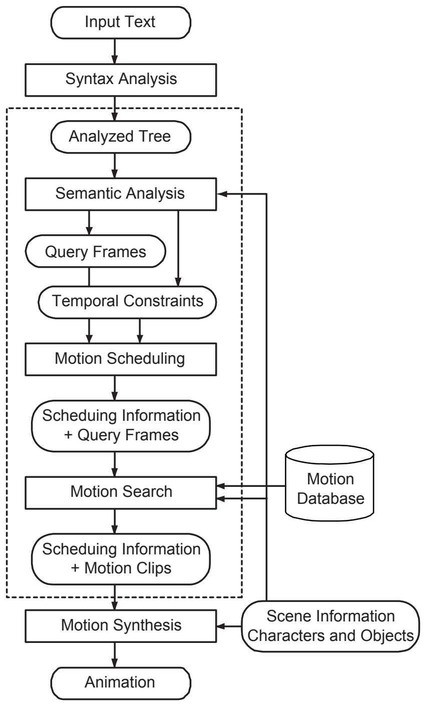
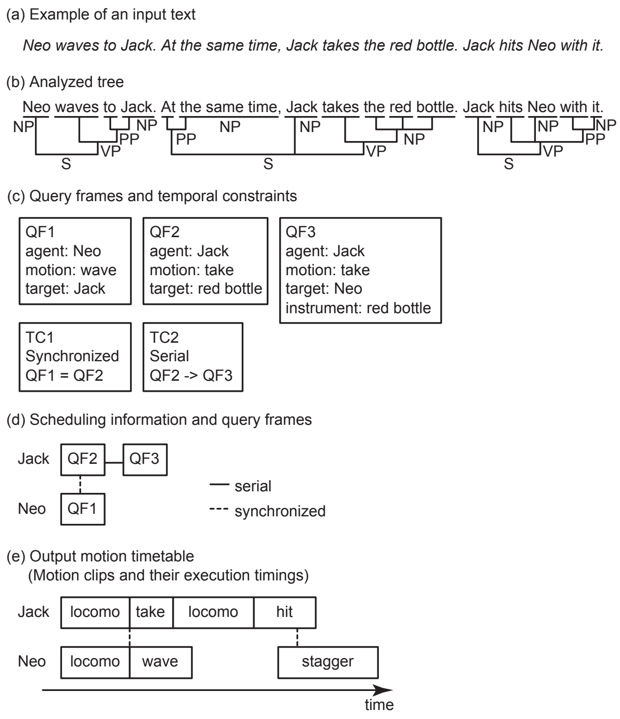


Figure 2. System overview



(a) Example of an input text
Neo waves to Jack. At the same time, Jack takes the red bottle. Jack hits Neo with it.
(b) Analyzed tree
Neo waves to Jack. At the same time, Jack takes the red bottle. Jack hits Neo with it.
NP NP PP VP S PP NP NP NP VP S NP NP NP PP VP S
(c) Query frames and temporal constraints
QF1 agent: Neo motion: wave target: Jack
QF2 agent: Jack motion: take target: red bottle
QF3 agent: Jack motion: take target: Neo instrument: red bottle
TC1 Synchronized QF1 = QF2
TC2 Serial QF2 -> QF3
(d) Scheduling information and query frames
Jack QF2 QF3
Neo QF1
serial
synchronized
(e) Output motion timetable (Motion clips and their execution timings)
Jack locomo take locomo hit
Neo locomo wave stagger
time

Figure 3. Example of data representation.

respectively.

Based on the temporal constraints, motion scheduling determines the execution order of each motion clip, which corresponds to each query frame as shown in Fig. 3(d). Note that exact execution times are not decided at this point, because the duration of each motion is not known until motion clips are searched from the database and automatic motions are added later.

The motion search is applied for each query frame. In addition, when it is necessary, automatic motions are inserted before the motion. Finally, motion clips and their execution timings are passed to the motion synthesis module as a motion timetable, as shown in Fig. 3(e).

The motion synthesis generates an animation by smoothly connecting given motion clips. The interactions between characters and between a character and objects are handled by this module based on the information that the motion clips have.

The scene information contains characters and objects and their initial postures. Currently, our system supposes that the scene information is provided by the user.

The scope of this paper is the components in the dotted box in Fig. 2. There are many tools for syntax analysis that can be used with our system. The Stanford parser [15] is used for our implementation. For motion synthesis, our system uses an external animation system [16]. The system generates continuous motions from given motion clips and their execution timings. The system determines an appropriate synthesis method for each transition based on the constraints between the foot and the ground during motions. Alternatively, another commercial animation system such as MotionBuilder, Maya, 3ds Max, etc. can be used.

IV. Motion Database

In this section, we describe the representation of motion data. We first explain the case frame that is used in natural language processing. Then, we explain our motion frame, which is inspired by the case frame. We also describe our database of characters and motions.

A. *Case Frame*

The idea of a case frame was proposed by Fillmore [5]. A case frame represents the role of a verb. Each case of a case frame is a phrase that represents an aspect of the verb. Typically a case frame has the following cases:

- Agent: the person who performs the motion.
- Experiencer: the person who experiences something.
- Object: the object that an effect is caused to during the motion.
- Instrument: the object that causes an effect during the motion.
- Source: the source or origin of the motion.
- Goal: the goal or target of the motion.
- Time: the time when the motion is performed.
- Location: the location where the motion is performed.

Each case needs to be a specific type of entity. Some cases are mandatory for some verbs. A verb that has different roles depending on context has multiple case frames.

In general, natural language processing systems, a procedure to select a case frame for an input text is as follows. First, based on the types and dependency of phrases in the analyzed tree, candidate cases of each phrase are determined. By searching for case frames that match the candidate cases, the most appropriate case frame and all its cases are determined.

The case frame is a good way to extract and represent the meanings of texts. The case frame is widely used in many research papers such as [4][8]. However, the case frame is not suitable for representation of motion data for animation. From the view point of motion representation, each case has different roles depending on case frames. For example, the "object" case of a case frame could be an object that the character uses or another character that the character's motion causes an effect on. Moreover, the case frame does not contain information about postures and contact positions, which are important for selecting motions.

B. *Motion Frame*

We propose a motion frame, which contains the information about a motion clip. The motion frame is inspired by the case frame. However, we define the items of the motion frame based on importance when we search for a motion according to input texts.

There are many kinds of verbs in general English. However, our system handles only action verbs that involve a physical motion, in other words, verbs that can be visualized as an animation. Other kinds of verbs such as non-action verbs (e.g., "think", "believe") or state verbs (e.g., "know", "exist") are ignored in our system, because they are difficult to represent by a motion clip. Action verbs are categorized into intransitive, transitive, and ditransitive verbs. Intransitive verbs involve no other object (e.g., "he runs"). Transitive verbs include one target object/character/position (e.g., "he opens the door", "he hits her", "he walks to the door"). Ditransitive verbs include two target objects (e.g., "he gives her the book", "he cuts the bread with a knife"). For distractive verbs, one of the two target objects should be the object that the character possesses. We call such objects "instruments". Therefore, action verbs have at most one "target" object /character/position and at most one "instrument" object. We use them as items of a motion frame instead of cases in a case frame. In addition, contact position is used to select a motion that fits the environment and previous motions.

The items of the motion frame are as follows. An example of a motion frame is shown in Fig. 4. Note that some items may not have any value depending on the motion.

- Agent M_{agent_ref} : The reference to the character in the database who performs the motion.

Item	Value
Agent	human
Motion	take, pick up, get
Instrument	NULL
Target	appropriate size and weight ranges
Contact Position	hand position of contact
Initial Posture	standing
Adverbs	slowly

Figure 4. Example motion frame of "taking-an-object"

- Names of motion $M_{motion\_strings}$: The set of verbs that represent the motion. When a verb in the input text matches one of the motion names, the motion frame will be a candidate for the verb. In order to handle ambiguity, a motion frame may have multiple names. For example, a "taking-an-object" motion may have "take" and "pick up" as its names.
- Instrument $M_{instrument\_ref}$, $M_{instrument\_params}$: The object that the character uses in the motion. This is either a reference to an object in the database $M_{instrument\_ref}$ or the size and weight ranges of an object $M_{instrument\_params}$. If the motion requires a specific object such as "cutting with a knife", the object should be specified as a reference to the instrument. Otherwise abstract conditions of an object are specified. For example, if the motion is "poking something with a long object", then appropriate size and weight ranges of the object are specified.
- Target: The reference to an object M_{target_ref} or the size and weight ranges M_{target_params} are specified in the same way as the instrument. If the target is a character, the reference to the character is specified in M_{target_ref} .
- Contact position $M_{contact\_virtical}$, $M_{contact\_horizontal}$: the position of the end-effector when it contacts the target. Vertical and horizontal positions are handled differently. Because the horizontal position can be adjusted by lateral movement (see Section VI.C), vertical position is more important for motion selection. The contact position is automatically computed from the contact information (see Section IV.C). For example, if multiple "taking an object" motions are in the database and an input text "he takes the bottle on the ground" is given, then based on the position of the bottle, the appropriate taking motion (e.g., "taking an object with squatting") will be selected.
- Initial posture $M_{initial\_posture\_flag}$: the character's posture when the motion begins. Currently, it is represented as one of three states: standing, sitting, or laying down. The initial posture is used to select a motion that matches the terminal posture of the previous motion. In cases where no such motion is in the database, an automatic changing posture motion will be added (see Section VI.C).
- Adverbs $M_{adverb\_strings}$: The set of adverbs represent the style of the motion such as "slowly" or "happily".

Each item of motion frames must be specified by a user. However, this is not such a difficult task for users. For each motion frame (each motion clip), the user is asked to specify the agent, verbs, target, and instrument. The agent is selected from the character database. For the target and instrument, it is either an appropriate object or agent that is selected from the database or the size and weight range of an object. When the motion involves a specific object (e.g., "cutting with a sword"), the object should be selected.

Otherwise, object conditions are specified (e.g., "lifting up a light object using one hand"). The contact position is automatically computed form the motion and its contact information (see Section IV.C). The initial posture is also automatically computed from the motion clip. As a result, specifying the items of a motion frame is very easy.

C. Motion Data

Our system supposes that each motion is short and simple. A complex motion is difficult to represent by a motion frame. If a user wants to add a long motion to the database, the motion should be divided into pieces.

Some motions involve an interaction with an object or a character. This information is very important for generating animation and for selecting motions. Therefore, it is specified on the motion frame. The contact information consists of the contact type (hold, release, or hit), contact time (local time in the motion clip) and the end-effector (e.g., right hand). This information is also necessary for generating animation in the motion synthesis module (see Section V.C).

Some motions that interact with another character cause the reaction of the other character (e.g., "Jack hits Neo. Neo falls"). Usually such cooperative motions are captured or created at the same time but are stored as separate motion clips. In our system, such cooperative motions are specified on the motion frame. If a motion has cooperative motions and no cooperative motion is indicated in the input text, the system automatically executes a cooperative motion (see Section VI.C). In addition, when two cooperative motions include physical contact, the timings and the initial positions of these motions are coordinated (see Section VI.A).

D. Character and Motion Database

We use an object-oriented framework for the character and motion database. As shown in Fig. 5, each character is considered to be an object that has various motions as its methods. A character is inherited from a base character. A motion of the base character can be overridden by another motion. The motions that are not overridden are used as the motions for the derivative character. In this way, the hierarchy of characters and their motions are efficiently managed.

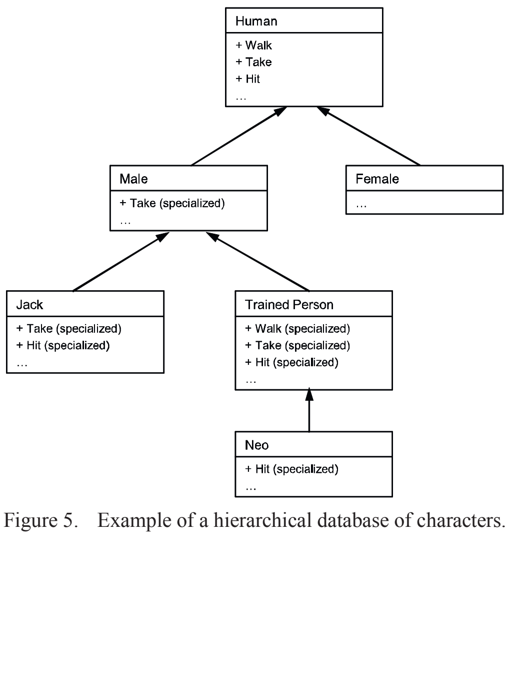


Figure 5. Example of a hierarchical database of characters.

When a user wants to create a new character, they can simply add the new character that is inherited from a base character in the database and add character-specific motions to that character. Even if they do not have many new motions for the new character, the motions of the base characters are used. In this way, users can add new characters very easily.

The database can be implemented in various ways. If the characters and motions are implemented using an object-oriented programming language (e.g., C++ or Java), we would represent motions as objects rather than methods and implement a mechanism of motion inheritance on the character class, because it is practically difficult to handle motions as methods using such programming languages.

V. MOTION SEARCH

In this section, we explain how to search for an appropriate motion for each verb in an input text. Handling of multiple verbs and motions is explained in the next section.

A. Query Frame

To select a motion that matches an input text, we use a query frame, which has the same items as the motion frame, and its items are determined by analyzing the syntax tree of the input text (see Fig. 3(b)). Scene information is also used to determine some items.

Although natural language processing techniques have advanced in recent years, it is still a challenge to understand general texts, because it requires not only language processing but also a large knowledge of the world. However, our system is supposed to take script-like text and only motion-related descriptions in the text matter. This makes the natural language analysis much easier than general natural language processing systems such as machine translation or summarization systems. Moreover, because scene information, such as characters and objects, is given in advance, we do not need the same large dictionary required by general natural language processing systems.

As explained in Section IV.B, unlike generic semantic analysis, motion searches only need a target and an instrument for each verb. Therefore, we determine these by applying the following rules for each verb in an input text.

- If a noun represents a character in the scene and the verb is dependent on the noun, the character is considered as the agent (subject) of the query frame Q_{agent_ref}.
- If two nouns are dependent on the subject that the verb is related to, they are considered as the target Q_{target_ref} and the instrument $Q_{instrument\_ref}$. (e.g., In "Jack gives Neo the book", "Neo" is the target and "the book" is the instrument.)
- If only one noun is dependent on the subject, it is considered as the target Q_{target_ref}.
- If a preposition phrase (e.g., "to Neo") is dependent on the subject, it is considered as the target Q_{target_ref} or the instrument $Q_{instrument\_ref}$ depending on the preposition. If the preposition is "with" and the noun in the phrase represents an

object, the object is used as the instrument. Otherwise, the noun is used as the target.

- If the character is holding an object, the object is also used as $Q_{instrument\_ref}$, even if it is not specified in the input text.

After the names of the target and instrument are determined, we obtain the reference or value of each item from the scene information. We suppose that the characters or objects in input texts always exist in the scene. Therefore, unlike general semantic analysis, by looking up the scene information all nouns in input texts are determined.

The target character or object that is indicated in the input text is searched from the scene information and the reference and position are set to the query frame. When the target is a character and a body part is indicated in the text such as "She hit him in the head", the reference and position of the body part is set. When the target is an object in the scene, the target size and weight are set in the query frame. The instrument object that is indicated in the input text is also set to the query frame.

B. *Evaluation of Motion Frame*

Based on the query frame from an input text, a motion frame that best matches the query frame is searched from the database. The motion search is done in three steps.

In the first step, all candidate motion frames in which the motion name and agent match the query frame are selected from the database. Any motion frames that have the agent character or its base characters can be candidates.

In the second step, the motion frames whose items do not match the query frame are excluded from the candidates. If the query frame has a target M_{target_ref}, or M_{target_params} and/or an instrument $M_{instrument\_ref}$, or $M_{instrument\_params}$ but a motion frame does not, then it is excluded. Moreover, if a motion frame has target parameters, instrument parameters, or the vertical contact position, and the values of the query frame exceed the specified ranges, then that motion frame is also excluded.

In the third step, all candidate motion frames are evaluated based on the similarity between the motion frame and the query frame items using the following equation:

$$\begin{aligned} E = {} & w_0 R\left(M_{target\_params}, Q_{target\_params}\right) \\ & + w_1 R\left(M_{instrument\_params}, Q_{indtrument\_params}\right) \\ & + w_2 D\left(M_{contact\_virtical}, Q_{contact\_virtical}\right) \\ & + w_3 D\left(M_{contact\_horizontal}, Q_{contact\_horizontal}\right) \\ & + w_4 F\left(M_{initial\_posture\_flag}, Q_{initial\_posture\_flag}\right) \\ & + w_5 A\left(M_{adverbs}, Q_{adverbs}\right) \\ & + w_6 H\left(M_{agent\_ref}, Q_{agent\_ref}\right) \end{aligned} \tag{1}$$

where $R(M,Q), D(M,Q), F(M,Q), A(M,Q), H(M,Q)$ are the functions that compute normalized distance (0.0~1.0) between size and weight parameters, contact positions, posture flags, adverbs, and hierarchical positions, respectively. The distances between the size and weight range of the motion frame and the object size and weight of the query frame are computed so that the distance becomes zero when the values are at the center of the range and the distance becomes one when the values are at the edge of the range. The distances between posture flags and adverbs are computed so that the distance is zero when they match and otherwise the distance is one. The distance between hierarchical positions of the characters is computed from the number of inheritances between them (see Fig 5). The candidate motion frame whose evaluation is the smallest will be selected and used for animation. $w_0 \sim w_6$ are weight parameters. They can be set for each motion frame in the case that some items are important for the motion. In our current experiments we use 1.0 for all weights of all motions.

C. *Motion Modification*

The motion clip of the selected motion frame is used for animation. However, even if the closest motion frame is selected, the contact position may not exactly match the query frame. In that case, the motion clip is modified using inverse kinematics. The posture of the character during the motion is modified so that the contact position of the end-effector (e.g., hand) matches the target position in the query frame.

When the character is far from the target, changing the end-effector position is not enough. In addition, when the character executes the selected motion it may need to first take an instrumental object or change its posture (e.g., standing up). These cases are handled by adding automatic motions before the selected motion instead of modifying the selected motion. Automatic motions are explained in Section VI.C.

VI. MOTION SCHEDULING

In this section, we explain how our system handles multiple motions from an input text. Basically, the system searches for a motion for each verb in the input text. However, in order to make an animation, the execution timing of each motion must also be determined. Moreover, the continuity of motions should be considered. For example, when a character makes contact with an object in the scene, the character must first move close to the object. Our system takes care of this kind of continuity of motions.

When multiple characters perform multiple motions the motions should be scheduled. However, an exact execution time for each motion is not usually specified in the input text. In order to determine the motion schedule, we need information about the motions such as duration and contact information.

Our motion schedule works as follows. First, temporal constraints are extracted from input texts in addition to query frames (Section VI.A). Second, query frames are roughly scheduled based on the temporal constraints (Section VI. B). Note that at this point, only process orders of query frames are determined. Finally, by searching for a motion frame that matches each query frame in order of process, the execution timing of each motion is determined. When automatic motions are required to be executed before a motion, they are added incrementally (Section VI.C). By repeating this process for all query frames, the motion clips and their execution timings are determined.

A. *Temporal Constraints*

Temporal constraints are extracted from input texts. The types of temporal constraint are serial execution or synchronized execution between two verbs. A serial execution constraint has the execution order of two motions. A synchronized execution constraint has relative execution timing. Temporal constraints are generated from a syntax tree as follows:

1. For all pairs of sequential verbs in the input text, serial execution constraints are assigned. For example, when the input text "Jack walks in the room. Neo stands up." is given to the system, a serial execution section constraint (Jack, walk) to (Neo, stands up) is generated.
2. When a word that indicates a reverse order exists in the input text (e.g., "after"), the order of the serial execution constraint is reversed. If a serial execution constraint is already created, the old constraint is overridden. For example, when the input text "Jack walks in the room after Neo stands up." is given to the system, a serial execution constraint (Neo, stands up) to (Jack, walk) is generated.
3. When a word that indicates synchronization exists in the input text (e.g., "at the same time" or "while"), a synchronized execution constraint is added. If there is a conflicting constraint, it is overridden. For example, when the input text "Jack walks in the room. At the same time, Neo stands up." is given to the system, a synchronized execution constraint (Neo, stands up) and (Jack, walk) is generated. The relative timings between two motions are set to zero so that they start at the same time.
4. When the motions of two characters are cooperative motions and they include contact with each other, a synchronized execution constraint is added and the relative execution timings of the two motions are determined based on their contact information (Section 4.3). For example, when the input text "Jack hits Neo. Neo falls" is given to the system, a synchronized execution constraint (Jack, hit) and (Neo, fall) is generated. At this point, the relative timings are not set. They will be set based on the contact times in the searched motion data, when the motions are searched later.

B. *Scheduling Query Frames*

Based on temporal constraints, the query frames are scheduled roughly at first. After that, the process order of all query frames (verbs) is determined. For motions that have a synchronized execution constraint, their process orders are temporarily set as one of them being processed first. The exact timings of all query frames are determined in the process order.

For each query frame, a motion clip is searched from the database as explained in Section V.B. Before searching each motion, the scene condition is set to the time when the motion is executed because the selected motion may change depending on the position of the character or object that the motion involves. The execution timing of the motion is determined based on the duration of the selected motion. The next motion is started just after the previous motion is finished if they have a serial execution constraint. If they have a synchronized executing constraint, their execution timings are determined based on the contact timings of the selected motions.

This process is repeated from the first motion to the last. When multiple query frames are synchronized based on the temporal constraints, the motions for all query frames are searched and their execution timings are delayed until all constraints are satisfied.

C. *Automatic Motions*

During the motion scheduling and motion search, a searched motion can sometimes not be executed. In that case, automatic motions are generated and added before the searched motion. As explained earlier, the purpose of our system is to reuse motion data without complex motion planning which may require additional programming for each motion. Therefore, our system deals with minimum automatic motions. The additional motions are also selected from the database. Therefore, each character is easily customized by adding specific kinds of motion to the database without adding any rules or modules.

If a motion includes interaction with another character or an object in the scene (i.e., a query frame contains a target object or character), the character has to be in the right place for contact with the object or character. If not, the system automatically adds motions for the character to move to the right place using simple rules. The system adds 'turn to the target', then 'step' or 'walk', and 'turn to the target' in this order, if they are judged to be necessary by the character's position and orientation.

When a character uses an instrument in a motion (i.e., a query frame contains an instrument and the character does not hold it), the character should pick up the instrument object before they use it. When a motion to take the instrument is not explicit in the input text, a 'take' motion is selected from the database. When the character is away from the instrument, locomotive motions are also added before the taking motion.

For motion searches, if there is no candidate motion whose initial posture matches the terminal posture of the previous motion (i.e., the initial posture of a query frame does not match any of the candidate motion frames), a changing posture motion such as standing up is added. In this case, all motions that include a state change will be candidate motions.

As explained in Section VI.A, when a motion involves interaction with another character, a cooperative motion of the other character follows. When a selected motion frame has cooperative motions and any of them are not indicated in the input text, the default cooperative motion and a temporal constraint of the motion frame are automatically added.

VII. Experiment and Discussion

We have implemented our method and motion database. Currently, the system has six characters as shown in Fig. 5 and about 50 motions that are collected from a commercially available motion capture library. We have tested our system with some short sentences and found that an appropriate motion was selected from each sentence even though the same verb is used in different sentences. An example of the generated animation is available at author's web site (http://www.cg.ces.kyutech.ac.jp/).

In order to evaluate our framework, we tested it with a published movie script (The Matrix, 1999). Because our motion database does not yet have enough data, we checked if our system could handle the descriptions in the movie script and output appropriate query frames. There were about 830 actions (verbs) in the script. We found that about 78% of them were processed by our system without any problem. However, 8% required additional language processes such as one verb representing multiple motions (e.g., "take A and B", "do A twice", or "they look each other"), pronouns (e.g., "they", "it", or "everyone"), infinitives (e.g., "he try to stand up") and passive verbs. 5% were verbs that cannot be represented by a motion (e.g., "miss" in "he shoots her and misses", vague representation such as "he stares into the darkness"). 5% were verbs for representing initial states in the scene but not actions (e.g., "they are dead", "he stands"). 4% were actions representing locomotion but no specific position (e.g., "he walks away"). As we discuss later, a non-text-based interface is suitable for specifying initial states or positions of locomotion. There were also descriptions for motions of objects. Currently our system cannot handle object motions. However, it is possible to extend our system to handle them, because they are simpler than human motions.

The fundamental principle of our framework is to make use of motion data without requiring any additional motion specific rules. Currently, our system does not support high-level motion planning such as automatically dividing complex motion into small motions or path planning with object avoidance. Because we use simple rules for automatic locomotion, the resulting animations are not so natural. This can be solved by adding more motion data and some sophisticated modules that generate new motion from a number of motion data sources such as [14].

Our system supposes that scene information such as the positions of objects and characters is given by a user. The existing text-to-scene system [8] can be integrated with our system. However, specifying the positions using natural language can be harder than using a conventional mouse-based interface. So can specifying locomotion path. From a practical viewpoint, a hybrid of a text-based interface and a conventional interface might be more useful.

With our current system, if the user is not satisfied with or wants to change an output motion, they must change the input text and they cannot change the output motions directly. To address this, we are going to develop a natural language-based motion editing interface with which a user can change generated motions interactively by giving instructions to agents, as real directors do with actors.

VIII. CONCLUSION

We have proposed an animation system that generates animation from natural language text such as movie scripts or stories. Our future work includes the expansion of both the system and the motion database. Currently, making animation is very difficult, especially for nonprofessional creators. We believe that our system will solve this issue and provide many creators with a way to express their stories as animation.

REFERENCES

[1] Terry Winograd. Understanding Natural Language. Academic Press, 1972.

[2] N. Badler, R. Bindiganavale, J. Allbeck, W. Schuler, L. Zhao, and M. Palmer. "Parameterized action representation for virtual human agents", In Embodied Conversational Agents, pp. 256-284, 2000.

[3] R. Bindiganavale, W. Schuler, J. Allbeck, N. Badler, A. Joshi, and M. Palmer. "Dynamically altering agent behaviors using natural language instructions", In Proc. of Autonomous Agents 2000, pp. 293-300, 2000.

[4] Takenobu Tokunaga, Kotaro Funakoshi, and Hozumi Tanaka. "K2: animated agents that understand speech commands and perform actions", In Proc. of 8th Pacific Rim International Conference on Artificial Intelligence 2004, pp. 635-643, 2004.

[5] Charles J Fillmore. The case for case. In Universals in Linguistic Theory, pp. 1-88, 1968.

[6] Kaoru Sumi and Mizue Nagata. "Animated storytelling system via text", In Proc. of International Conference on Advances in Computer Entertainment Technology, 2006.

[7] Hiromi Baba, Tsukasa Noma, and Naoyuki Okada. "Visualization of temporal and spatial information in natural language descriptions", Transaction on Information and Systems, E79-D(5), pp. 591-599, 1996.

[8] Bob Coyne and Richard Sproat. "Wordseye: an automatic text-to-scene conversion system", In Proc. of SIGGRAPH 2001, pp. 487-496, 2000.

[9] Ken Perlin, and Athomas Goldberg, "Improv: A System for Scripting Interactive Actors in Virtual Worlds", In Proc. of SIGGRAPH '96 Proceedings, pp. 205-216, 1996.

[10] Matthew J. Conway. Alice: Easy-to-Learn 3D Scripting for Novices, PhD Dissertation, University of Virginia, 1997.

[11] Masaki Hayashi, Hirotada Ueda, Tsuneya Kurihara, Michiaki Yasumura, "TVML (TV program Making Language) - automatic TV program generation from text-based script –", In Proc. of Imagina '99, pp. 84-89, 1999.

[12] Hyunju Shim, Bo Gyeong Kang, "CAMEO - Camera, audio and motion with emotion orchestration for immersive cinematography", In Proc. of International Conference on Advances in Computer Entertainment Technology (ACE) 2008, pp. 115-118, 2008.

[13] C. Rose, M. F. Cohen, and B. Bodenheimer. "Verbs and adverbs: Multidimensional motion interpolation", IEEE Computer Graphics and Applications, vol. 18, no. 5, pp. 32-40, 1998.

[14] Lucas Kovar and Michael Gleicher. "Automated extraction and parameterization of motions in large data sets", ACM Transactions on Graphics, vol. 23, no. 3, pp. 559-568, 2004.

[15] Dan Klein and Christopher D. Manning. "Fast exact inference with a factored model for natural language parsing", In Advances in Neural Information Processing Systems 15 (NIPS 2002), pp. 3-10, 2003.

[16] Masaki Oshita. "Smart motion synthesis", Computer Graphics Forum, vol. 27, no. 7, pp. 1909-1918, 2008.