Proceedings of 2004 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 28 - October 2, 2004, Sendai, Japan

Multi-Attribute Utility Analysis in the Choice of a Vision-Based Robot Controller

Nicholas R. Gans and Seth A. Hutchinson

ngans@uiuc.edu[1], seth@uiuc.edu[2]
[2]Dept. of General Engineering
[2]Dept. of Electrical and Computer Engineering
The Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
Urbana, IL USA

Abstract—**We present an example of the use of multi-attribute utility analysis in the design of a robot system. Multi-attribute utility analysis is a tool used by Systems Engineers to aid in deciding amongst numerous alternatives. Its strength lies in the fact that very different metrics can be compared, and that it takes into account human preferences and risk attitudes.**

As a design tool, multi-attribute utility analysis is performed off line, during the system design phase, to choose among possible designs, components, gains, etc. We offer a demonstration of multi-attribute utility analysis in designing a Hybrid Switched-System Visual Servo System. We have previously introduced such a system, and here use multi-attribute utility analysis to to select a switching algorithm that best suits the needs of a specific user.

I. Introduction

The ultimate goal of much research is to provide a benefit to human society. Research in robotics can improve human safety and health and simultaneously improve the quality and reduce the price of consumer goods. When an automated system is to serve the needs of a human, that person's specific performance needs must be met. Multi-Attribute Utility Analysis (MAUA) [1], [2] is a method used by systems engineers to directly gage a human's preferences and attitude toward risk. It provides a way to equate distinct system attributes, including those that do not have a natural metric. In a design application, MAUA is performed off line during the design phase to select among competing designs, components, gains etc., using metrics such as failure rate, cost, speed, and even non-concrete measurements such as user satisfaction. Additionally, MAUA can account for human preferences and attitudes toward risk. For example, different applications can have different acceptable failure rates, and different people using the same system may be willing to accept different failure rates as well.

Visual servo control is the use of image data in the closed loop position control of a robot end-effector. There are two general approaches to visual servo control: Image-Based Visual Servoing (IBVS), and, Position-Based Visual Servoing (PBVS). In IBVS, an error signal is measured in the image, and is mapped directly to actuator commands. In PBVS systems, features are detected in an image, and used to generate a 3D model of the environment. The error is then computed in the Cartesian task space. There are extensive resources detailing these methods [3], [4], [5], [6], [7]. It is well known that both methods have specific strengths and shortcomings [8]. Furthermore, these systems are complementary in the sense that IBVS performs well where PBVS performs poorly, and vice versa.

We have proposed a switched system approach to visual servoing and experimentally verified its potential [9]. A random switching rule can be applied, where at each iteration or time unit the system randomly selects between using IBVS or PBVS. As time, and the number of switches, increases, we can expect performance to be influenced by both IBVS and PBVS. The resulting system performs reasonably in controlling both the image and the position, rather than extremely well in one and poorly in the other.

Our switched system previously used a binary switching rule, at each iteration the choice to use IBVS or PBVS was made by a binary random process with equal probability of 0.5 of choosing either one. However, this may not be the best switching rule for all configurations or all uses. Industrial robots can have reaches ranging from over two meters to less than a third of a meter, and even smaller in surgical or MEMS applications. Likewise, the use of wide and narrow view lenses imposes different needs in feature control. A system that is biased to select IBVS will provide better control of the image features; a PBVS biased system will better control the position. Additionally, failure of a visual control system can vary in severity. For instance, failure can result in a small increase in scrap rate or a fatal catastrophe depending on the application.

For these reasons, users selecting a binary coefficient can have very different performance goals and very different attitudes toward risk. To provide a means of selecting a binary coefficient, we look toward the field of decision theory and multi-attribute utility analysis. Multi-attribute utility analysis provides an ideal method of gaging a user's performance needs, preferences, and attitude toward risk [1], and we can directly apply this information to select a

0-7803-8463-6/04/$20.00 ©2004 IEEE

switching rule.

In this paper we use MAUA to select a binary switching rule for a switched visual servo system. A decision maker's preferences and risk attitudes toward position control, feature control, and control time will be used to create a three-attribute utility function. We will then use this function to rate the utility of several switched system, each using a different probability in the binary random switching rule. Section II will discuss image based and position based visual servoing, and introduce our switched system. MAUA is best explained along with an example. Section III will discuss the background of MAUA and steps taken to prepare a multi-attribute utility analysis while we present a MAUA of the switched-system visual servoing system for a specific user. Finally, Section IV will present the results of the example utility analysis.

II. Visual Servoing

A. Position Based Visual Servoing

The task in PBVS is to regulate the error between the current camera pose and the goal pose. Given a current camera position and orientation $\mathbf{r}$ and goal position $\mathbf{r}^*$ (from here on, variables in the goal configuration will be denoted with *), the transformation relating them is described by a translation and rotation of the camera frame. The translation and rotation are quantified by ${}^{c^*}\mathbf{T}_c \in \mathbb{R}^3$ and ${}^{c^*}\mathbf{R}_c \in SO(3)$, respectively. There are a number of ways to decompose the rotation matrix to three variables, in the following discussion we will use and $\mathbf{R} \equiv \mathbf{u}\theta$, where θ is a measure of rotation about the vector $\mathbf{u}$.

Given a collection of feature points in the image, there are numerous methods to estimate the position and orientation of the camera [10], [11], [12]. These methods differ in speed, accuracy and the number of feature points required. Some methods require a CAD model of the 3D points as well. With $\mathbf{r}^*$ known and $\mathbf{r}$ estimated from the feature points in the image, we define the pose error as

$$\mathbf{e}_p = \mathbf{r}^* - \mathbf{r} = [{}^{c^*}\mathbf{T}_c, \mathbf{u}\theta]^T. \tag{1}$$

with derivative

$$\dot{\mathbf{e}}_p = \dot{\mathbf{r}}$$

Moving the robot back along the error vector generates a velocity vector of

$$\dot{\mathbf{r}} = -\lambda_p \mathbf{e}_p \tag{2}$$

where λ_p is a scalar gain coefficient. The derivative of the error then is

$$\dot{\mathbf{e}}_p = -\lambda_p \mathbf{e}_p. \tag{3}$$

Clearly, all errors will tend to zero with time, and this system is Asymptotically Stable (AS). It is not Globally Asymptotically Stable a map from $SO(3)$ to three variables can never be a global map. While the position error tends monotonically to zero, we have no control over the position of the image points. If there is any rotation present the feature points will move along curves as the camera

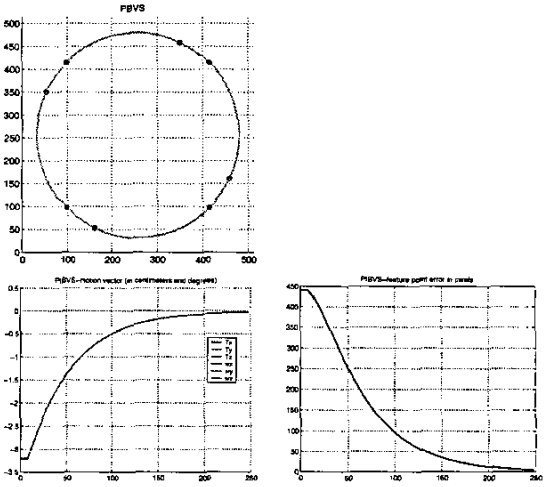


Fig. 1. Example Of Typical Motions For Feature Points And Camera In PBVS

undergoes rotation and translation, this is seen in Figure 1. The first image shows the trajectories of four feature point as the camera moves from its initial pose to the goal pose. The points marked by 'O' are the features seen when the camera is at the initial point and the points marked with '*' seen when the camera is at its goal position. A large circular path is traced out in the image and the features are close to leaving the image. However, the camera velocity vector in the next figure shows an exponential decrease along a single axis.

The limited imaging surface of a camera makes it possible for the feature points to leave the image. In this case the system can no longer reconstruct the motion parameters and the task cannot be completed. We will define failure of a visual servo system to be any situation in which it does not successfully zero the error.

B. Image Based Visual Servoing

In Image Based Visual Servoing, the task we are regulating exists in the image space. A collection of n feature points is extracted from the image. Each feature point has coordinates in the image plane, $\mathbf{p}_j = [x_j \; y_j]^T$, where x and y are the horizontal and vertical distance from the center of the image. We can define the image error between a point's position $\mathbf{p}_j$ in the current image and its position $\mathbf{p}_j{}^*$ in a goal image stored in memory as

$$\mathbf{e}_{ij} = \mathbf{p}_j{}^* - \mathbf{p}_j. \tag{4}$$

The image error for each point is when the camera is at the goal pose. Thus the task of correctly positioning the end effector becomes the same as moving the feature points to their goal positions.

The motion of an image point, $\dot{\mathbf{p}}$, is related to the motion

of the camera,$\dot{\mathbf{r}}$, by

$$\dot{\mathbf{p}} = \mathbf{J}_i\dot{\mathbf{r}} \quad (5)$$

$$= \begin{bmatrix} \frac{1}{z} & 0 & -\frac{x}{z} & -xy & 1+x^2 & -y \\ 0 & \frac{1}{z} & -\frac{y}{z} & -1-y^2 & xy & x \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

where $\mathbf{J}_i$ is known as the image Jacobian[4], [5].

We can take the time derivative of (4) and combine it with (5) to get

$$\dot{\mathbf{e}}_{i_j} = -\mathbf{J}_i\dot{\mathbf{r}}. \quad (6)$$

If we have three or more feature points, we can stack the vectors and matrix of equation (6) to build a full rank Image Jacobian, and solve for the robot motion as

$$\dot{\mathbf{r}} = \lambda_i\mathbf{J}_i{}^+\mathbf{e}_i. \quad (7)$$

where $\mathbf{J}_i{}^+$ is the general inverse of $\mathbf{J}_i$. This gives the proportional control law

$$\begin{aligned} \dot{\mathbf{e}}_i &= -\lambda_i\mathbf{J}_i\mathbf{J}_i{}^+\mathbf{e}_i \\ &= -\lambda_i\mathbf{e}_i \end{aligned} \quad (8)$$

This system is Asymptotically Stable (AS). It is not Globally Asymptotically Stable since there exist camera positions or feature point configurations which cause the image Jacobian to become poorly conditioned. In this situation the image Jacobian inverse may not exist.

Another issue is that of camera retreat. As described by Corke and Hutchinson [13], the image trajectories will follow a straight line to their goal configuration, requiring a change of scale must take place to turn the normally curved trajectories into straight lines. This scaling is achieved by pulling the camera back along it's optical-axis. An illustration of this phenomenon is given is Figure 2. In contrast to the PBVS system, the feature points move along nearly straight paths to their goal positions, but there is a large translation along the optical-axis.

Most robot systems have a reachable space on the order of meters. Thus camera retreat may cause the robot to extend to its joint limits during visual servoing, resulting in failure. Alternately, retreat can seriously affect the camera by causing the focus to be incorrect or increasing quantization noise, affecting performance of the system.

C. Switched System Visual Servoing

In order to mitigate the troubling aspects of the systems, we have introduced switched system visual servo controllers [14], [9]. These systems switch between using IBVS and PBVS at different points during the period of the task. Switching can be triggered depending on the state of the two errors $\mathbf{e}_p$ and $\mathbf{e}_i$, at specific time intervals, or randomly. Here we explore random switching, using a binary random variable to pick a system at each iteration of the algorithm. Our previous research had used a binary random variable with a 50% chance of selecting IBVS or

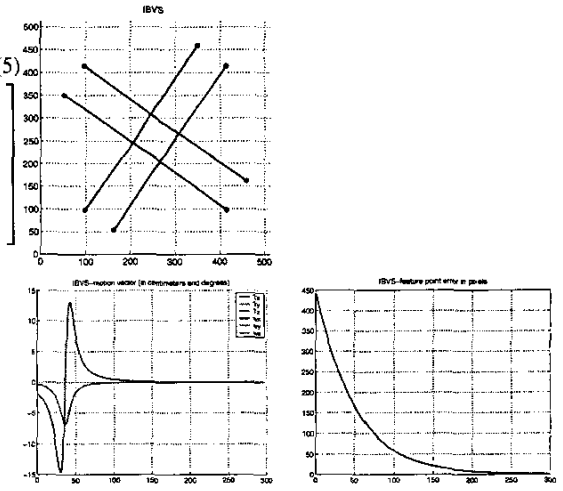


Fig. 2. Example Of Camera Retreat Under IBVS With Large Rotation

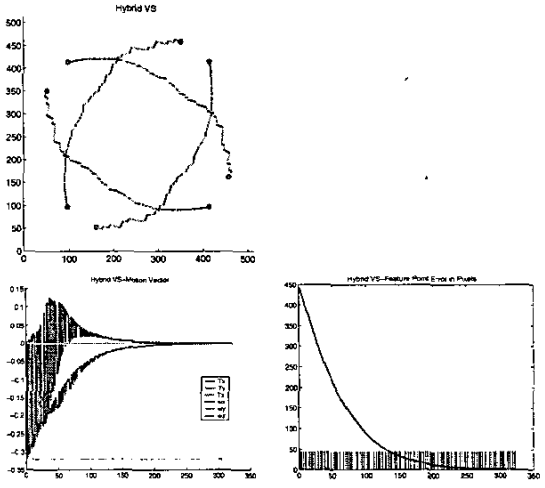


Fig. 3. Example Of Camera Retreat Under Switched System VS With Large Rotation

PBVS. We demonstrated a marked decrease in failures due to extreme robot motions or lost feature points.

An example of this hybrid system running is seen in figure 3. The feature point trajectory is now a slight curve, as opposed to the previous large curve or straight path. The camera also experiences some camera retreat, but far less than under IBVS.

III. Multi-Attribute Utility Analysis

As discussed in section II-C, we have previously used a switching visual servo system that had a 50% chance of selecting IBVS or PBVS at each iteration. This resulted in fewer system failures. However, while a switched system may be desirable to reduce the chance of failure, some tasks or configurations may be more lenient toward extremes in camera motion than image feature motion. One example

is a large robot with no danger of collision with nearby objects, but which has a narrow-view camera lens. The converse situation can exist where it is desirable to trade excess robot motion to better constrain your features in the image. A binary function that favors one system over the other will grant these results.

Multi-Attribute Utility Analysis (MAUA) can be applied to determine the best binary function. MAUA is a well established method for comparing options with multiple objectives. Furthermore, it allows the inclusion of not only preference, but attitudes toward risk [1], [2].

The person choosing the system referred to as the decision maker. Typically this is someone knowledgeable in the field and closely involved in the task at hand. A few examples of decision makers include a project manager in charge of implementing a control system, or a customer for whom a system is being designed or performance improved. MUAU is intrinsically an individual pursuit. A change of decision makers will require the analysis be ran again.

The goal is to maximize the system's expected utility for the decision maker. Utility is a unitless measurement, which allows vastly different attributes to be compared and also incorporates the decision maker's preference and risk aversion. Each attribute is assigned a utility function which is monotonic with the attribute. The utility functions can then be weighted and combined to form a Multi-Attribute Utility Function (MAUF). If statistical data is available concerning the attributes, we can then solve for the expected value of the MAUF for different weights and pick the one that maximizes the expected utility. This will be elaborated upon below.

Given n attributes, $x_1, ..., x_n$, the multi-attribute utility function , $U(x_1, x_2, ..., x_n)$, is defined by the equation

$$1 + KU(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} [Kk_i U_i(x_i) + 1] \quad (9)$$

where U_i are individual utility functions for each attribute, k_i are individual scaling or weighting constants, and K is a normalizing constants which is the non-zero solution to

$$1 + K = \prod_{i=1}^{n} [Kk_i + 1]. \quad (10)$$

There are several basic steps to the development of a MUAF, each of which will be presented here. First a decision maker must be chosen to make value judgments. Second is the choice of suitable attributes and confirmation that the decision maker can view them independently from each other. Third is the development of the individual utility functions and scaling constants. Finally, given information about probable outcomes for the task, the MUAF can be used to evaluate the different options.

For the purposes of this paper, the decision maker was an Electrical Engineering graduate student, who has been involved in vision based robot control for two years. First we sought attributes that measured non-ideal performance of a system. In the image plane, we define a line segment

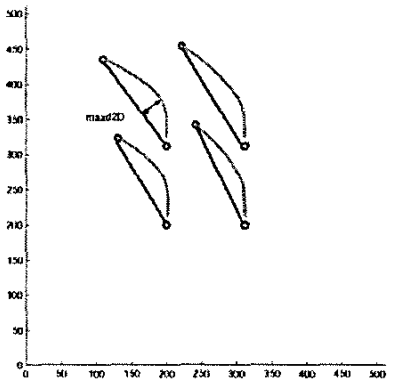


Fig. 4. Illustration of $d2D$

for each feature from the initial feature point coordinate, $\mathbf{p}$, to the goal coordinate $\mathbf{p}^*$. At each iteration we measure the distance from each feature point to the line segment, and note the largest distance. We will refer to this measurement as $d2D$, and is illustrated in Figure 4. We expect IBVS to move the features along trajectories very close to these line segments, so $d2D$ should remain small for all but the most difficult tasks. PBVS, which offers no control over the image features is expected to have much larger measurements of $d2D$.

Similarly, we define a three dimensional line segment in the camera workspace from the initial camera position $\mathbf{T}$ to the goal positions, $\mathbf{T}^*$. At each iteration we measure the 3D distance from the camera's current position to the line segment; we will refer to this measurement as $d3D$. We expect PBVS to have small measurements for $d3D$, while it will be much larger for IBVS. We ignore camera orientation due to the difficulty in measuring distances and establishing a "straight" line in SO(3), and the fact that erroneous translation are often troublesome for IBVS while it typically performs well for rotations.

For two of the attributes, we used the *maximum* $d2D$ and $d3D$ seen over the course of a visual servo task. These two attributes are referred to as x_{2D} and x_{3D}, and we expect IBVS and PBVS to perform as discussed previously. Finally, we measure the failure rate for each system, and x_{fail}. This gives three attributes.

To use MAUA it is necessary to insure that the decision maker can view these attributes independently in the sense of mutual preferential independence and mutual utility independence [1]. Preferential independence exists if a decision maker's preference for attribute Y does not depend on attribute X. For example, a small value of Y might always be preferred to a large value, regardless of the value of X. Preferential independence exists for most decisions.

Utility independence is more complex. It insures that a decision makers preferences for an attribute Y for *uncertain* outcomes do not depend on X. For example, suppose the decision maker prefers the scenario of a 50% chance of either Y1 or Y2 and 100% chance of X1 to the scenario of

30% chance of Y1, 70% Y2 and 100% X1. Then for mutual independence to hold he or she must also prefer 50% chance of either Y1 or Y2 and 50% chance of X2 to 30% chance of Y1, 70% Y2 and 50% X2. These requirements must hold for any values of the attributes, and all attributes must be tested against each other.

For these attributes, the decision maker determined they were mutually independent. The next step was to determine individual utility functions for each attribute, and to build the multi-attribute utility function. The decision maker decided that x_{2d} =175 pixels was the maximum he would accept and $d3D_{max}$ =1.5 meters was the largest acceptable $d3D_{max}$, since he felt that exceeding these values would likely lead to system failure. The largest failure rate the decision maker was willing to accept was 20%. This gives us the following utilities for the extremes of the domains,

$$U_{d2}(175) = 0,\ U_{d3}(1.5) = 0,\ U_{fail}(20) = 01$$
$$U_{d2}(0) = 1,\ U_{d3}(0) = 1,\ U_{fail}(0) = 1. \qquad (11)$$

Note that it is possible to have x_{2d} and x_{3d} larger than the decision maker's chosen maximums, indeed during simulations the systems often exceeded these limits. Any value larger than these maximums is given a utility of zero. This is justifiable since there is no need to distinguish between failures. For example, assume a robot has a one meter reach. The system will fail if the robot reaches it's joint limits, regardless of whether x_{3d} is two meters or five meters, and any viable robot controller will safely stop the robot. Neither system fails "worse" than the other. Still, a system that fails more often will have have a lower average utility due to the high probability of a zero utility.

The decision maker was queried using the certainty-equivalent and probability methods to determine the individual utility functions [15]. This amounts to assigning a utility of 1 for the best outcome, which is 0 for all three attributes here, and a utility of 0 for the worst acceptable outcome. Picking the measure of the attribute that the user feels has a *utility* halfway between the best and worst gives the measure that has a utility of .5. This is repeated for several utility levels. The functions were then line fitted to polynomial functions.

It is important to note that a utility of $x \in (0,1)$ often does not correspond to the attribute value at that same percentile between the highest and lowest values of the utility function range. For example, the point at exactly half way between the best and worst attributes often does not have utility of 0.5.

The utilities chosen reveal characteristics of the decision maker. A decision maker who does assign utility x to the value at x(×best-worst) for all x is said to be *risk neutral* and will have a linear utility function. A tendency to assign a utility of x to a value below the risk neutral line indicates the decision maker is *risk adverse* and will accept a known certain, mediocre performance over an uncertain, possibly very good or possibly very bad performance. The opposite is known as *risk seeking*, in which the decision maker will accept the risk of bad performance for the chance

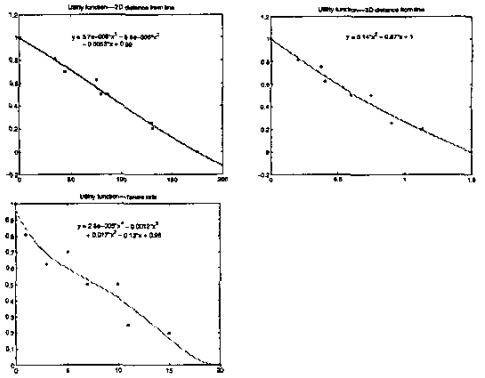


Fig. 5. Utility functions for 2D and 3D

of getting good performance, and prefer this chance to a known moderate outcome. It is not unusual to show different behaviors over different portions of the utility function range.

The resulting Utility functions are as follows:

$$\begin{aligned} U_{2D}(x_{2d}) &= 3.7 \times 10^8 * x_{2d}^3 - 8.6 \times 10^6 * x_{2d}^2 - \\ &\quad 0.0053 * x + 1 \\ U_{3D}(x_{3d}) &= 0.14 * x_{3d}^2 - 0.87 * x_{3d} + 1 \\ U_{fail}(x_{fail}) &= 2.8 \times 10^{-}5.00 * x_{fail}^4 - .0012 * x_{fail}^3 + \\ &\quad .017 * x_{fail}^2 - .13 * x_{fail}^2 + 1 \end{aligned}$$

Plots of the points and fitted functions for the nonlinear functions are seen in Figure 5. They are monotonic functions, which is necessary for use in a MUAF.

The decision maker is quizzed as follows to determine the value of the scaling constants k_i [15]. The decision maker assigns a utility U_{chosen} to the combined function (9)when two of the attributes are at their worst and one attribute is at its best value. The k_i can then be solved for the attribute at its best. Using the values found in 11, k_2d this is as follows:

$$\begin{aligned} 1 + KU(x_{2d}, x_{3d}, x_{fail}) &= [Kk_{2d}U_{2d}(x_{2d}) + 1] \times \\ &\quad [Kk_{3d}U_{3d}(x_{3d}) + 1] \times \\ &\quad [Kk_{fail}U_{fail}(x_{fail}) + 1] \\ 1 + KU_{chosen} &= [Kk_{2d}U_{2d}(0) + 1] \times \\ &\quad [Kk_{3d}U_{3d}(1.5) + 1] \times \\ &\quad [Kk_{fail}U_{fail}(20) + 1] \\ U_{chosen} &= k_{2d} \end{aligned}$$

This was repeated for all k_i ,which were determined to be $k_{d2D} = 0.35$, $k_{d3D} = 0.375$, $k_{fail} = 0.15$. As stated, K is the non-zero solution to (10) and was determined to be $K = -0.8187$. This gives us enough information to use the MUAF $U(x_{2d}, x_{3d}, x_{fail})$ as in (9). Plots of the MAUF for U vs x_{2d} and x_{3d} for several values of x_{fail} are seen in Figure 6.

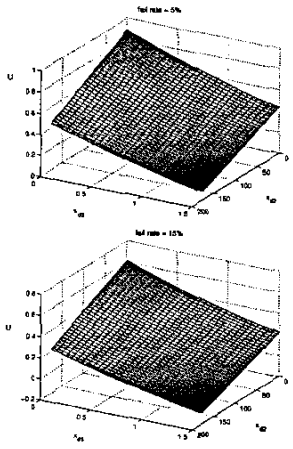


Fig. 6. Plots of U for varying x_{2d} andx_{3d} and several fixed values of x_{fail}

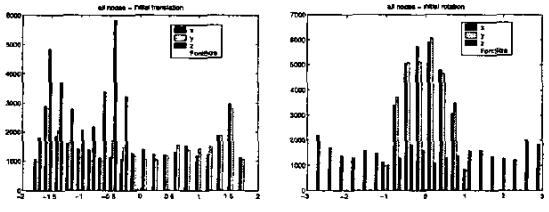


Fig. 7. Histograms of the sampled configuration space

In order to gain information on the probable performance of the visual servo systems, we then ran simulations for IBVS, PBVS, and several switched systems with differing binary functions. Specifically we tested systems that selected IBVS 25%, 50% and 75% of the time. We sampled the six-dimensional configuration space (translation and rotation about three axes), representing 30,000 unique initial camera poses. This is a very large sampling, but SE(3) is a large space and we wanted to insure sufficient sampling density.

The size of the configuration space was a 4m×4m×2m box with the simulated image features at the center of the bottom face. The orientations ranged from -2π to 2π for rotation about the camera optical axis (z-axis), and $-\pi$ to π for rotations about the x- and y-axes. These two axes were more limited since a rotation with magnitude greater than π will result in the feature points lying behind the camera. Histograms of the sampled translations and rotations are seen in figure 7. The sampling is not uniform due to the sampling method used, a rapidly-exploring random tree (RRT) [16], [17]. An RRT offers many advantages, such as completeness, but such a discussion is beyond the scope of this paper.

After running the IBVS, PBVS and hybrid systems for the entire 30,000 camera positions, we plotted cumulative distribution functions for x_{2d}, x_{3d}. Running these simulations took approximately 16 hours of computer time.

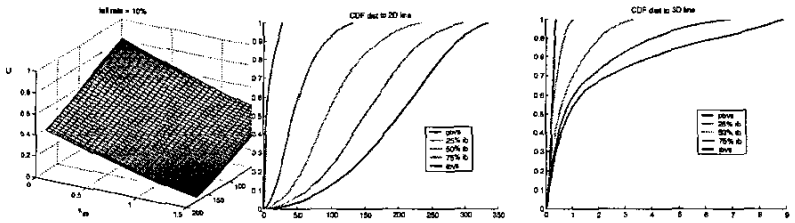


Fig. 8. Cumulative Distribution Functions for Measured Attributes

Simulations were distributed over four Pentium 4 PC's to reduce the running time. The cumulative density functions for the systems vs the distance metrics are shown in Figure 8. As expected, the probable x_{2d} is much smaller for IBVS than PBVS. Likewise the probable x_{3d} is much smaller for PBVS than IBVS. The hybrid systems lie in between along regular gradations.

The failure rate is a discrete value, so there are no density functions. PBVS has a failure rate of 3.52%, while IBVS had a failure rate of 6.12%. The switched system choosing IBVS 75% of the time had a failure rate of .8%, while the 50% IBVS and 25% IBVS switched systems had a failure rate of zero .

We can use this information, along with the MAUF to evaluate the VS systems and choose the system with the highest utility. This information is most easily presented in the form of a decision tree, which is presented in Appendix A. There are five chance nodes, one for each visual servo system. In order to accurately gage the performance of each system, while keeping the amount of data manageable, the chance nodes have five branches corresponding to five ranges of decreasing performance. Each branch has a weighting of 0.2 corresponding to a 20% chance of choosing that branch. In effect the first branch supposes there is a 20% chance the system will operate somewhere in its best 20% of performance. The value of the attribute that corresponds to 0.2 in the CDF's are then used in the MAUF, so the utility value that results is best thought of as a lower bound. The second branch states there is a 20% chance of performance to the 20%-40% of performance, and gives the lower bound of utility in this range. The subsequent branches continue this analysis trend. A detailed examination of the decision tree will be presented in the next section.

IV. Results

A decision tree was created to present the results of the multi-attribute utility analysis, and is presented in Appendix A. For the utility function and weighting functions derived here, IBVS comes out on top with a utility of 0.577. The switched system biased toward IBVS is second with utility of 0.525, followed by the PBVS biased switched system and neutral switched system with utilities of 0.459 and 0.253, respectively. PBVS comes in last with utility of 0.43, this points to the fact that IBVS dominates the other

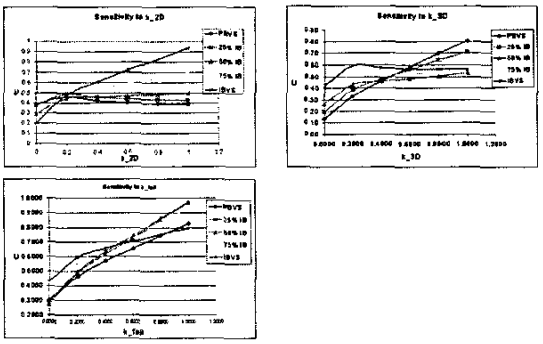


Fig. 9. Analysis of Sensitivity of U to weighting constants

systems, a fact that will be reinforced during a subsequent sensitivity analysis.

This domination can be seen by inspecting the utility results shown in the decision tree and the CDF's in Figure 8. Both IBVS and PBVS perform similarly well for their respective best attribute . However, the 3D performance of IBVS is better than the 2D performance of PBVS. This is seen in the CDF's. For x_{3d}, IBVS has a steep slope that becomes shallow after about 0.6, while for x_{2d} PBVS is shallow over the entire range. The result is that PBVS has a greater probability of having large values for its off-attribute. Looking at the utility results in the decision tree, PBVS only has a non-zero U_{2D} for its first branch. IBVS has non-zero U_{3D} for three branches. All of the hybrid systems have shapes similar to these, gradually shifting from IBVS to PBVS.

We can deduce that utilities are more dependent on how bad each system is capable of performing than on how well each system is capable of performing. Additionally, since a relatively low weighting was given to the failure rate, IBVS's poorer performance for this attribute did not cost it too heavily.

To examine the sensitivity of the analysis to changes in the weighting constants, we vary a single constant from 0 to 1 while keeping the others the same as chosen by the decision maker. We can view the resulting total utilities as a function of the shifting attribute, as seen in Figure 9. When varying k_{d2D}, IBVS would be chosen for $k_{d2D} > 0.21$, but when varying k_{dDD}, PBVS becomes the best choice only when $k_{d3D} > 0.58$. Additionally, IBVS's poorer failure rate only becomes and issue when $k_{fail} > 0.4$, at which point the switched systems become more attractive.

V. Conclusion

We have performed a multi-attribute utility analysis in an effort to choose a vision based control system for a robot manipulator. Two general systems exist, image based and position based. Each one offers strengths and weaknesses, but recent analysis showed that a hybrid system that switched between them showed promise. By biasing the switching toward image based or position based we can obtain a large number of potential controllers with a wide spectrum of performance. The choice of a switching rule should allow the system to perform according to a system designer's preferences, and MAUA is a natural method to choose the rule.

IBVS is known to perform best in terms of feature point motion in the image, and PBVS is known perform best in terms of distance optimal robot motion. Indeed, both of these systems are rated above any hybrid method when the feature point motion or robot motion are the preferred performance metric to improve. The hybrid systems have lower failure rates than IBVS or PBVS, and if the failure rate is the attribute most heavily weighted, they are chosen. However, IBVS dominates the other systems in the sense that it is preferred over the largest range of the three attributes.

References

[1] R. Keeney and H. Raiffa, *Decisions with Multiple Objectives*. Cambridge, UK: Cambridge University Press, 1993.

[2] M. Merkhofer and R. Keeney, "A multiattribute utility analysis of alternative sites for the disposal of nuclear waste," *Risk Analysis*, vol. 7, pp. 173–194, 1987.

[3] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. on Robotics and Automation*, vol. 8, pp. 313–326, June 1992.

[4] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE Journal of Robotics and Automation*, vol. RA-3, pp. 404–417, Oct. 1987.

[5] J. Feddema and O. Mitchell, "Vision-guided servoing with feature-based trajectory generation," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 691–700, Oct. 1989.

[6] P. Martinet, J. Gallice, and D. Khadraoui, "Vision based control law using 3d visual features," 1996.

[7] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, Oct. 1996.

[8] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The confluence of vision and control* (D. Kriegman, G. Hager, and S. Morse, eds.), vol. 237 of *Lecture Notes in Control and Information Sciences*, pp. 66–78, Springer-Verlag, 1998.

[9] N. Gans and S. Hutchinson, "An experimental study of hybrid switched system approaches to visual servoing," in *Proc. of the International Conference on Robotics and Automation, 2003*. May 2003.

[10] O. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment," 1988.

[11] Z. Zhang and A. Hanson, "reconstruction based on homography mapping," 1996. Zhang Z. and Hanson A.R. 3d reconstruction based on homography mapping. In ARPA Image Understanding Workshop, Palm Springs, CA, 1996.

[12] D. DeMenthon and L. S. Davis, "Model-based object pose in 25 lines of code," in *European Conference on Computer Vision*, pp. 335–343, 1992.

[13] P. Corke and S. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 4, pp. 507–515, 2001.

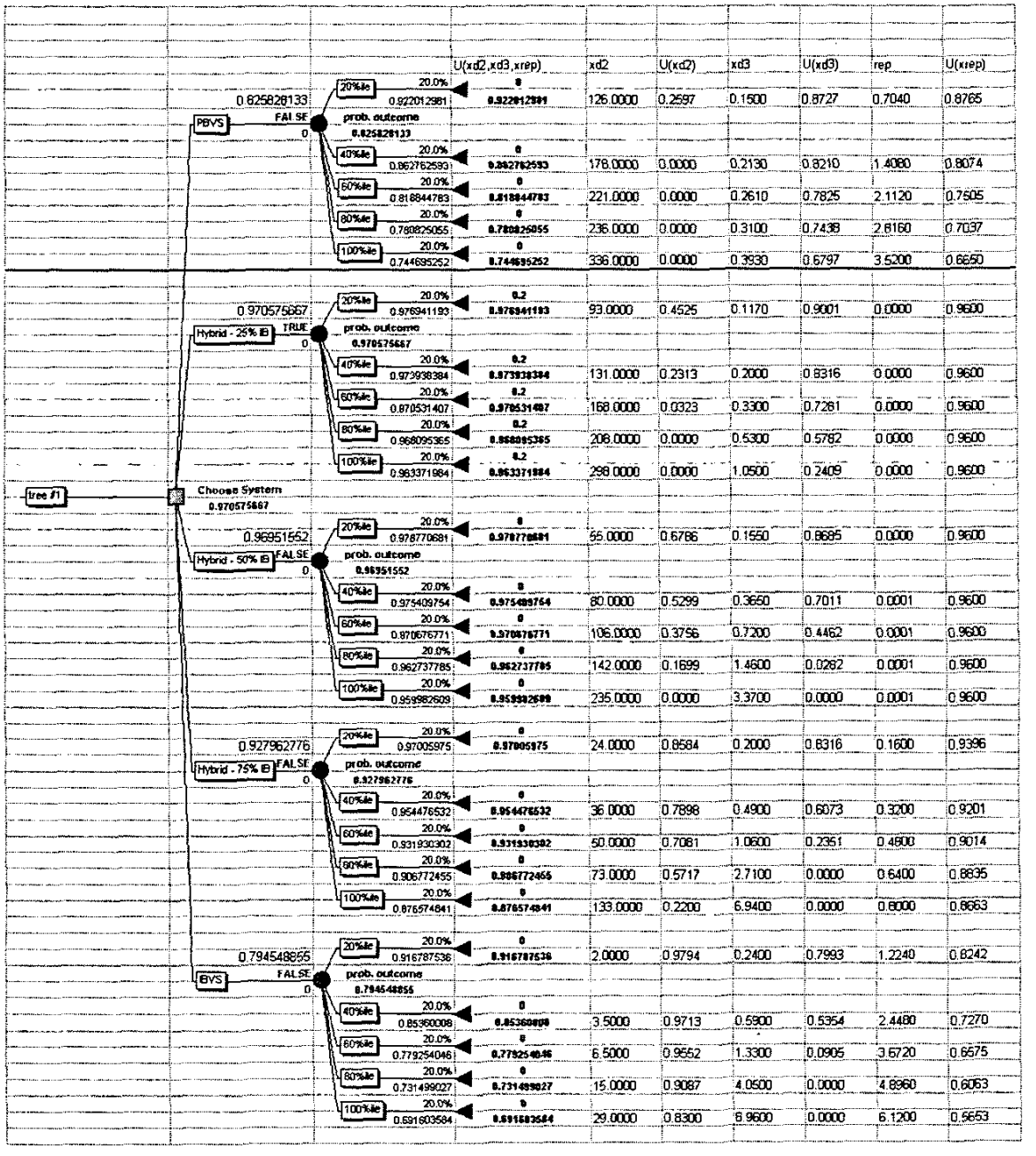
[14] N. R. Gans and S. A. Hutchinson, "A switching approach to visual servo control," in *to appear in Proc. IEEE Int'l Symp. on Intelligent and Control*, 2002.

[15] D. Thurston, "A formal method for subjective design evaluation with multiple attributes," *Research in Engineering Design*, vol. 3, pp. 105–122, 1991.

[16] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[17] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pp. 293–308, 2000.

Appendix



	U(xd2,xd3,xrep)	xd2	U(xd2)	xd3	U(xd3)	rep	U(xrep)
20%ile	0.922012981	126.0000	0.2597	0.1500	0.8727	0.7040	0.8765
40%ile	0.862762593	178.0000	0.0000	0.2130	0.8210	1.4080	0.8074
60%ile	0.818844783	221.0000	0.0000	0.2610	0.7825	2.1120	0.7505
80%ile	0.780825055	236.0000	0.0000	0.3100	0.7439	2.8160	0.7097
100%ile	0.744695252	336.0000	0.0000	0.3930	0.6797	3.5200	0.6650
20%ile	0.976941193	93.0000	0.4525	0.1170	0.9001	0.0000	0.9600
40%ile	0.973938384	131.0000	0.2313	0.2000	0.8316	0.0000	0.9600
60%ile	0.970531407	168.0000	0.0323	0.3300	0.7281	0.0000	0.9600
80%ile	0.968095365	208.0000	0.0000	0.5300	0.5782	0.0000	0.9600
100%ile	0.963371984	298.0000	0.0000	1.0500	0.2409	0.0000	0.9600
20%ile	0.978770681	55.0000	0.6786	0.1550	0.8695	0.0000	0.9600
40%ile	0.975499754	80.0000	0.5299	0.3650	0.7011	0.0001	0.9600
60%ile	0.970676771	106.0000	0.3756	0.7200	0.4462	0.0001	0.9600
80%ile	0.962737785	142.0000	0.1699	1.4600	0.0282	0.0001	0.9600
100%ile	0.959982609	235.0000	0.0000	3.3700	0.0000	0.0001	0.9600
20%ile	0.97005975	24.0000	0.8584	0.2000	0.8316	0.1600	0.9396
40%ile	0.954476532	36.0000	0.7898	0.4900	0.6073	0.3200	0.9201
60%ile	0.931930302	50.0000	0.7081	1.0600	0.2351	0.4800	0.9014
80%ile	0.906772455	73.0000	0.5717	2.7100	0.0000	0.6400	0.8835
100%ile	0.876574841	133.0000	0.2200	6.9400	0.0000	0.8000	0.8663
20%ile	0.916787536	2.0000	0.9794	0.2400	0.7993	1.2240	0.8242
40%ile	0.85360808	3.5000	0.9713	0.5900	0.5354	2.4480	0.7270
60%ile	0.779254046	6.5000	0.9552	1.3300	0.0905	3.6720	0.6575
80%ile	0.731499027	15.0000	0.9087	4.0500	0.0000	4.8960	0.6063
100%ile	0.691603584	29.0000	0.8300	8.9600	0.0000	6.1200	0.5653

Fig. 10. Decision Tree