Proceedings of the 1997 IEEE
International Conference on Robotics and Automation
Albuquerque, New Mexico - April 1997

Improving Task Reliability by Fusion of Redundant Homogeneous Modules Using Voting Schemes

Paolo Pirjanian[1], Jeffrey A. Fayman[2], Henrik I. Christensen[1]

[1]Laboratory of Image Analysis,
Aalborg University, Fredrik Bajers Vej 7D,
DK-9220 Aalborg East, Denmark
{paolo,hic}@vision.auc.dk

[2]Department of Computer Science,
Israel Institute of Technology – Technion,
Haifa 32000, Israel
jefff@.cs.technion.ac.il

Abstract

In this paper we present a technique that can help us design system components (behaviors) with reliability in mind. Our approach is similar to N-version programming [3]. A team of functionally redundant behaviors vote for a set of possible actions. A voter then fuses the votes and selects the action which has received the maximum number of votes. We draw some design guidelines and show that with careful design, the selected action will have higher probability of success (reliability) than the reliability of each of the behaviors. A comprehensive set of experiments together with results are presented in which a team of tracking behaviors control a stereo camera head in the active vision task of smooth pursuit. The results show that the approach dramatically improves tracking performance.

1 Introduction

Current robotic and vision systems are able to operate reliably in well structured environments. However, if they are placed in environments which they are not programmed to handle, they may fail completely. Reliability in robotics has been studied along several main paths: *reactivity*, *error recovery* and *uncertainty handling*[1]. Reactivity [1] provides immediate responses to unpredictable environmental changes through a tight coupling of perception and action. Reactive architectures have shown improved reliability when compared with classical sense-plan-act architectures. In error recovery techniques a set of dedicated modules monitor the task continuously and upon detection of an error (cognizant failure) an appropriate corrective action is invoked, which can handle unanticipated situations [4, 7]. The basic idea behind uncertainty handling techniques is to have explicit "models" of the robot's sensing and action capabilities. Using this model the agent (robot) can predict what actions to take in order to increase the expected utility or the reliability of its task [5].

In this paper, we present an approach for constructing reliable purposive modules or behaviors. The technique is based on the exploitation of redundancy in which a set of functionally equivalent (homogeneous) behaviors vote for a set of possible actions (e.g., move left, right, up, down). A "voter" then chooses the most appropriate action, corresponding to the action with the highest probability of success. The basic idea behind this approach is similar to that of sensor fusion with the hypothesis that the overall reliability will be improved by combining pieces of evidence provided by independent/partially independent sources. Fusion in this framework is done in a goal-directed manner, so that only the information, necessary to achieve the (specific) goal of a behavior, are fused.

The remainder of the paper is organized as follows. In section 2, we present our approach. In section 3, we present a set of experiments that validate the presented approach. We have implemented a team of homogeneous modules for performing smooth pursuit on a stereo robotic head, and measured system performance both with and without our technique. Our experiments show a dramatic increase in system performance when the voting technique is employed. We conclude with a discussion in section 4.

2 Behavior fusion approach

In the following section we formalize a behavior as a mapping that assigns preferences to each possible

[1]Uncertainty handling can be further divided into the two related areas of *explicit* uncertainty handling and methods exploiting *redundancy* (such as sensor fusion).

action. From a behavior's point of view, the action with the highest preference can best meet its objective(s). Then we define teams of behaviors, which have the same objective(s), and are committed to achieving/maintaining them through common effort, using voting. Subsequently a formal framework is presented that can allow us derive equations for the reliability of behaviors as well as behavior teams. These theoretical findings are then used to draw guidelines that can help us design teams of behaviors with improved reliability.

2.1 Behaviors & behavior teams

We formalize a behavior, b, as a mapping from an action space, Θ, to the interval $[0,1]$:

$$b : \Theta \rightarrow [0,1]. \tag{2.1}$$

The action space, $\Theta = \{\theta_1, \theta_2, ..., \theta_n\}$, is defined to be a finite set of possible actions. The mapping assigns to each action $\theta \in \Theta$ a preference, where the most appropriate actions are assigned 1 and undesired/illegal actions are assigned 0.

A set of functionally equivalent behaviors, that pursue the same common goal, will be denoted *homogeneous behaviors*[2]. A set of homogeneous behaviors that in combination pursue a specific goal will be denoted a *behavior team*, $\mathcal{BT}$, characterized by a tuple: $\mathcal{BT} = (\mathcal{B}, \Theta, \delta)$, where

- $\mathcal{B} = \{b_1, b_2, ..., b_n\}$ is a set of homogeneous behaviors,
- $\Theta = \{\theta_1, \theta_2, ..., \theta_k\}$ is an action space common to all $b \in \mathcal{B}$,
- $\delta : \Theta \rightarrow [0,1]$ is an operator for composition of the (outputs of the) behaviors, $\mathcal{B}$.

The outputs of the behaviors are combined using δ and the most appropriate action chosen is θ' where $\delta(\theta') = \max\{\delta(\theta)|\theta \in \Theta\}$.

2.1.1 Voting composition operators

Various composition operators have been suggested and studied in the literature [2]. This work is only concerned with voting schemes, which define a class of composition operators that are computationally efficient (see [6] for a survey). A general definition for a class of voting schemes, known as *weighted consensus voting*, is given by

[2]Apart from having the same goal these behaviors can be based on different sensing modalities, algorithms etc. to achieve the same goal.

DEFINITION 2.1 (m-OUT-OF-n VOTING) *An m-out-of-n voting scheme, $V : \Theta \rightarrow [0,1]$, where n is the number of homogeneous behaviors, is defined in the following way*

$$V(\theta) = \begin{cases} \Gamma(b_1(\theta), ..., b_n(\theta)) & \text{if } \sum_{i=1}^{n} v_i(\theta) \geq m; \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where

$$v_i(\theta) = \begin{cases} 1 & \text{if } b_i(\theta) > 0; \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 1, ..., n. \tag{2.3}$$

is the voting function and $\Gamma : [0,1]^n \rightarrow [0,1]$ is a function for combining the preferences of the behaviors for a given action.

A behavior gives a vote for an action if its preference for that specific action is > 0. If $\geq m$ behaviors vote for an action, θ, then their preferences are combined using Γ, which can be a simple averaging of the preferences or more sophisticated functions. Our design objective, outlined later, is to choose a voting scheme, δ, that satisfies the following requirement.

REQUIREMENT 1
A behavior team, $\mathcal{BT} = (\mathcal{B}, \Theta, \delta)$, is at least as reliable as the most reliable behavior, $b \in \mathcal{B}$ of the behavior team, i.e., $Rel(\delta(\mathcal{BT})) > \max\{Rel(b)|b \in \mathcal{B}\}$.

Where $Rel(b)$ is a measure of reliability defined as the portion of successful runs of a behavior, b, or behavior team, $\mathcal{BT}$. The requirement formalizes the hypothesis that the reliability of the system can be improved by fusing redundant behaviors in an appropriate way.

2.2 Behavior & behavior team reliability

Knowing the ground truth, i.e., the correct decision under given circumstances we can classify the outcome of each cycle of a behavior as either a "success" or as a "failure". We thus define the reliability of a behavior as its probability of success, $p = P\{\text{success}\}$.

The outcome of a behavior can be described by a Bernoulli random variable, X, in the following way. If we let $X = 1$ when the outcome of a behavior is successful and $X = 0$ if it is a failure, then the probability mass function of X is given by

$$\begin{aligned} P\{X = 1\} &= p \\ P\{X = 0\} &= 1 - p, \end{aligned} \tag{2.4}$$

where p is the reliability of the behavior in question. Now consider the behaviors $\mathcal{B}$ of a behavior team

as a system of $n = |\mathcal{B}|$ components, each of which has either a successful outcome or it fails. Then letting

$$x_i = \begin{cases} 1 & \text{if behavior } i \text{ is successful;} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

we call $\mathbf{x} = (x_1, x_2, ..., x_n)$ the state vector. Then we define a function $\Phi(x_1, x_2, ..., x_n)$ such that

$$\Phi(x_1, x_2, ..., x_n) = \begin{cases} 1 & \text{if } \mathcal{BT} \text{ is successful under} \\ & \text{state vector } (x_1, x_2, ..., x_n); \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

The function $\Phi(x_1, x_2, ..., x_n)$ is called the *structure function.* Suppose now that the outcome of the behaviors $b_1, b_2, ..., b_n$ are given by a set of Bernoulli random variables, $X_i, i = 1, ..., n$ as described in Equation (2.4). Then let

$$r(p_1, ..., p_n) = P\{\Phi(X_1, ..., X_n) = 1\}. \tag{2.7}$$

This function represents the probability that the behavior team is successful (i.e., the reliability of the behavior team) when the behaviors of the team have reliabilities $p_i, i = 1, ..., n$. Equation (2.7) is called the *reliability function.* The reliability of a behavior team depends on the reliability of the individual behaviors, the number of behaviors, $|B|$, their statistical dependence and the criteria for selecting the appropriate action. The latter part is determined by the voting scheme (specifically the parameters m and n), whereas the others are either given parameters or known/assumed properties in specific situations. Thus to calculate the reliability function we need the structure function which depends on the voting scheme.

2.2.1 k-out-of-n systems

The general expression for the reliability can be facilitated by the following definition.

DEFINITION 2.2 (k-OUT-OF-n SYSTEM) *A system of n behaviors is called a k-out-of-n system if it is successful only when at least k out of its n behaviors are successful.*

The structure function of a k-out-of-n system is given by

$$\Phi(x_1, x_2, ..., x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i \geq k; \\ 0 & \text{otherwise} \end{cases}$$

The reliability function of a k-out-of-n system can then be calculated by

$$\begin{aligned} r(p_1, ..., p_n) &= P\{X_i = 1 \text{ for at least } k \text{ behaviors} \\ &\quad i = 1, ..., n\} \\ &= \sum_{l=k}^{n} \sum_{\mathbf{x} \in \Omega(l,n)} \psi(\Delta_1(\mathbf{x}), ..., \Delta_n(\mathbf{x})), \\ &\quad \text{for } k = 1, ..., n \end{aligned}$$

where $\psi(\mathbf{x})$ is the probability that the outcome is $\mathbf{x} \in \Omega(k, n)$, $\Omega(k, n)$ is the set of all the permutations of (the outcome of) n behaviors where k of them are successful, 1, and $n - k$ are not, 0. The function $\Delta : \{0, 1\}^n \rightarrow [0, 1]$ is defined in the following way

$$\Delta_i(x) = \begin{cases} p_i & \text{if } x(i) = 1; \\ 1 - p_i & \text{otherwise} \end{cases} \tag{2.8}$$

To investigate under which conditions Requirement 1 is satisfied the following inequality has to be solved

$$\sum_{l=k}^{n} \sum_{\mathbf{x} \in \Omega(l,n)} \psi(\Delta_1(\mathbf{x}), ..., \Delta_n(\mathbf{x})) > \max\{p_1, ..., p_n\} \quad \text{for } k = 1, ..., n \tag{2.9}$$

Solution to this inequality is, however, conditioned on knowledge about the statistical dependence among the behaviors and the reliability of the behaviors. An analytical solution can not be given to Equation (2.9) because it is ill-posed in its general form. The following theory which applies to the general form of Equation (2.8) can help us draw design guidelines for choosing an appropriate system.

THEOREM 2.1
For a given behavior team, the reliability of a k-out-of-n system is greater than or equal to the reliability of a $k+1$-out-of-n system. Formally

$$r(k) \geq r(k+1) \ , \ \textit{for } k = 1, ..., n \tag{2.10}$$

where for convenience we denote the reliability function of a k-out-of-n system $r(k)$.

Theory 2.1 states that that system reliability increases as k decreases (proof omitted).

2.3 Design guidelines

Theorem 2.1 inductively implies that a 1-out-of-n system will maximize the reliability function. It is, however, important to note that the choice of a voting scheme should not only depend on maximizing the reliability function but also on other properties such

as *probability of decision* which we denote *stability*. It can be shown that system stability decreases as k decreases. In conclusion the reliability function increases and the stability decreases as the k value decreases. Thus it is clear that in choosing an appropriate voting scheme a trade-off must be made between increasing the reliability and increasing the stability of the system. Hence, given a set of behaviors $\{b_1, ..., b_n\}$ with reliabilities $p_1, ..., p_n$ and a minimum desired value of reliability $r_{\min}$ a procedure for choosing a voting scheme could be:

1. If $r_{\min} > \max r(p_1, ..., p_n)$ goto step 5.
2. Choose a k-out-of-n system defined by a m-out-of-n voting scheme, where $k = n - m + 1$, with the lowest k that fulfills equation (2.9). Goto step 4.
3. Increase k by 1.
4. If $r(p_1, ..., p_n) < r_{\min}$ goto step 3 else goto step 5.
5. Stop.

Step 1 checks if it is at all possible to achieve the minimum required reliability $r_{\min}$ for the given set of behaviors. If not the reliability can be improved by either adding a new behavior or improving the reliability of the behaviors already in the team or both. Step 2 chooses a voting scheme that satisfies Requirement 1 for highest possible stability. If this case does not fulfill one's minimum requirement for reliability (step 4) then step 3 trades off stability for higher reliability.

3 Experimental validation

The goal of our experiments is to investigate the hypothesis that fusion of homogeneous behaviors can lead to improved reliability, thus four modules, for smooth pursuit, are implemented as well as a module which fuses their results, using a voting scheme. Smooth pursuit is the process by which a moving object is tracked. The goal is to keep the retinal position of the moving object in the fovea. The four implemented behaviors for smooth pursuit are:

- Blob Tracking (BLOB): In our implementation, the input image is thresholded to remove any spurious pixels, then the centroid of the above threshold pixels is computed.
- Image Differencing (IDIFF): Image differencing is performed by differencing two consecutive frames. Differencing segments the scene into static and moving regions as only objects that have changed position between two consecutive images will have non zero pixel values.
- Edge Tracking (SOB): Edge tracking is similar in nature to blob tracking, however, rather than finding the centroid of the entire blob, the centroid of blob edges is found. In order to find the edges, an edge operator is used on the input image.
- Template Matching (TM): The idea behind template matching is to detect a particular object in an image by searching for instances of a separate sub-image (template) which looks like the desired object. Correlation provides the basis of template matching. The template location which provides the maximal similarity measure is selected as the location of the object in the image.

The output of each tracking behavior is a triangular function (see figure 3) with the peak at the image position of the tracked object. The voter in these experiments (denoted FUSE) implements a 2-out-of-4 voting scheme.

3.1 Experimental setup

In each experiment, a robotic manipulator, holding the object to be tracked, effects horizontal translatory motion consisting of two motion segments: from the starting location to a location 200cm to the right and return to the starting location. In the experiments, we measured the ability of each module to track various combinations of objects and backgrounds. We call each such combination a *scenario*. A subset of the scenarios used in our experiments is shown in figure 1.

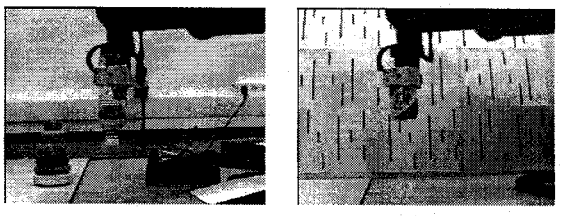
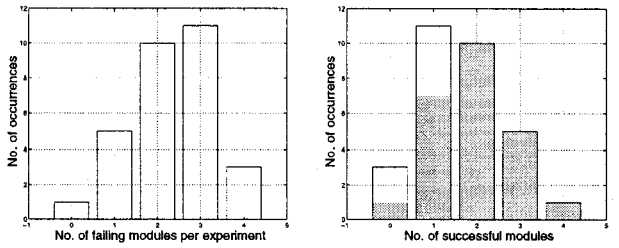


Figure 1: Scenarios used in experiments. Various combinations of objects and backgrounds provide differing complexities.

A total of 30 *experiment sets* were conducted using 6 scenarios, where each experiment set consisted of testing each of the four motion tracking modules plus the fusion module on a scenario. To account for variations in lighting and other conditions, each scenario was used for 5 experiment sets. The scenarios were chosen so as to push the limits of one or more of the modules at a time so that they would fail. The number of failing modules, thus defines an index for the complexity of a scenario. E.g., if scenario, a, leads to the failure of 3 of the modules and another scenario, b, leads to the failure of 1 module, then we say that scenario a is more complex than scenario b. The histogram in figure 2(a) shows the distribution of the number of failed modules for each of the 30 experiments.



(a) (b)

Figure 2: (a) Distribution of the number of failing modules at each trial during 30 trials. The fusion module is not taken into consideration here. Scenario complexity increases from left to right. (b) Distribution of the number of successful modules at each trial during 30 trials and relation to the number of times fusion succeeds. The filled areas correspond to the portion of times where fusion succeeds when only the given number of modules succeed in an experiment. Scenario complexity decreases from left to right.

Two measures are used to quantify each module's ability to track: *absolute error* and *relative error*. Absolute error is a yes/no answer to the question of whether a module successfully tracks during a single experiment. A module is said to track an object successfully if some part of the object is located at the image center during the entire motion sequence. The ratio of the number of successful runs to the total number of runs provides a measure of module reliability.

Relative error quantifies the quality of tracking, i.e., a measure of how well tracking is performed. As an expression for relative error we use the distance (in pixels) from the image center (where the tracked object should be in the ideal case), to a fixed point on the moving object. We term this expression *distance error*. We also separate this distance into its X (horizontal) and Y (vertical) components, to investigate the contributions of these components to the error[3]. Here we present the mean and standard deviation of the distance error, along with the mean and standard deviations of the corresponding X and Y components, for the 30 experiments (see table 2).

3.2 Experimental results

Absolute error results are listed in table 1 along with module reliabilities. Module reliabilities, listed in the last column, are calculated as the ratio of the number of successful runs to the total number of runs (30). Table 2 summarizes the relative error results, and in particular the mean and standard deviations of the distance error. The table also lists for both X and Y the

[3]To obtain the distance error, we recorded each motion sequence to video tape which was analyzed manually to obtain the ground truth (i.e., where the object was in the image) as a reference.

Module	# Successes	# Failures	Reliability
Blob Track	11	19	**36.7%**
Image Diff	14	16	**46.7%**
Edge Track	12	18	**40.0%**
Temp Match	13	17	**43.3%**
Fusion	24	6	**80.0%**

Table 1: **Absolute error**. Module success and failure rates. Reliability is calculated as the ratio of successful runs to the total number of experiments. The main results are highlighted.

mean value of distance error along with corresponding standard deviation.

The plot in figure 2(b) illustrates the performance of the fusion module as a function of scenario complexity. In three of the experiments none of the modules succeed, nonetheless the fusion module (surprisingly) succeeds in one of these three cases. Note also that in 11 of the cases only one module succeeds, but in 7 out of the 11 cases the fusion module manages to track successfully. How the fusion can succeed even though all or the majority of the individual modules fail is interesting and will be explained in the remainder of this section.

Figure 3 contains plots of the actual outputs generated during one experiment. The plots show the outputs generated by the five modules over 151 frames (only frames 41 to 91 are shown). The output of each module is illustrated as a triangular window of width equal to 10 pixels. Only the plots for the X-axis are shown. The output of each module should be interpreted as the vote for moving the image center to a given pixel. The commanded output is given to the camera head driver, which translates the pixel values to joint angles and drives the motors. The last plot within each subplot is the output of the fusion module, which combines the votes received from the individual modules and chooses the action with the maximum vote, indicated with the dot in the figures.

In frame 51 it is seen that IDIFF and TM lose track of the object while fusion tracks based on information provided by BLOB and SOB. In frame 71, TM resumes tracking, and in frame 81, IDIFF resumes tracking. Additionally, in subsequent frames it seems that BLOB drifts away and later resumes tracking.

During the experiments, if run independently IDIFF fails to track the object through the entire experiment. However, in figure 3 it is seen that IDIFF can resume tracking, when run in conjunction with the other modules. When run independently, if a module loses track of the object, due to some artifact in the image which confuses the algorithm, it may not have the

Module	Mean X	Std.Dev X	Mean Y	Std.Dev Y	Mean	Std.Dev.
Blob Tracking	136.1	188.9	110.0	198.1	**185.9**	**266.4**
Image Differencing	44.0	86.4	12.8	48.8	**49.9**	**97.3**
Edge Tracking	72.1	109.0	25.9	92.7	**82.6**	**139.7**
Template Matching	76.4	125.1	30.2	111.8	**88.6**	**164.4**
Fusion	17.2	24.1	5.2	25.8	**19.8**	**34.3**

Table 2: **Relative error.** Module performance results for 30 runs. Relative error is presented by the mean distance error and standard deviation. The statistics for the corresponding X and Y components are also listed. The main results are highlighted.

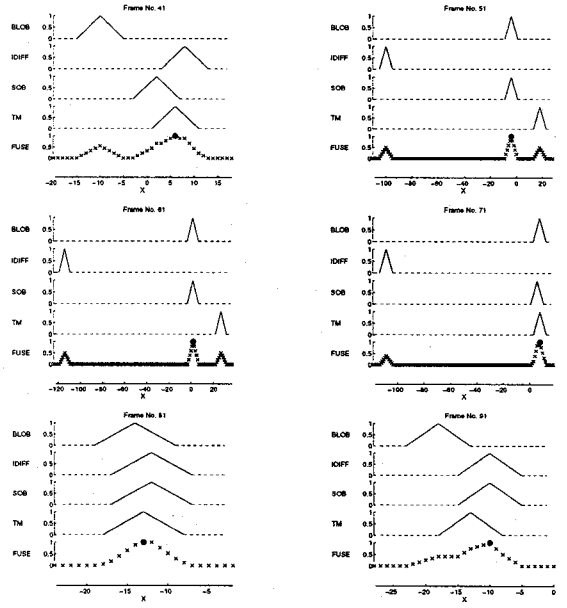


Figure 3: Sequence showing the tracking results and the fusion process. During the sequence the camera is driven by the fusion module. The plots present each modules votes for each action (here motion in the horizontal direction). These votes are combined by the fusion module (FUSE) and the best action indicated by the dot in the plots is selected.

possibility to correct itself, because losing the object eventually causes the object to leave the region of interest (fovea) or even the image. However, with fusion this problem can be corrected, as other modules may continue to drive the object into the region of interest, where the object/motion is searched for, giving a failing module the opportunity to regain tracking. This will only work if the cause for failure of the modules is disjunct, so that the rarely simultaneously.

4 Concluding remarks

We have introduced an approach, where reliable behaviors can be constructed by a careful integration of less reliable ones. A set of experiments in smooth pursuit were conducted to confirm that the performance of several integrated motion analysis modules is better than the performance of any of the participant modules. As one of the peer reviewers points out "*A reliability of* 80% *does not seem so spectacular ...*', but the point is that due to the complementarity of the behaviors the fusion module is able to handle more scenarios than any other behavior. We have not quantified the specific reasons for the failure of the fusion module. This, however, does not change the conclusion that the presented technique can improve task reliablility.

5 Acknowledgments

This work is supported in part by ECIS. The experiments were conducted at the Intelligent Systems Laboratory, Israel Institute of Technology. Thanks to Dr. E. Rivlin for providing the equipment and other lab. facilities. Special thanks are due to Prof. E. Granum for his support and encouragement of this work. Also thanks to the reviewers for providing us with useful comments.

References

[1] Ronald C. Arkin. Integrating Behavioral, Perceptual and World Knowledge in Reactive Navigation. *Robotics and Autonomous Systems*, 6:105–122, June 1990.

[2] I. Bloch. Information Combination Operators for Data Fusion: A Comparative Review with Classification. *IEEE transactions on Systems, Man, And Cybernetics, Part A: systems and humans*, 26(1):42–52, January 1996.

[3] J. Dugan and M. Lyu. System-reliability analysis of an n-version programming application. *IEEE Transactions of Reliability*, 1994.

[4] Erann Gat and Greg Dorais. Robot Navigation by Conditional Sequencing. *IEEE Int. Conf. on Robotics and Automation*, 2:1293–1299, 1994.

[5] Steen Kristensen. Sensor planning with Bayesian decision theory. In *Reasoning with Uncertainty in Robotics*. University of Amsterdam, The Netherlands, December 1995.

[6] Behrooz Parhami. Voting Algorithms. *IEEE Transactions on Reliability*, 43(3):617–629, December 1994.

[7] Reid G. Simmons. Structured Control for Autonomous Robots. *IEEE Trans. on Robotics and Automation*, 10(1):34–43, February 1994.