# *Introducing A Vision-Based Posture Tracking System Using the TMS320C40 DSP*

*APPLICATION BRIEF: SPRA191*

*Authors: Jitendra Malik, Professor of Computer Science*
*Ali Rahimi*
*Michael Shilman*

*Department of Computer Science*
*University of California at Berkeley*

*Digital Signal Processing Solutions*
*August 1997*

TEXAS INSTRUMENTS

## IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

## TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

# Contents

# Figures

# Introducing A Vision-Based Posture Tracking System Using the TMS320C40 DSP

## Abstract

The fields of telepresence and virtual reality have experienced a surge of activity in recent years. The goal of these fields is to place users in an immersive environment that behaves like the real world. To achieve this goal, fields require the use of next-generation input devices such as data gloves, head-mounted displays, and various other immersive peripherals. Posture capture and recognition offer a unified and more expressive means of interacting with computers.

This application brief presents a vision-based posture tracking system called *Predator*. Predator is a motion tracking system that allows a computer to unintrusively analyze the posture of the subject in real time. The package includes a vision processor based on the Texas Instruments (TI™) TMS320C40 digital signal processor (DSP) and includes a library of routines that makes it easy to write software that interfaces with Predator.

This document was an entry in the 1995 DSP Solutions Challenge, an annual contest organized by TI to encourage students from around the world to find innovative ways to use DSPs. For more information on the TI DSP Solutions Challenge, see TI's World Wide Web site at www.ti.com.

## Introduction

Our basic design goal was to build a real-time posture tracker that requires no setup on the part of the subject being tracked and can be used in almost any surrounding. We chose a design-based computer vision because it is the only feasible technology that allows us to track bodies while maintaining versatility and unintrusiveness. Furthermore, we have focused our design around techniques and algorithms that easily lend themselves to parallelization and can execute in real time on modern parallel architectures

We make a distinction between posture tracking and recognition. *Posture recognition* determines the posture of a body by finding a match between the visual input and the pre-recorded posture templates. A large amount of domain-specific information is required for this type of recognition to work. Conceptually, the output of a motion recognition system is a name for the state of the body, such as walking or sitting.

On the other hand, *posture tracking* requires less pre-recorded information to recognize human limbs and track their position. The output of a tracking system consists of a stream of body joint positions.

We believe that recognition is a simpler task than tracking when vision is involved. Much more information is available in a recognition system and fewer states of the body are meaningful and need to be analyzed. In contrast, a tracker must he able to analyze every pose the body takes on without any prejudice regarding the likelihood of some poses over others.

Our focus was to allow Predator to perform its task in real time without requiring the actor to wear any special gear or undergo an initialization protocol and still make a minimum number of assumptions about the surroundings of the actor.

Several attempts at satisfying these goals have been made by other groups, but no solution has successfully included real-time processing while simultaneously remaining passive and unintrusive. To date, vision-based solutions have either made limiting assumptions about the scene or used techniques and algorithms that render real-time implementations unfeasible using today’s hardware.

For example, First Sight, one of the best-known vision-based posture trackers, decomposes the body into primitives that do not lend themselves to real-time analysis with today's hardware. Some researchers have attempted to fit deformable body models onto the actor's posture, but these deformable model techniques also prevent real-time processing.

Others have suggested that the actor perform a ritual dance in front of the camera to help identify limbs and determine a model for the body before the tracking begins. Some research has focused on the modeling stage, but little successful work has been done on actual tracking once the parts have been identified.

Other technologies have been successfully applied to posture tracking. One common real-time approach uses infrared sensors instead of cameras and requires the subject to wear infrared reflectors on critical parts of his body (such as the joints).

The most widely used technique is based on magnetic field theory. These systems track motion by detecting the orientation and position of receivers placed on an actor's body with respect to a fixed transmitter placed near the actor. The Flock of Birds is perhaps the best known and most widely used motion capture system on the market. The transmitters are about the size of a die and connected to processing units by cable. The Flock requires the scene floor to be scattered with transmitters, since the transmission range between transmitters and receivers is limited to about three feet, which would prohibit the actor from move around.

Both approaches usually provide very high throughput: 30 frames per second for infrared systems and about 120 samples per second for magnetic systems.

However, today's motion capture solutions share the same shortcoming: each requires the actor to he equipped with cumbersome devices. This requirement is inconvenient and time consuming, especially considering that the cables connecting the receivers to the processing units in magnetic-based system are about as thick as a telephone backbone cables.

Our approach provides a passive means of capturing human motion without the clutter of the paraphernalia required by other systems. We achieve this result admittedly at the expense of some accuracy, but complete passiveness seems a worthwhile trade. Predator also has the potential to track multiple subjects simultaneously, a feature that other systems cannot deliver without additional hardware.

## Hardware

Predator is composed of several pieces of hardware, each of which plays a role in the transformation from visual data to joint information. A typical setup consists of

- Color CCD cameras positioned around the scene
- TMS320C40 parallel DSP
- Host machine to control the TMS320C40
- Client machine to make use of the joint data

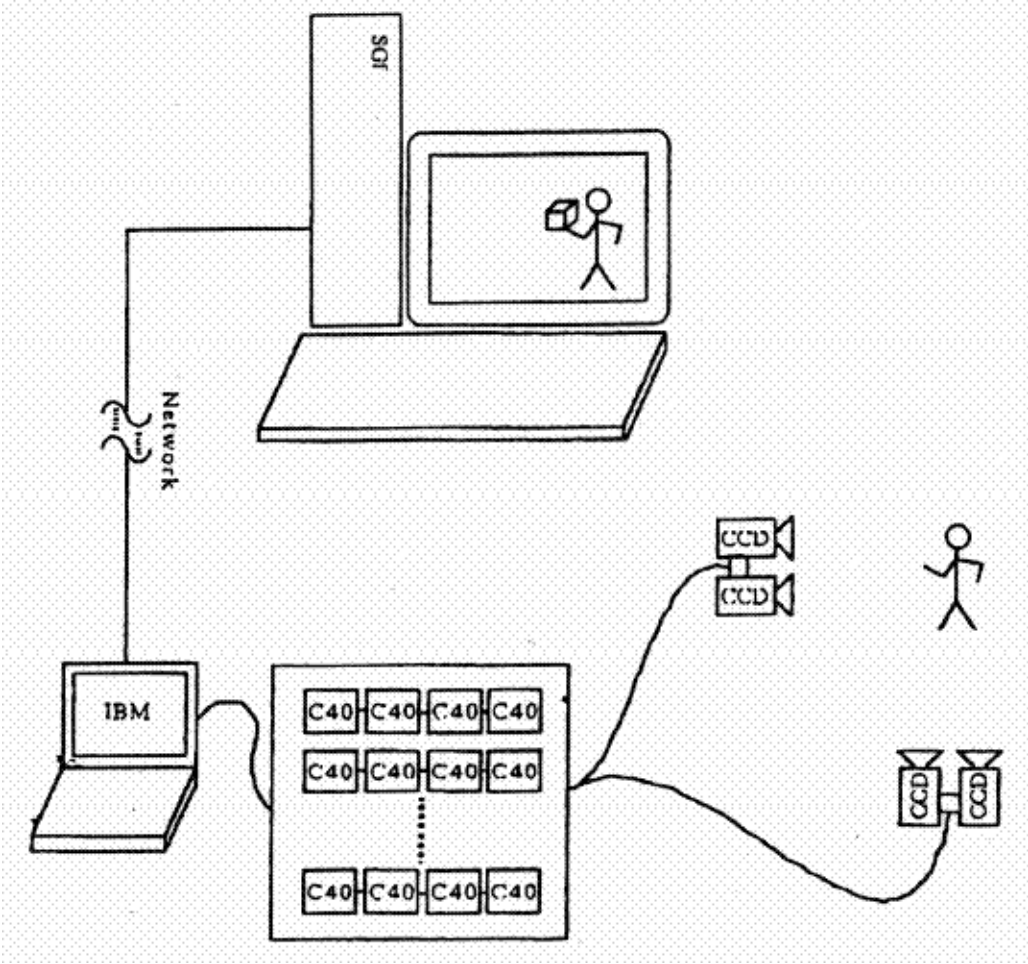
Predator can make use of stereoscopic cameras when the background is heavily cluttered or changing or the cameras are stationary. It can also accommodate monoscopic cameras when the background is static and enough contrast exists between the actor and the background. Regardless of the option, several cameras are needed to deal with the occlusion often present in one view. The task of disambiguating occluded postures becomes much easier once several views are available.

The TMS320C40 parallel DSP is a collection of 8 TMS320C40 TIM modules mounted on VME motherboards manufactured by Traquair Data Systems. Other components of the TMS320C40 parallel DSP include a power supply and ventilation. We optimized the connection topology of the processor network for stereoscopic processing with only one pair of cameras, since we have not yet implemented occlusion reasoning on our TMS320C40 prototype.

We use an IBM ThinkPad equipped with a Traquair ISA C40 rnotherboard to boot the TMS320C40 network, perform further processing on the data, and transmit the joint information to a client. The ThinkPad and its onboard C40 perform reasoning based on motion history and interpolate between data sets when the parallel processor is unable to provide accurate data at a high enough rate. The package joint information is then sent over the network to the client machine or machines.

Everything below and including the host machine is considered part of the Predator system. The client machines can use the tracking information provided by Predator for a variety of applications as described before. We are building a virtual volleyball environment as a sample application. Figure 1 graphically summarizes a typical setup.

*Figure 1. Predator Components*




Note: From right to left: The cameras, TMS320C40 parallel DSP, machine, and client.

# Algorithm

In a way, one can think of Predator as performing a transformation on visual input. From one end, Predator takes in video data from cameras. From the other end, Predator returns joint position information. Internally, this transformation is implemented as a series of stages, each one providing the input for a successive stage. Our description of the process focuses on using only one pair of stereo cameras. We also briefly glance at the monoscopic version of Predator. We do not cover multiple cameras because we have not yet fully investigated their use in handling occlusion.

The stereo processing pipeline performs region analysis on the input images and determines the front-most object by grouping regions together. By using the disparity between the two stereo views, the processing pipeline then skeletonizes the region corresponding to the body and identifies the limbs by wing-specific knowledge of the human body. These steps are described in detail in the following sections.

## Filtering

The images are first passed through a lowpass filter to smooth out the image. We also subsample the image to convert it to a lower resolution. Converting the image to a lower resolution speeds up processing a great deal in the following stages. The high-resolution image can still he used in later stages for improved accuracy.

## Region Growing

We sweep the image from one corner to the opposite corner, left to right, and top to bottom, recursively marking adjacent pixels of similar color with a tag unique to the region. We close off the image whenever a pixel is examined with color that is dissimilar to the color of the region. Once an entire region is identified, the sweep continues where it left off last by initiating a new floodfill whenever it reaches an untagged pixel.

The floodfill itself consists of a depth-first traversal of the image. A pixel is tagged with the region's marker only if its RGB values fall within a certain range of the mean RGB values of the current branch of the traversal. If the pixel is tagged, the average RGB values of the branch art updated, and the process is repeated on each of its eight neighbors.

However, if the pixel's RGB values don’t fall in the required range, we return to the parent stack time, restoring the notion of the *current* pixel and the mean ROB values. In this way, all adjacent pixels of similar color will be interned as part of the same region.

Note that a slight bias exists in our segmentation algorithm because we perform our sweep in a left-to-right, top-to-bottom fashion. The average RGB values of each branch are biased toward the top left-most pixel in the region, because every branch must include that pixel. Through experimentation, we have determined that this bias is not significant enough to warrant a randomized or a more sophisticated approach.

Region growing is often implemented in a scanline fashion in which the image is strictly visited from top to bottom, line by line. We allow our recursive transversal to examine the image in any order it deems appropriate. In addition, we don't incur the cost of the union-find primitives required in scanline-based approaches.

## Finding Correspondences

The filtering and region growing stages are applied to both views of a stereo camera. The correspondence stage is where the stereo information comes together. Each region in the left-view image is matched with its corresponding region in the right-view. The correspondence is made based on:

- Color (each channel is compared separately)
- Bounding box width and height
- Area
- Position of centroid with respect to the bounding box
- Vertical position of the surfaces

We allow a higher error tolerance for the area and the centroid location criteria for smaller objects since the region growing stage can be heavily influenced by small color fluctuations when only a few pixels are present. This algorithm takes $O(nm)$ time to perform its task, where $n$ and $m$ are the number of regions detected in both views. However, $n$ and $m$ will be small enough that this task should not take prohibitively long. If the number of regions in each view is outrageously large, then tile color tolerance in the region growing stage should be increased. Any surface is discarded that doesn't have a match in both views.

## Finding the Front-Most Regions

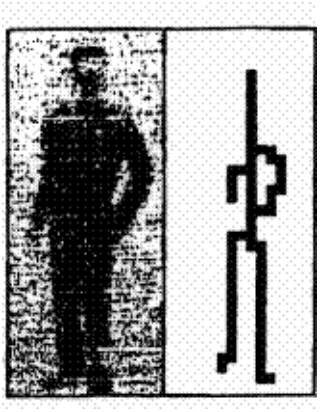
The next step is to find surfaces closest to the camera. We loop through all the regions in one view and compute the average horizontal position disparity between corresponding surfaces. Surfaces whose disparity is greater than a certain function of the mean are kept. All other regions are discarded. We currently keep all surfaces whose disparity is greater than *7/4* of the mean disparity and remove the others.

## Grouping Regions to Form a Body

Having found the front-most regions, we must now determine which of them belong to the body. We remark adjacent regions so that they will be merged into a larger region, thus growing a region that corresponds to the whole body. For cases in which nearby regions are not in contact, we may draw a thick line between the centurions of the disconnected object and the closest surfaces in the body in order to enforce conductivity.

The above presents a slightly simplified version of our algorithm. Consider the case where the actor is wearing a black suite and is standing in front of a white background, his arm resting on his hips to make a loop (see Figure 2).

*Figure 2. Loop on a Body - Placing an Arm on the Loop in the Skeleton*



Our segmentation algorithm recognizes the background area enclosed by the arm and the torso as one region and the rest of the body as another. If we blindly group the enclosed background area with the rest of the body, we will get an incorrect representation of the body's shape.

The solution is to impose an additional criterion that must be met before a region is grouped with the body; that is, the region may not be grouped along if other regions enclose it. The condition is sufficient to preserve the correct shape of the body but adds an annoying restriction for which we have not yet reconciled: the input image of the body should not have a region enclosed within a larger region. If it does, the inscribed region will be dropped out after this stage and an artificial loop will be created in an inappropriate place.

Consider, for example, an actor in a black body suit boasting a large red dot on the chest. The red dot will appear as a hole in the body's final bitmap, completely confusing subsequent stages in the processing. Notice that this restriction doesn't apply to cases where the actor is wearing short sleeved or even striped shirts

## Skeletonization

The skeletonization phase takes the bitmap of the body and returns a skeleton consisting of a set of pixels representing the structure of the posture in a compact form. We introduce this stage by presenting some terms used in this application brief.

**NOTE:**
These definitions apply to our work only and may differ from definitions applied by others.

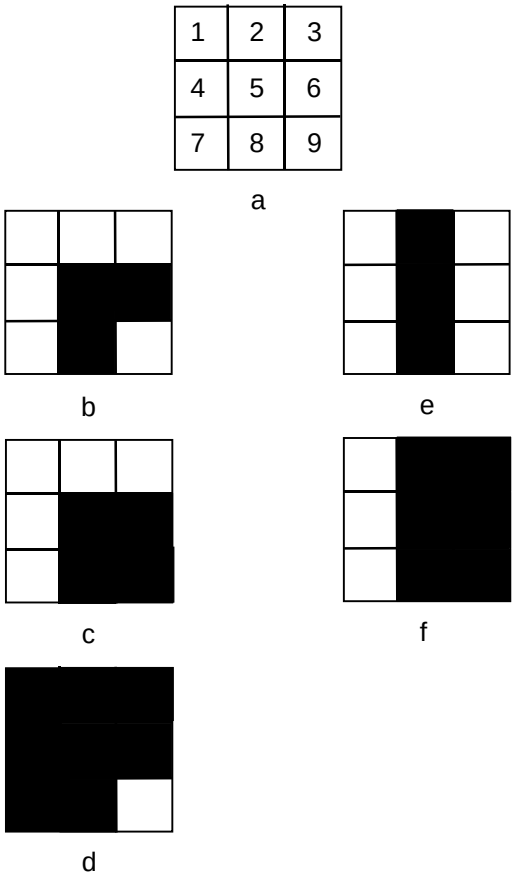
| | |
|---|---|
| **Member** | A component of pixels 1 pixel thick connected with adjacency k=4. The final skeleton has one member for each arm and leg, one for the spine, and one for the head. A member cannot be self-intersecting; that is, it should be impossible to draw a non-nil path that begins and ends on the same pixel without visiting other pixels in the member no more than once. |
| **One-Pixel-Thick Member** | A member in which every pixel is critical |
| **Skeleton** | A collection of connected members. By *connected*, we mean that from any member, it is possible to use k=4 adjacency to reach any other member in the skeleton. |
| **Critical Member** | A member is critical if its removal will cause the skeleton to become disconnected. |
| **Cyclic Member** | A non-nil path that starts and ends on the same member without visiting other members more than once. This is not a strict definition but a property of cyclic members important to us. |
| **Joint Pixel** | A pixel in the skeleton shared by two or more members. Critical and cyclic members must be delimited by two joint pixels. |

**Extremity Pixel** A member pixel with only one neighbor. See the discussion below for details.

The implementation of our skeletonizer requires a test for one pixel thickness, so we present an algorithm that efficiently determines whether a member is one pixel thick. We examine each of its pixels and its surrounding eight neighbors as a 3x3 binary matrix.

Conceptually. if a k=4 path exists within the 3x3 matrix through the center pixel from one edge of the matrix to any other edge, that pixel is critical only if there are no alternative k:=4 paths which don't go through the center pixel. Figure 2 depicts some cases of interest.

*Figure 3. Local 3x3 Matrix Used to Determine Criticality of a Point in a Member*




Notes: a. Our numbering convention
b. In this case, the center pixel is critical because 856 is the only path between 8 and 6.
c. In this case, 5 is not critical because an alternate path 896 exists between 8 and 6.
d. 5 is again not critical because 8741236 is an alternate path for 856. (Notice that three other paths exist with alternate paths.)
e. Another case where 5 is critical.
f. Non-critical 5.

Armed with these definitions and criteria, the inner workings of the skeletonizer become easy to understand. The main step in this stage is to successively erode the contour of the body, discarding non-critical edge points until the remaining connected component is one pixel thick.

We do not erode center pixels that are not involved in any paths in their local 3x3 matrix since they represent extremity pixels and are needed for our stopping case. Eroding extremity pixels causes the skeleton to eventually diminish into one single pixel and disappear after one final iteration.

The medial axis we obtain contains artifact members in addition to the desired limb members. To augment the problem, the medial axis is also extremely sensitive to fluctuations around the contour of the body and may produce radically different artifacts with the slightest change in the contour. We therefore precede the skeletonization with a smoothing algorithm and follow it with a cleanup procedure.

The smoothing step consists of performing a single pass around the edge of the body to remove extremity pixels. Although this step doesn't prevent extraneous members from being generated, it helps ensure that they will not be as pronounced as the legitimate members.

The cleanup stage consists of decomposing the skeleton into members classified as limbs, critical members, or cyclic members. We can show that our erosion technique produces no cyclic members if the original bitmap has no holes. Since the segmentation stage guarantees that holes are produced only for looping limbs, we know that cycles in the skeleton must correspond to limbs and therefore must not be removed.

Furthermore, since we know that the skeleton must be connected, critical members cannot be removed either. Thankfully, the only critical members our erosion technique can create are those which connect limbs together.

The only such member in the case of the human body is the spine. Detecting the skeletal artifacts then becomes a tractable problem: artifact members can manifest themselves as only non-critical, non-cyclic members, or what we call *floating members.* To clean up the skeleton then, we simply remove the floating members one by one until there are no more than six limbs (head, arms, legs, and spire). All free members and some circular members contribute to the number of limbs. The only circular members that do not contribute to the count are redundant parts of the spine.

## Member Identification

We have yet to identify the limbs. If we assume that the body is right side up, the upper part of the spine joins three limbs (the head and the arms) and the lower part connects only the two legs. We can easily discern the head from the arms based on length. We handle cases separately where the arms loop to the hip. We have not provided provisions for determining whether the body is upside down, since our motion history engine will be able to provide us with that information once it is implemented.

## Fitting

The final step is to fit consecutive and adjacent lines to each limb and to output the coordinates of the endpoints of these lines. Specifically, for each remaining limb, we would like to fit consecutive lines that share a common endpoint and a length ratio similar to that of the corresponding human limb. For each member, we determine the line parameters that minimize an energy function. This is based on the similarity of the ratio of the lengths of the line and the ratio of body members, and the least mean square error of the fit to the points of the member.

Since the stereo information is only used in identifying the front-most region, it is easy to replace it with something simpler for cases where the background can easily be separated from the body. A feasible segmentation approach is to store the background image before a human enters the scene, and then simply subtract this image from subsequent input frames.

Subtraction is simple and efficient, but poses several limitations on the scene. First, there must be a high contrast between the body and the background. Complications arise if the actor happens to walk in front of a region of the wall that bears the same colors as the actor's clothing. The camera must also be of very high quality, as it must produce the exact same pixels after every frame.

The quality of the camera won' t matter, however, if the lighting conditions aren't perfect. Older fluorescent lights flicker noticeably, perturbing the static regions of the background. Remember also that with a stereo setup, the operator has the ability to pan the camera and follow the actor around the scene. Monoscopic setups can't provide this luxury. Although filtering can solve a few of these types of problems, most of them cannot be remedied so easily.

## Support Software

We built a development environment for our multiprocessor to facilitate our implementation tasks. Since UNIX is the preferred development platform for most serious research work, we wrote various pieces of support software, including

- Our own C40 network boot loader
- Linux drivers for various C40 boards
- Sophisticated configuration package for C40s providing configuration inheritance
- Profiler to identify bottlenecks in our software
- Worm-router for message passing
- Port of the ANSI *C* studio library
- Microkernel based on our own multithreading package.

We plan to develop additional pieces of system software for the C40. One of our goals is to extend our microkernel to allow us to simply define logical stages in a pipeline and let the system dynamically decide at run-lime how to distribute the tasks among the processors in the network. This would eliminate the typical profile-edit-compile cycle to which we have become accustomed in low-level parallel processing development.

## Further Work

We must add several new features and make a few changes before Predator can become a marketable tool.

We have not yet fully designed the occlusion-reasoning engine. Predator requires multiple cameras to be installed on the scene. Now the actor must assume simpler poses. As a holdover, we provide a simple module capable of making use of skeletons from multiple views. It outputs the most coherent set of joint information provided to it by the pipeline corresponding to each view. We have begun designing a more sophisticated module.

The use of motion history is another feature that could benefit Predator in terms of reliability. Our system currently makes almost no use of previously gathered data. Hence, it cannot benefit from the added accuracy and speed that could be gained from its previous results. Known physical constraints on a human body can be used to prune the number of possible poses a body can take after a previous pose. Presently, we make use of previous body locations and perform our segmentation in only a small region of the Image.

We've considered using multi-resolution processing in our vision algorithm to further increase efficiency. We can take advantage of the fact that every part of the scene doesn't require the same amount of attention. As mentioned above, we already take advantage of the fact that only image regions corresponding to the human body will be passed further up the pipeline. Segmentation is performed only in regions where the body is likely to be located. Other stages of the pipeline can take advantage of the fact that some regions of the image are more important than others.

We would also like to more closely couple the skeletonization and the segmentation algorithms for added robustness. Knowledge of the skeleton may prove to be useful in performing segmentation.

In general, we would like to make our system more conscious of the work it has already performed. By making better use of feedback, we hope to improve both efficiency and accuracy.

## Conclusion

We have introduced Predator, a vision-based posture tracking system. Predator takes as input video footage of a human actor and produces a stream of joint position information representing the position of the actor's limbs over time. Our design goal was to build an unintrusive system to analyze the video data in real time without requiring the user to wear any special gear or undergo any other initialization steps.

## Team Background

Jitendra Malik is a professor of computer science at UC Berkeley. He is the head of the UC Berkeley vision group, which is heavily involved in the California PATH project. PATH aims at improving the highways of tomorrow through technology. The UC Berkeley vision group has already implemented two major components of the PATH project on C40's. One project constitutes the attitude control system for an autonomous car. The other involves building a device to be placed at every city road intersection to replace the loop-detector technology employed in traffic control today. We have hopes of producing a reliable C40-based system within a year. The federal government has determined that computer vision is the preferred technology on which the roadways of the future should be based. Our group is receiving $1,000,000 per year from the federal government to pursue our research.

Ali Rahimi is a third year undergraduate majoring in electrical engineering and computer science, He is a member of the computer vision group at UC Berkeley and conducts his research under Professor Malik. His research focus is vision-based tracking and operating system design. Part of his research in the UCB Robotics Lab involves implementing real-time vision systems on a custom C40 platform.

Michael Shilman is a fourth year UCB undergraduate majoring in electrical engineering and computer science. His research interests focus on virtual environments and next-generation user interface peripherals as they relate to CAD. The Berkeley CAD group is responsible for some of today's most widely used circuit design and DSP tools (Spice and Ptolemy among others). His role in this project consisted of building a sample client application that demonstrates our system.