Choosing Good Distance Metrics and Local Planners for Probabilistic Roadmap Methods

Nancy M. Amato, O. Burchan Bayazit, Lucia K. Dale, Christopher Jones, Daniel Vallejo

Abstract — This paper presents a comparative evaluation of different distance metrics and local planners within the context of probabilistic roadmap methods for planning the motion of rigid objects in three-dimensional workspaces. The study concentrates on cluttered three-dimensional workspaces typical of, e.g., virtual prototyping applications such as maintainability studies in mechanical CAD designs. Our results include recommendations for selecting appropriate combinations of distance metrics and local planners for such applications. Our study of distance metrics shows that the importance of the translational distance increases relative to the rotational distance as the environment becomes more crowded. We find that each local planner makes some connections that none of the others do — indicating that better connected roadmaps will be constructed using multiple local planners. We propose a new local planning method we call `rotate-alt' that often outperforms the common straight-line in C-space method in crowded environments.

Keywords — Motion Planning, Probabilistic Roadmaps, Distance Metrics, Local Planners

I. INTRODUCTION

Automatic motion planning has application in many areas such as robotics, virtual reality systems, and computer-aided design. Although many different motion planning methods have been proposed, most are not used in practice since they are computationally infeasible except for some restricted cases, e.g., when the robot has very few degrees of freedom (dof) [12], [16]. For this reason, attention has focused on randomized methods, such as randomized potential field methods (e.g., [5]).

Recently, a class of randomized motion planning methods, called probabilistic roadmap methods (PRMs), has gained much attention (see, e.g., [1], [4], [11], [14], [22]). These methods use randomization during pre-processing to construct a graph in C-space (a roadmap). Queries are processed by connecting the initial and goal configurations to the roadmap, and then finding a path in the roadmap between these two connection points. PRMs have been shown to perform well in practice, answering difficult queries in fractions of seconds [4], [14].

PRM roadmap construction is generally performed in two stages: node generation and connection. In the node generation stage, collision-free configurations of the robot are generated according to some sampling strategy in C-space (e.g., uniformly [14], near constraint surfaces [4], [6], or on the medial axis [22]). In the connection stage, connections (edges) are made between node pairs if a path connecting them can be found by a `local' planning method.

Although PRMs may vary in terms of high-level node generation and connection strategies, they all make heavy use of primitive operations such as collision detection, local planners (for connecting roadmap nodes), and distance computations (to select which connections to attempt, since it is infeasible to try them all). Thus, the choice of appropriate techniques for these operations can crucially impact the efficiency and success of the PRM.

This paper presents a comparative evaluation of distance metrics and local planners in the context of PRMs. Our study concentrates on motion planning for rigid objects in cluttered three-dimensional workspaces typical, e.g., of virtual prototyping applications such as maintainability studies in mechanical CAD designs [7]. Such applications present difficult motion planning problems which typically require navigating narrow passages in the configuration space [11]. We believe such problems are amenable to solution by PRMs — but this will require a fine-tuning of every PRM component, including primitive operations such as distance computations and local planners.

II. DISTANCE METRICS

Distance metrics are used in PRMs to determine which nodes one should attempt to connect using a local planner. A good metric will limit the number of calls to the local planner by classifying enough connectable nodes as close. It must also be fast to compute, since distance calculations are one of the most numerous operations in a PRM.

A. C-Space Representation

We represent configurations of rigid objects in three-space by six-tuples \( c_i = (x_i, y_i, z_i, \theta_1^i, \theta_2^i, \theta_3^i) \), where the first three coordinates define the position and the last three the orientation in Euler Angles. The orientation coordinates are represented as values in \([0, 1]\). Orientational differences are measured in the shortest direction. To obtain generalizable results, we normalize the orientation coordinates (range \([0, 1]\)) with respect to the position coordinates (no fixed range); see [2] for details.
TABLE I  
DISTANCE METRICS STUDIED

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description of $d(c_1,c_2)$</th>
<th>m/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>$(\sum_{i=1}^3 p_i^2 + \sum_{i=1}^3 q_i^2)^{\frac{1}{2}}$</td>
<td>4.6</td>
</tr>
<tr>
<td>Scaled Euclidean</td>
<td>$(\sum_{i=1}^3 p_i^2 + (1 - \theta) \sum_{i=1}^3 q_i^2)^{\frac{1}{2}}$</td>
<td>4.7</td>
</tr>
<tr>
<td>Minkowski</td>
<td>$(\sum_{i=1}^3 p_i^2 + \sum_{i=1}^3 q_i^2)^{\frac{1}{2}}$</td>
<td>11.6</td>
</tr>
<tr>
<td>Mod. Minkowski</td>
<td>$(\sum_{i=1}^3 p_i^2 + \sum_{i=1}^3 q_i^2)^{\frac{1}{2}}$</td>
<td>24.2</td>
</tr>
<tr>
<td>Manhattan</td>
<td>$\sum_{i=1}^3</td>
<td>p_i + \sum_{i=1}^3 q_i</td>
</tr>
<tr>
<td>Bounding Box</td>
<td>max. dist. between BB vertices</td>
<td>33.7</td>
</tr>
</tbody>
</table>

$P_i = |c_2(x_i) - c_1(x_i)|$, and $Q_i = n|c_2(\theta_i) - c_1(\theta_i)|$, where $i \in \{1,2,3\}$, $n$ is the normalization factor.

B. Distance metrics evaluated

Our study considered five C-Space and two workspace distance metrics. See Table I; times shown are averages of 10,000 computations.

Five C-space metrics were selected for evaluation based on effectiveness and efficiency concerns (see, e.g., [10, 20]). The Euclidean distance in C-space considers C-Space as a Cartesian space and gives both position and orientation the same importance. The scaled Euclidean distance changes the relative importance of the position and orientation components through the scale parameter $s$. We note that the value of the metric parameters will depend on the representation chosen for C-space. The Minkowski distance is the generalization of the Euclidean distance which uses a parameter $r$ in place of the $2$; as with Euclidean, both position and orientation are given the same importance. Our so-called modified Minkowski metric distinguishes between the position and orientation coordinates using the parameters $r_1$ (position) and $r_2$ (orientation). The Manhattan metric is the usual Manhattan distance in $\mathbb{R}^6$. Note the Minkowski metric approaches the Manhattan distance as $r$ tends to infinity.

The workspace metrics we chose to study are both simple metrics based on Euclidean distances in the workspace. The first is the Euclidean distance between the center of mass of the robot in the two configurations (the center of mass is approximated by averaging all object vertices). Although this is a very rough estimation, it is fast and it gives physical distance in workspace. The second workspace metric makes use of the bounding box of the robot, and finds the maximum Euclidean distance between any vertex of the bounding box in one configuration and its corresponding vertex in the other configuration. Note that while the center of mass metric is invariant with respect to orientation, the bounding box metric is not. Hence, while the scaled Euclidean and modified Minkowski metrics enable us to examine the importance of translation and rotation in C-Space, these metrics will help us study these issues in the workspace.

Although there are many other types of metrics, such as Riemannian metrics [20], Hausdorff distance [10], rotation distance [23], growth distance [18], or the minimum distance between any point on the robot in the two configurations (e.g., [8, 19]), they were not selected because they were either too expensive for use in PRMs or were only applicable to convex objects.

III. LOCAL PLANNERS

Local planners are used in PRMs to make connections between nodes when building the roadmap. They must be fast (so many connections can be attempted) and deterministic (so paths don’t have to be saved).

A. Local Planner Resolution Issues

Start and goal configurations of the robot are denoted by $c_1$ and $c_2$, respectively. The paths tested by our local planners consist of a sequence of configurations that differ from their predecessors and successors by at most some fixed resolution (in at least one coordinate). The resolution differed for position and orientation coordinates, and also varied according to the environment. Given $(c_1, c_2)$, and a resolution for each coordinate, our planners calculate an increment vector $I$ which is used to determine the neighboring configurations tested by the planners.

B. Local planners evaluated

Our first planner is the common straight-line in C-space method (see, e.g., [14]), which interpolates without bias along a six-dimensional straight line from configuration $c_1$ to $c_2$, and checks all points at some fixed resolution on that line for collision.

We call our second (parameterized) family of local planners rotate-at-$s$, $0 \leq s \leq 1$. This planner first translates from $c_1$ to an intermediate configuration $c'$, rotates to a second intermediate configuration $c''$, and finally translates to $c_2$. The parameter $s$ represents the fractional part of the translational distance between $c_1$ and $c_2$ that the robot travels from $c_1$ to $c'$. The straight-line planner is used to plan between $(c_1,c')$, between $(c',c'')$, and between $(c'', c_2)$. Note that the paths often will have smaller swept-volumes than the straight-line paths. Our study considered $s = 0, \frac{1}{2}, 1$.

We also study two A*-like planners: A*-clearance and A*-distance. The basic A* strategy is to compute a set of neighbors of $c_1$, select the most promising neighbor $c'$, and iterate with $c'$. The process terminates when $c_2$ is reached, no further movement is possible, or after some set number of iterations (see, e.g., [15, 21]). To make our A*-like methods faster, we limit the number of steps to some small multiple (e.g., 6) of the steps taken by the straight-line planner, and we consider just three neighbors: configurations where (i) both the position and orientation, (ii) only the position, and (iii) only the orientation, coordinates are incremented towards the goal. Both methods move to neighbor (i) if it is collision-free. Otherwise, A*-clearance selects the neighbor with maximum clearance from the workspace obstacles, and A*-distance selects the neighbor that has minimum distance to the goal.
IV. EXPERIMENTAL DESIGN

Our experiments were coded in C++ and conducted on SGI O2 workstations. Although there exist several collision detection algorithms ([9], [13], [17]), we used the C-Space Toolkit [21] since it was available to us and satisfied our computational requirements. In this paper, we show only an abstract of the results. Complete experimental results with more environments can be found in [2]. Intuitively, we expect the A* like planners to make more connections, but to require more time, than the other local planners. We also expect the relative importance of the rotational and translational distances to depend on the environments.

A. Environments

Our study considers two basic environments representative of cluttered three-dimensional workspaces, and three difficulty levels of each basic environment.

The first environment consists of seven unit cubes (12 triangles each); six obstacle cubes and one movable cube (see Fig. 1(a)). The centers of the obstacle cubes were placed on three parallel planes, one each on the front and back planes, and four in the middle plane, arranged so that they surround a cubical region centered on the middle plane. The hardness of the problem was controlled by varying the distance between the front, middle, and back planes: 1.1 (hard), 1.6 (moderate), and 2 (easy).

The second environment contains two twisted (α shaped) tubes (1008 triangles each); one obstacle tube and one movable tube (see Fig. 1(b)). The reader might be familiar with the puzzle where the objective is to separate the intertwined tubes. The ‘hardness’ of this problem was controlled by scaling the obstacle tube in one dimension, which widened the gap between the two prongs of the α. The scale factors were 1 (hard), 1.5 (moderate), and 2.5 (easy).

To calibrate the relative difficulty of our environments we compared the total number of connections made in each version. Table III shows these statistics for three local planners; the other planners showed similar trends.

B. Experiments

Our experiments were designed to: (i) select parameters for distance metrics, (ii) select metrics for local planners, (iii) select local planners for environments, and (iv) study the benefits of using multiple local planners.

We generated 600 free configurations (RandCfgs) as test nodes; 50 of the TestCfgs were near C-obstacle surfaces and 50 were generated at random (generally not near C-obstacles, referred to as free configurations). We used the method described in [3] to generate configurations near the surfaces of C-Space obstacles, and the method described in [14] for free space nodes.

For each local planner, we tried to connect each configuration in TestCfgs to every configuration in RandCfgs. To rate a metric (for a given local planner), we computed the distances using that metric between each node in TestCfgs and every node in RandCfgs, sorted these distances, and analyzed those connections made to the closest k nodes. In [2], we analyze results for k = 25, 50, 100; in this paper, we show only the results for k = 25 since they are the most relevant to PRMs.

V. EXPERIMENTAL RESULTS

A. Computational requirements

Since PRMs perform a large number of distance computations and local planning queries, their computation times are important factors to consider when choosing among them. Distance metric evaluation times are shown in Table I; values are averages of 10,000 computations.

Some times for representative local planning queries are shown in Table II; values shown are averages of approximately 100 computations. For these tests we considered the hard alpha-alpha environment. We generated 10 StartCfgs of the robot on C-obstacle surfaces. For each c, ε StartCfgs we randomly generated 20 GoalCfgs at approximately the same distance from c, (so most GoalCfgs are removed from C-obstacle surfaces). We then tried to connect c, to each c, ε GoalCfgs using each local planner. We report the successful and unsuccessful queries separately in Table II. Running times are averages over all relevant connection attempts. The average number of collision detection calls for the set of 200 local planning queries

---

**TABLE II**

<table>
<thead>
<tr>
<th>LOCAL PLANNER</th>
<th>Successful Time (μsec)</th>
<th>Failed Time (μsec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>start-straight-line</td>
<td>250.70</td>
<td>9000</td>
</tr>
<tr>
<td>rotate-at-0</td>
<td>50.70</td>
<td>8022</td>
</tr>
<tr>
<td>rotate-at-1</td>
<td>508.40</td>
<td>9982</td>
</tr>
<tr>
<td>A*-clearance</td>
<td>107940</td>
<td>22841</td>
</tr>
<tr>
<td>A*-distance</td>
<td>906.70</td>
<td>25321</td>
</tr>
</tbody>
</table>

**TABLE III**

<table>
<thead>
<tr>
<th>ENV./PLANNER</th>
<th>Easy</th>
<th>Mod</th>
<th>Hard</th>
<th>Succ.</th>
<th>Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>cube/Str-Line</td>
<td>20.2%</td>
<td>14.2%</td>
<td>10.0%</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>cube/Hot-1/2</td>
<td>22.3%</td>
<td>16.1%</td>
<td>11.4%</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>cube/α-clear</td>
<td>24.1%</td>
<td>16.1%</td>
<td>12.3%</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>alpha/Str-Line</td>
<td>11.9%</td>
<td>10.3%</td>
<td>8.6%</td>
<td>1065</td>
<td>56</td>
</tr>
<tr>
<td>alpha/K-at-1/2</td>
<td>14.2%</td>
<td>13.8%</td>
<td>11.8%</td>
<td>1065</td>
<td>50</td>
</tr>
<tr>
<td>alpha/α-clear</td>
<td>25.6%</td>
<td>22.3%</td>
<td>19.7%</td>
<td>3785</td>
<td>170</td>
</tr>
</tbody>
</table>
TABLE IV

<table>
<thead>
<tr>
<th>ENVIR</th>
<th>BEST SCALLED EUCLIDEAN VALUES</th>
<th>SURFACE TO CONNECTIONS</th>
<th>FREE TO SURFACE CONNECTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>STRL</td>
<td>R-AT-</td>
<td>A-CLEAR</td>
</tr>
<tr>
<td></td>
<td>Score</td>
<td>Score</td>
<td>Score</td>
</tr>
<tr>
<td>6-cube easy</td>
<td>2.71</td>
<td>2.82</td>
<td>3.38</td>
</tr>
<tr>
<td>6-cube mod</td>
<td>2.86</td>
<td>2.98</td>
<td>3.13</td>
</tr>
<tr>
<td>6-cube hard</td>
<td>1.23</td>
<td>1.54</td>
<td>1.84</td>
</tr>
<tr>
<td>alphas easy</td>
<td>2.35</td>
<td>1.00</td>
<td>4.71</td>
</tr>
<tr>
<td>alphas mod</td>
<td>1.44</td>
<td>0.91</td>
<td>3.90</td>
</tr>
<tr>
<td>alphas hard</td>
<td>1.50</td>
<td>0.83</td>
<td>3.56</td>
</tr>
</tbody>
</table>

TABLE V

<table>
<thead>
<tr>
<th>DISTANCE METRIC RECOMMENDATIONS BY PLANNER/ENV</th>
<th>METRIC RECOMMENDATIONS (BEST PERFORMER IN BOLD)</th>
<th>good</th>
<th>easy</th>
<th>hard</th>
</tr>
</thead>
<tbody>
<tr>
<td>STRL</td>
<td>cube</td>
<td>alpha</td>
<td>SE, MM</td>
<td>SE(50)</td>
</tr>
<tr>
<td>R-AT-</td>
<td>cube</td>
<td>SE, MM</td>
<td>SE(50)</td>
<td>SE, MM, CM</td>
</tr>
<tr>
<td>A-CLEAR</td>
<td>cube</td>
<td>SE, MM</td>
<td>SE(50)</td>
<td>SE, MM, CM</td>
</tr>
</tbody>
</table>

B. Selecting parameters for distance metrics

Three of our metrics require parameters: scaled Euclidean (s), Minkowski (r), and modified Minkowski (r_1, r_2, and r_3). To determine good values for these parameters, we tested several and selected the best for further evaluation.

For each local planner, we evaluated each distance metric parameter. We defined a scoring mechanism intended to give higher scores to parameter values which identified more "close" nodes that could be connected by that local planner. For each TestCfg, the RmpCfgs were partitioned into five sets according to relative distance as determined by the metric/parameter(s) under consideration: 1 (closest 10), 2 (next 15), 3 (next 25), 4 (next 50), and 5 (remaining 500). The score of a metric/parameter(s) for a specific local planner was

\[ 4^{(n_1/10)} + 3^{(n_2/15)} + 2^{(n_3/25)} + 1^{(n_4/50)}, \]

where \( n_i \) is the number of connections that planner made in set \( i \). While different scoring mechanisms may lead to different recommendations, this one was selected based on our experience with PRMs.

A selected summary of our results for the scaled Euclidean and Minkowski metrics is contained in Table IV. A general observation was that the relative importance of the translational (rotational) distance between the two configurations increased (decreased) as the environments became harder. Also, we noted that the straight-line and rotate-at-\( 1/2 \) were similar, as were the two A*-like planners.

For the scaled Euclidean metric, the optimal \( s \) value increases as the environment gets harder for all local planners and environments. For the surface to surface connections, the optimal \( s \) is usually higher in the alpha puzzle than in the 6-cube environments. Overall, \( s = .75 \) and \( s = .90 \) performed well. Our results for the Modified Minkowski values were generally supportive of these results as well.

For the Minkowski metric in the 6-cube environments, most local planners reached their peak at \( r = 1.5 \) for surface to surface connections, and \( r = 4 \) for free to surface connections. In the alpha puzzle environments, the best values were \( r = 1.5 \) for surface to surface connections and \( r = 2.5 \) or \( r = 3 \) for free to surface connections. We note the optimal parameter for a particular metric will depend on the representation chosen for C-Space. However, the general trend should be invariant to this choice.

C. Selecting metrics for local planners

After selecting interesting parameter values for the various metrics, we were left with a total of 12 different distance metrics for each environment. All four non-parameterized metrics (Euclidean, Manhattan, Center of Mass, and Bounding Box) were analyzed in both environments. In the 6-cube environment, we selected the Scaled Euclidean with \( s = \{ .25, .5, .75, .9 \} \), the Minkowski with \( r = \{ 1.5, 4 \} \), and the Modified Minkowski with \( (r_1, r_2, r_3) = \{ (2, 0.5, 2), (2, 2.5, 2) \} \). In the alpha puzzle environment, we selected Scaled Euclidean with \( s = \{ .25, .75, .9 \} \), Minkowski with \( r = \{ 1.5, 2.5 \} \), and Modified Minkowski...
with \((r_1, r_2, r_3) = \{(2, 1, 5, 2), (2, 2, 5, 2), (2, 5, 2, 2)\}\). In each of the six environments, each local planner was evaluated with the 12 selected metrics.

A selected summary of distance metric recommendations for three local planners in the easy and hard environments is shown in Table V. Generally, the best metrics placed more importance on the translational distance than on the rotational distance. Our recommendations take both efficiency and effectiveness into account (we defined efficiency as the speed of a distance metric and effectiveness as its score, i.e., the characterization of more connectable nodes as close). When results differed for free to surface and surface to surface connections, preference is given to surface to surface connections as those are considered most difficult. When the best metric is computationally expensive, we suggest a more efficient alternative (with comparable effectiveness). For example, we can be seen in Table V that in many cases when the best performer was the Minkowski or Modified Minkowski metric, we instead recommend using the Scaled Euclidean. Our recommendations were based on the fact that the Scaled Euclidean’s performance was almost as good and its computational requirements are about 20\% and 40\% those of the Minkowski and Modified Minkowski, respectively (see Table I). The individual scores for each metric/planner/environment combination can be found in [2].

Generally, the metrics performed comparably in all environments. An exception was the bounding box, which performed worse in the alpha puzzle than in the 6-cube environments, likely because the bounding box is not a good approximation of the \(C\)-shape.

D. Selecting local planners for environments

After determining which distance metrics were best suited for each local planner in each type of environment, we then compared the planner and distance metric combinations. The straight-line and rotate-at-\(\frac{1}{2}\) planners behaved similarly, while the \(A^*\)-like planners were generally the most effective (and expensive).

The 6-cube environment results are shown in Fig. 2. In the graphs, there is one bar for each local planner/distance metric combination, and the black (white) portion of each bar represents the surface to surface (free to surface) connections made. We see that the best local planners in all 6-cube environments are the \(A^*\)-like planners, followed by the rotate-at-\(\frac{1}{2}\) planner. In general, rotate-at-\(\frac{1}{2}\) outperforms the straight-line planner, probably because its swept volume is often smaller. The dramatic difference between the rotate-at-0 and rotate-at-1 for the free to surface connections (white portion) is because the goal configuration is always near a C-obstacle surface (i.e., rotation at the goal, \(s = 1\), will likely cause collision).

The alpha puzzle environment results, shown in Fig. 3, are similar to the 6-cube, with the advantage of rotate-at-\(\frac{1}{2}\) over the straight-line being more pronounced.

E. Using multiple local planners

The results shown in Table VI indicate that different local planners do indeed make different connections. In
each row (environment), the percentage of the connections made only by that local planner is shown. The straight-line is not shown because all its connections were made by the $A^*$-like methods. Similarly, the rotate-at-$\frac{1}{2}$ values are low in easier environments because most of its connections were also made by straight-line or one of the $A^*$-like methods. However, as the environments gets higher, the performance of rotate-at-$\frac{1}{2}$ increases. Finally, the values for both $A^*$-like methods are lower since they made many of the same connections (they are both biased to neighbor(i), which increments the position and orientation coordinates towards the goal). When we included only one of the $A^*$-like methods in the analysis, in every environment at least 22% of the total connections were made only by the $A^*$-like planner; this percentage increased with the difficulty.

Table VII shows the comparison of two local planners. In the top half of the table, column one shows connections made by the straight-line but not the rotate-at-$\frac{1}{2}$ planner, and vice versa in column two. Column four shows the percentage of the total connections made by the combination of the two planners, which was generally more than 50%. The results of the rotate-at-$\frac{1}{2}$ and $A^*$-clearance combination (bottom half of the table) show that more than 80% of all connections are made by this pair. This implies that using both the rotate-at-$\frac{1}{2}$ and $A^*$-like planners would result in good connectivity. A good algorithm should try the faster local planner first, only if the first one fails it should try to use second local planner. In this schema, the failing time also plays an important role (see Table II) since we would prefer the first local planners should fail fast so that we could try the next local planners in reasonable time.

VI. CONCLUSION

The main goal of our study was to determine good combinations of distance metrics and local planners for use by PRMs for planning the motion of rigid objects in cluttered environments. Our results (Table V), include recommendations for selecting distance metrics for various local planners in different types of environments. Generally, a good choice is the Scaled Euclidean metric, where more weight is placed on the position coordinates as the environment becomes more cluttered. Although it was not always the absolute best metric, its performance was comparable and it is quite efficient to compute.

The most powerful local planners were the $A^*$-like planners. Among the less expensive planners, the rotate-at-$\frac{1}{2}$ planner was the best, outperforming the common straight-line local planner. However, we also found that each of the tested local planners made some connections that the others did not. Thus, roadmap connectivity will be enhanced if the PRM uses multiple local planners. Based on our experience, we would recommend the following order: first, the rotate-at-$\frac{1}{2}$ and straight-line planners, next, the rotate-at-0 and rotate-at-1 planners, and finally, the $A^*$ planners.

ACKNOWLEDGMENT

The alpha puzzle was designed by Boris Yamron of the Computer Graphics & Systems Group at GE CRD.

REFERENCES