Single RGB-D Camera Teleoperation for General Robotic Manipulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: We propose a teleoperation system that uses a single RGB-D camera as the human motion capture device. Our system can perform general manipulation tasks such as cloth folding, hammering and 3mm clearance peg in hole. We propose the use of non-Cartesian oblique coordinate frame, dynamic motion scaling and reposition of operator frames to increase the flexibility of our teleoperation system. We hypothesize that lowering the barrier of entry to teleoperation will allow for wider deployment of supervised autonomy system, which will in turn generates realistic datasets that unlock the potential of machine learning for robotic manipulation. Demo of our systems are available online.

Keywords: Teleoperation, Robot Manipulation, RGB-D Sensor

1 Introduction

In machine learning, training data should approximate the test distribution. Not surprisingly, a key ingredient in the successes of learning algorithms is diverse and realistic datasets. The collection of such datasets is made possible by existing platforms whose main purpose is not dataset collection. For example, without the internet, it would have been prohibitively costly to collect the 22 million images in ImageNet at the same level of visual diversity. Similarly in NLP, language models are trained from large corpus crawled from the internet [1], ensuring syntax and semantic diversity. In summary, existing data collection platforms have enabled rapid progress in both CV and NLP.

A key property of these platforms is that the collection of diverse datasets is only a by-product of deploying socially useful services. The immediate values to society offered by these platforms offsets the cost of large-scale data collection. From this perspective, a key bottleneck preventing the progress of learning for manipulation is the lack of a data acquisition mechanism in the wild, such as in actual homes and offices. In fact, we are faced with a chicken-and-egg problem. From a learning perspective, developing generalizable manipulation robots relies on large-scale and realistic datasets. However, collecting such datasets requires deploying robots to perform tasks in the real world. How can we break this cyclic dependency? How do we develop semi-autonomous robots that can perform useful tasks in realistic environment conditions, thereby generating realistic datasets as a by-product?

We hypothesize that supervised autonomy [2, 3] through remote teleoperation will play a key role. In supervised autonomy, the robots do not need to generalize well to all operating conditions. A human-in-the-loop can take over control of the robot when the autonomous system is expected to fail. Current teleoperation interfaces usually require special hardware that is not readily available, such as Virtual Reality headset controller or game controller. We are interested in reducing the barrier of entry to teleoperation. Specifically, we are interested in the following question:

Given only one RGB-D camera as the motion capture device, to what extent can a trained human operator command a robotic arm to perform complex manipulation tasks?

Figure 1 illustrates the tasks we perform with a single camera. Our contributions are summarized as:

- A teleoperation system for manipulation which uses a single RGB-D camera for motion capture.
- Novel application of non-Cartesian oblique coordinate frame to define the operator frame, dynamic motion scaling, and repositioning of operator frame for more intuitive control.

https://sites.google.com/view/manipulation-teleop-with-rgbd

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.
We propose a teleoperation system which uses a single RGB-D camera to capture human motion. The human operator does not have access to any form of tactile feedback. Tasks (a)-(g) are evaluated on a real robot arm. Dual arm task (h) is performed in simulation.

- Demonstration that our system can successfully accomplish complex manipulation tasks.

To the best of our knowledge, in teleoperation, we are the first to demonstrate the successful completion of several complex manipulation tasks while using only a single RGB-D camera as a sensing device. We also show preliminary results on removing depth sensing while still allowing the operator to command the robot by performing monocular depth estimation.

We emphasize the mutually beneficial relationship between ours and orthogonal research directions, such as developing physically accurate simulator [4] or RL algorithms that learn from offline data [5, 6]. For example, if we are successful in reducing the barrier to entry of teleoperation and in turn enable large-scale data collection in the wild for robotic manipulation, the resulting datasets can possibly be used to improve the fidelity of physical simulators [7, 8, 9].

The organization of the paper is as follows. We discuss related work in section 2, section 3 and section 4 illustrate the hardware and software design of our teleoperation system. section 5 describes our experimental results and we end the paper in section 6.

2 Related Works

Manipulation Dataset Collection The need for large-scale datasets is well recognized within the robot learning community. There have been various efforts to collect real robot data in lab settings [10, 11, 12, 13] or to develop low-cost control interface for simple tasks such as grasping [14]. We pursue a different philosophy and aim to lower the cost of teleoperation to allow for real world deployment. There have also been efforts to scale up data collection by taking advantage of human videos [15] or simulations [16, 17, 18], which face their own domain shift challenges when transferred to real robots. [19] proposes to use phone’s IMU as the motion capture device to control a robot arm. We demonstrate that our system can use a single RGB-D camera to control two robot arms and perform a task that necessitates two arms for successful execution. We also test our systems on tasks that require distinctively different skills to accomplish.

Marker-based motion capture and teleoperation. Motion capture system tracks human motion and is the key ingredient for robot teleoperation. Pioneering motion capture works use special-purpose markers or gloves to track the 2D or 3D positions of critical points [20, 21, 22, 23, 24]. The tracking results can be further combined with Virtual Reality devices for better feedback to human operators [25]. The marker-based solution often suffers from self-occlusion, thus redundant markers are often placed on the gloves to improve tracking consistency. Additional hardware, such as inertia measurement unit [26, 27], electromyography [28], and haptic devices [29], can be used to further improve tracking performance. Tactile feedback [30, 31, 32] to the human operator provided by specially-design gloves also improves the controllability of the teleoperated robots. While these systems offer good teleoperation performance, they require the human operator to have access to hardware that are not widely available, and are thus difficult to deploy widely.
Vision-based motion capture and teleoperation. Due to their less stringent hardware requirement, vision-based markerless hand tracking has received significant attention from the teleoperation community. [33, 34, 35] tracks the hand poses via monocular camera or depth sensor. A hand model can be also be used to improve tracking performance [36, 37]. These works use the hand pose tracking results to control a robot arm with parallel gripper for simple tasks, e.g. pick and place. In contrast, our teleoperation system can perform complex tasks such as cutting and cloth folding. DexPilot [38] also demonstrates successful completion of complex tasks using a multi-finger hand. To control the multi-finger hand, DexPilot requires high tracking accuracy of most finger joints. The human operators thus require access to four RGB-D cameras with well-calibrated extrinsic to register the complete points of human hand. The operators are also constrained to move within a limited workspace above a black-clothed table to command the robot. The human operator in our teleoperation system can perform complex tasks while using only 1 RGB-D camera without extrinsic calibration. We however recognize that the multi-fingered setting studied in DexPilot is significantly more challenging than our parallel yaw setting.

Hand pose estimation. Both marker-based and vision-based teleoperation system make use of hand pose estimation techniques, which have advanced rapidly due to the integration of deep learning and hand models. Motivated by the success in body pose estimating of blend skinning techniques [39, 40], MANO [41] models shape variation from hand scans by learning pose dependent blend shapes and represents the hand as a combination of shape parameters and pose parameters. [42, 43, 44] also adopt linear blend skinning model for hand pose estimation and tracking. Our teleoperation system illustrates that state-of-the-art vision-based hand pose estimation techniques are accurate enough to allow for the completion of tasks that require mm-level precision, such as peg-in-hole.

3 Hardware System

Figure 2 illustrates the schematic view of our hardware system. The operator observes the current status of robot workspace, such as the state of the robot and the objects, via a display device in real-time. The operator can move their hands to control the motion of robot arm and grippers. The stream of human motion as captured by the RGB-D camera is processed by an edge computing device to estimate the hand poses. The edge computing device then sends a position command to the PC in the robot space, which is then executed by the robot controller.

There are two cameras in the robot space: a wrist-mounted camera to capture detailed interaction between the grippers and objects, and a tripod-mounted camera to observe the full view of robot workspace. The RGB video from the two cameras are streamed to the display device in the operator space. We use the built-in controller provided by the manufacturer. It already takes count of self collision and collision with the mounting table thus no further motion planning is required.

4 Software System

Figure 3 illustrates the inputs and outputs of the software components in our teleoperation system. The three components are: hand pose estimator, motion correspondence, and robot controller. The following sub-sections describe them in more details. We use the distance between right thumb and index tips to determine when the gripper should close [37, 36]. We will use monogram notation in the following subsection to describe frame transformation [45].
4.1 Hand Pose Estimator

The hand pose estimator takes as input RGB-D frames and outputs 21 2d positions of keypoints in image space and the wrist pose in camera space for each hand. Given the RGB frame, we use MediaPipe [46] to estimate the 2d positions of hand keypoints. Given the keypoints, we compute axis-aligned bounding-box to crop the image around hand region. The cropped images are then input into a hand pose regressor proposed by Frankmocap [44] to predict the orientation of the wrist in camera space. We also obtain the 3d positions of the wrist in camera space by deprojecting from the depth map. The hand pose estimator is implemented as a ROS [47] node with publish rate of 10Hz.

4.2 Motion Correspondence

Why do we need operator frame? An issue that frequently appears in teleoperation is viewpoint mismatch. Human operators naturally interpret their motion commands to the robots with respect to an egocentric frame. If the base frame used for controlling the robots is not aligned with the egocentric frame, the operators need to understand the relationship between their egocentric frames and the robots’ base frame and perform mental rotation during teleoperation [48, 49, 50]. Such calculation severely impedes the ease of teleoperation. To solve the viewpoint mismatch problem, our system allows the operator to define an egocentric frame that naturally aligns with the base frame used for controlling the robots.

Constructing the Cartesian operator frame. The base frame in our case is the robot wrist camera frame \{wc\} as illustrated in Figure 4. Our system defines the wrist camera frame \{wc\} prior to teleoperation and keeps it fixed. For each axis of the frame \{wc\}, the system asks the operator through a GUI to move their right wrist in a direction such that if the operator moves their wrist in the same direction during teleoperation, the end effector will move along the corresponding axis of the wrist camera frame. As such, the end effector base frame is aligned with the operator egocentric frame, eliminating the viewpoint mismatch issue.

Let \{c\} represents the operator space camera frame. During this procedure, the perception system records the positions of the right wrist with respect to (wrt) the camera frame \{c\}. Our system thus obtains three sets of wrist positions, one for each axis of the wrist camera frame \{wc\}. To define the pose of the operator frame wrt the camera frame, we first perform RANSAC line fitting for each set of positions and then normalize the resulting lines to unit length. Let the normalized line corresponding to the \(x\)-axis of \{wc\} be \(n_x\). Similarly for \(n_y, n_z\). To find the orientation of the operator frame, we project the matrix \([n_x, n_y, n_z] \in \mathbb{R}^3\times3\) to the closest orthogonal matrix \(R \in SO(3)\). To find the origin of the operator frame, we compute the point \(p\) closest to the three fitted lines \(n_x, n_y, n_z\) in a least square sense. Let \{cf\} denotes the Cartesian operator frame. The procedure above produces \(p\) and \(R\), which together form the pose \(cTc^f \in SE(3)\) of the Cartesian operator frame wrt the camera frame \{c\}.

Obtaining motion correspondence between operator and robot. Given the pose of the Cartesian operator frame \(cTc^f\), we now describe how to compute the desired pose of the end effector given the current pose of the right hand wrist during teleoperation. Let \{wr\} denotes the frame attached to the operator right hand wrist. Let \{w\}, \{e\} denote the robot world and end effector frames respectively. Given \(cTwr\), we need to compute \(wT^e_{desired}\). We compute the desired position and orientation separately. Given the current position of the right hand wrist \(c^{pwr}\), we express it wrt the Cartesian operator frame: \(c^{pwr} = cT_{desired} \cdot c^{pwr}\).
We then treat a scaled version of \( cf \cdot p^w \) to be the desired end effector position wrt the wrist camera frame \( \{ wcf \} \): \( wcf \cdot p^e = \alpha \cdot cf \cdot p^w \).

Since our system defines the frame \( \{ wcf \} \) prior to teleoperation and keeps the pose of \( \{ wcf \} \) wrt to the world frame fixed during teleoperation, we can find the desired end effector position with:

\[
\begin{align*}
    w^p_{desired} &= \underbrace{w^T_{wcf}}_{\text{rotation}} \cdot wcf \cdot p^e
\end{align*}
\]

\( \alpha \) is a scalar to control the scaling between human and robot motion. If \( \alpha = 1 \), the end effector moves the same distance as the operator wrist. The appropriate value of \( \alpha \) depends on the task. For task like peg in hole, we might want to use a smaller \( \alpha \) to allow for more precise robot control.

Similarly, given the orientation of the right hand wrist wrt camera frame \( c^R_{w} \), we can represent it wrt the Cartesian operator frame and then use \( \{ wcf \} \) as a “bridge” to find the desired orientation of the end effector wrt the world frame \( w^R_{desired} \). We elaborate on the rotational transformation procedure in the supplementary.

**Oblique operator frame.** To construct each axis of the Cartesian operator frame, our system asks the operator through a GUI to move their right wrist in a direction such that if the operator moves their wrist in the same direction during teleoperation, the end effector will move along the same axis of the wrist camera frame. However, such promise by the system is often not possible because the 3 best fit lines to the operator wrist positions are usually not orthogonal. This necessitates the projection step to SO3 as discussed above. However, projecting the matrix whose columns are the best-fit lines to SO3 implies that as the operator moves their wrist in the same direction as one of the best fit lines, the robot end effector might not move along the corresponding axes of the wrist camera frame. In other words, if the human repeats the same motion during the operator frame definition procedure and during teleoperation, the robot end effector might move in different directions. We found such non-repeatability of control commands unintuitive and therefore introduce the use of non-Cartesian oblique coordinate frame to ensure motion repeatability.

An oblique frame is a frame whose axes are not orthogonal [51]. The difference between Cartesian and oblique coordinate frames are further illustrated in Figure 5. Given an arbitrary point in space, to find the measure number of a point wrt to an axis of an oblique coordinate frame, we project the point onto the axis along the direction parallel to the hyper-plane defined by the remaining axes. Let \( \{ of \} \) denotes the oblique operator frame. We next explain how to obtain the oblique frame \( \{ of \} \) from the cartesian frame \( \{ cf \} \). We express the axes of the oblique coordinate frame as vector in the Cartesian operator frame. To find the x-axis of the oblique coordinate frame, we find the vector passing through the origin of the Cartesian operator frame and is parallel to the best fit lines of the 3d wrist positions obtained when defining the x-axis of the Cartesian operator frame. Denote the x-axis of the oblique frame in the Cartesian frame by \( cf \cdot o_{f x} \in \mathbb{R}^3 \). Similarly, we can obtain the y-axis and z-axis of the oblique frame \( cf \cdot o_{f y}, cf \cdot o_{f z} \in \mathbb{R}^3 \). The normal to the hyperplane defined by the x and y axes of the oblique frame is \( cf \cdot n_{xy} = cf \cdot o_{f x} \times cf \cdot o_{f y} \). We can similarly find the normal to the hyperplane defined by the remaining pairs of axis \( cf \cdot n_{xz}, cf \cdot n_{yz} \). Having defined these quantities, the
computation to transform the 3d position of operator wrist to desired robot end effector position is:

1. \( cf_{p_{wr}} = c f_{Tc} \cdot c f_{p_{wr}} \)

2. \( of_{p_{wr}} = \left[ of_{f_{x}}, of_{f_{y}}, of_{f_{z}} \right] / \left[ c f_{n_{xz}}, c f_{n_{yz}} \right] \)

3. \( of_{p_{y}} = \left[ c f_{p_{wr}}, c f_{n_{xz}} \right] / \left[ c f_{o_{f_{x}}, c f_{n_{yt}}} \right] \)

4. \( of_{p_{z}} = \left[ c f_{p_{wr}}, c f_{n_{xy}} \right] / \left[ c f_{o_{f_{y}}, c f_{n_{zx}}} \right] \)

5. \( wcf_{p_{e}} = \alpha \cdot \left[ of_{p_{x}}, of_{p_{y}}, of_{p_{z}} \right] \in R^{3 \times 1} \)

6. \( w_{e_{desired}} = w_{e_{wc}} \cdot wc_{p_{e}} \)

### 4.3 Dynamic Motion Scaling

The translational motion scaling factor \( \alpha \) determines the distance the end effector moves in response to movement by the operator. A smaller \( \alpha \) allows for more precise control of the end effector. For example, if \( \alpha = 0.03 \), then the end effector will only move 3mm if the operator right wrist moves by 10cm. This allows the operator to precisely control the position of the end effector without having to precisely control their wrist movement. Picking good values for \( \alpha \) is thus important to successfully performing the tasks. However, good values for \( \alpha \) change across tasks and even within one trial of a task depending on task progression. For example, in the peg in hole task, when the robot has not grasped the peg, a high value of \( \alpha \) allows the operator to quickly command the robot to a good pre-grasp pose. However, when the peg is grasped and near the hole, a small value of \( \alpha \) is preferable.

Thus, in the one arm setting, we use the estimated left hand poses to dynamically adjust the value of \( \alpha \) during task execution. More concretely, if the y coordinate of the left hand wrist wrt the camera frame is below a certain threshold, we set \( \alpha \) to 0.02. Otherwise, we set \( \alpha \) to 0.3. In addition, if the distance in image space between the left hand thumb and pinky tips is below a threshold, we set \( \alpha \) to 0. If \( \alpha = 0 \), there is no robot motion. Whenever \( \alpha \) changes, we reinitialize the origin of the operator frame to be the current right wrist 3d position wrt camera frame. This allows the operator to reset their position when they are almost out of the field of view of the camera. Dynamic motion scaling is an extremely useful yet simple technique that the operator takes advantage of extensively for difficult tasks such as peg in hole.

## 5 Experiments

### 5.1 Task descriptions

The key question we investigate is whether given only a single RGB-D camera as the motion capture device, can an operator teleoperate the robot to perform complex manipulation tasks? To answer this question, we design a set of single arm manipulation tasks in real world or a dual arm coordination task in the simulator as shown in Figure 1. For the real robot experiments, we consider seven object manipulation tasks: pick and place, peg in hole, hammering, cutting fruit, folding cloth, cord untangling, and object storage in drawer. For the simulated experiment, we consider two-arm manipulation. We describe the tasks in more details below:

**Pick and Place** At the beginning of each episode, three typical objects from the YCB Dataset [52] are placed in random positions on a table. The robot needs to grasp the object and move at least 0.6 meters to reach the target position. The pick and place serves as the entry level manipulation task.
Pick and Peg in Cutting Folding Object Storage Cord

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Pick and Place</th>
<th>Peg in Hole</th>
<th>Hammering</th>
<th>Cutting Fruit</th>
<th>Folding Cloth</th>
<th>Object Storage in Drawer</th>
<th>Cord Untangling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>8.9 ± 1.3</td>
<td>9.7 ± 0.8</td>
<td>3.8 ± 0.8</td>
<td>3.9 ± 0.1</td>
<td>5.7 ± 0.7</td>
<td>7.0 ± 1.2</td>
<td>4.4 ± 1.1</td>
</tr>
</tbody>
</table>

Table 1: Time and std to complete each task (in minutes)

Peg in Hole Three different types of pegs and holes are used in this task, ranked by difficulty from easy to hard: cylinder, pentagon, and hexagon. When the pegs are fully inserted in the holes, the clearance between them is no more than 3 mm. We use this task to demonstrate the precision of our tele-operation system since a small position error will lead to a failure.

Hammering In this task, the robot first picks up a hammer with the gripper. The robot then needs to reach a small bench and hammer the wooden cylinder into the hole. This task evaluates whether the operator can grasp the hammer with a suitable pose and transmit force to the cylinder in the correct direction. Such tool manipulation tasks are of common interest in robotics community [53, 54].

Cutting Fruit Cutting fruit is another tool manipulation task. The robot uses a knife to slice a watermelon chunk into two pieces. Similar to hammering, cutting requires the robot to grasp the knife with the right pose and transmit force in the correct direction to the watermelon chunk.

Folding Cloth Given a piece of cloth initially lying flat on a table, the robot folds the cloth twice along the diagonal. This task demonstrates the ability of our system to manipulate deformable object.

Moving Large Container Two robot arms move a large container from one planar surface to another. Due to hardware issues, we demonstrate the successful completion of this task by our system in the simulator DRAKE [55]. This task tests the coordination of two arms where each hand of the operator controls each robot arm. The control the second robot arm, we apply the same motion correspondence procedure described in section 4 to the left hand wrist.

Cord Untangling One end of a cord is fixed. The remaining cord section is tangled to make a knot. The robot needs to untangle the cord by grasping and pulling the cord in the right direction.

Object Storage in Drawer This task requires executing four primitive manipulation skills: pull the drawer to open, pick an object, place it into the drawer, and close the drawer. The task also requires manipulating articulated object, a problem which recently receives significant attention [4, 56, 57]. For each tasks, the operator performs 3 evaluation trials after a period of practice, except for Pick and Place where we perform 5 trials and Moving Large Container where we only perform 1 trial. We highlight that the Cord Untangling and Object Storage in Drawer tasks were not seen by the operator before evaluation. The first time the operator interacts with the objects present in these two tasks is during evaluation. Such unseen tasks allow us to test the adaptability of the system and the operator to new tasks and operating conditions.

5.2 Results of using RGB-D camera to capture hand poses

For all the tasks, our teleoperation system successfully completes the evaluation trials. This fact is even more surprising for the Cord Untangling and Object Storage in Drawer tasks where the human has not practiced with the objects before the evaluation trials. Such successes on unseen tasks demonstrate the generality of our teleoperation system and can potentially allow human operators to remotely control robot in the wild in unseen conditions. The time to completion for each tasks are shown in Table 1. Please view video recording of task execution with the link in the abstract.

An interesting behavior we notice across tasks is failure recovery. For example in the first trial of the Cord Untangling task, the end effector became stuck under a table in the robot space. Without manual reset of the robot, the operator was able to unstuck the end effector by moving under the table.
and eventually complete the task. A common failure mode is closing the gripper without grasping the objects. This happens more frequently with small objects since neither the wrist camera or the side camera provide distinctive visual cues to the operator whether the object is between the two grippers. However, this failure mode is easy to recover from because the operator can retry the grasp.

In addition to failure recovery, the operator also discovers interesting manipulation strategy such as dynamic manipulation. In the Cord Untangling task when the end of the cord (with higher mass density) is stuck underneath itself, the operator quickly moves their wrist to induce a large position error, thus inducing a large acceleration of the robot, which unsticks the end of the cord due to inertia.

To ablate the benefit of the dynamic motion scaling scheme introduced in subsection 4.3, the operator performs the peg in hole task while using a constant translational scaling factor $\alpha$. For a range of $\alpha$ values in $0.03, 0.3, 0.5, 1.0$, the operator was not able to successfully accomplish the task when given 15 minutes to teleoperate the robot. For $\alpha = 0.03$, the operator can not command the robot to reach the peg before going out of the field of view of the operator space camera. For higher values of $\alpha \in 0.3, 0.5, 1.0$, the operator fails to control the robot precisely enough to insert the peg into the hole. The ability to dynamically modify the scaling factor $\alpha$ is most beneficial for precision manipulation tasks and less important for simple task, as illustrated in Figure 7.

5.3 Results of using RGB camera to capture hand poses

Using RGB camera instead of RGB-D camera is a promising extension to our method. The main challenge is to obtain accurate 3D positions of hand wrists without depth information. Monocular depth estimation is in general a very challenging problem. However, given that we only requires the 3D position of the right wrist, we were nevertheless able to obtain promising results. In addition, in our problem, the perception NN can have strong prior over the shape and size of the human hand, which improve the performance of 3D hand estimation over the general depth estimation problem.

To estimate 3D hand pose from RGB, we crop the image around the hand region and predict the weak perspective transformation scale $s_h$ using the model from FrankMocap [44]. The weak perspective transformation approximates the true camera projection model. It assumes that for a small object, such as human hand, the depth value of any point on this object is the same. If the assumption holds, the pixel coordinates from weak projection well approximate the real projection model. We can thus approximate the depth simply by $\frac{C f}{s_h}$, where $f$ is the focal length, $C$ is a scalar constant.

Using only RGB to capture hand pose allows the operator to successfully complete 1 out of 3 trials of the pick and place task. In one other trial, the operator can grasp and move the three objects, but ends up knocking one of the objects over due to noisy depth estimate. In the remaining trial, monocular depth estimation fails when the right hand index and thumb tips come close together to signal that the gripper should close. This violates the weak perspective assumption and leads to a non-trivial drop in the estimated depth. We had to stop the trial early. Please refers to the demo video linked to in the abstract and supplementary for more information.

6 Conclusion

We propose a manipulation teleoperation system that uses a single RGB-D camera as the human motion capture device. The operator and our system can perform various complex tasks, even perform well on unseen ones. In the future, we are excited to expand the scopes of tasks that can be performed by our manipulation systems.
References


