“There are two ways of constructing a software design. One way is to make it so simple that there are obviously no deficiencies. And the other way is to make it so complicated that there are no obvious deficiencies.”

-- C.A.R. Hoare
Objectives

- Understand techniques for evaluating robot control code:
  - Theoretical / Analytical
  - Empirical
Techniques for Evaluating Robot Control Code

• Theoretical / Analytical:
  – Proofs of correctness, stability, liveness, etc.
  – Computational complexity comparisons

• Empirical (This is today's focus):
  – Selection of application domain/benchmark
  – Selection (or definition) of metrics of evaluation
  – Multiple experimental runs, to eliminate effect of specific starting conditions, environments, parameter settings, etc.
  – Average experiments over many trials
  – Maintain data on averages and standard deviations
  – Number of experiments needed: enough to ensure statistical significance
Empirical Evaluation and Validation of Robot Control Code

• Main point: since it is difficult to sufficiently model robot’s environment and interactions, it’s better to validate robot control code empirically

• However, in empirical evaluations, must ensure that performance evaluation is not skewed through limited test case selection

• Must generalize over:
  – Range of parameter settings
  – Range of robot environments (or, at least across a certain class)
  – (Perhaps) numbers of robots
  – Etc.
Typical Experimental Evaluation Approach

1. Define important variables/parameters of the system
2. Determine a range of values for variables/parameters to be tested
3. Define metrics to be used for evaluating system
4. Determine characteristics of environment which algorithm is designed to handle
5. Generate a variety of environments within this class to test
   - Best if environments are randomly selected
   - For example, generate obstacle size and location based upon a particular distribution
6. Determine how many experimental runs are needed to generate statistical significance
7. Determine a baseline performance to which the new algorithm will be compared
8. Run multiple experiments, varying starting position of robot randomly
9. Collect data over multiple experimental runs, and average data, also maintaining standard deviation
1. Define **important variables/parameters** of the system

- Want to define any variables/parameters that influence the outcome of the experiment

- Example variables/parameters:
  - Weights on force vectors
  - Robot speed/velocity profile
  - Robot sensing range
  - Robot communications range
  - Communications failure rate
  - Other application-specific parameters (e.g., in swarming, have to define preferred distance between robots)
  - Size of the environment
  - Etc.
2. Determine a **range of values** for variables/parameters to be tested

• For all the variables/parameters identified in step 1, you now must decide what the **range of values** should be.

• For example:
  – Number of robots: vary from 1 to 10 (or 100, or 1000), in increments of 1 (or 10 or 100)
  – Sensing range: vary from 1 to 10, in increments of 1 (or 2 or 3 …)
  – Communications range: vary from 1 to 10, in increments of 1 (or 2 or 3 …)
  – Communications failure rate: vary from 0 to 100, in increments of 10 (or 20, …)
3. Define **metrics** to be used for evaluating system

- Metrics are often application-specific
- For example, what metric do we want to use here?

---

QuickTime™ and a YUV420 codec decompressor are needed to see this picture.

---

- Formation-keeping: metric = average formation error:

\[
\sum_{t=0}^{t_{\text{max}}} \sum_{i \neq \text{leader}} d_i(t)
\]
3. Define **metrics** to be used for evaluating system (con’t.)

- Consider the DARPA Grand Challenge application/task:
  
  (DARPA_….avi movie)

- What characteristics do we want to measure?
3. Define **metrics** to be used for evaluating system (con’t.)

- What metric would you use here?

  - Box pushing: metric = perpendicular distance pushed per unit time

  $d_{\perp}(t) / t$
3. Define **metrics** to be used for evaluating system (con’t.)

- What about here?

QuickTime™ and a YUV420 codec decompressor are needed to see this picture.
3. Define **metrics** to be used for evaluating system (con’t.)

- Cooperative target tracking: metric = average number of targets observed (collectively) per unit time (with \( m \) robots and \( n \) targets)

\[
A = \frac{\sum_{t=1}^{T} \sum_{j=1}^{n} g(B(t), j)}{T} \\
\text{where:} \\
B(t) = [b_{ij}(t)]_{m \times n} \text{ such that } b_{ij}(t) = \begin{cases} 1 & \text{if robot } i \text{ is observing target } j \text{ at time } t \\ 0 & \text{otherwise} \end{cases} \\
g(B(t), j) = \begin{cases} 1 & \text{if } \exists \text{ an } i \text{ such that } b_{ij}(t) = 1 \\ 0 & \text{otherwise} \end{cases}
\]
Metrics: Application Dependent vs. Independent

- Easy to measure application-dependent characteristics
  - Numbers of objects moved
  - Distance traveled
  - Number of targets within view
  - Time of task completion

- Harder to derive application-independent measures
  - Issues of robustness, reliability, adaptivity, etc. are hidden
  - E.g., difficulties in adaptivity ==> measured as fewer objects moved
4. Determine **characteristics of environment** which algorithm is designed to handle

- Characteristics of the environment:
  - Wide open space?
  - Open space with clutter?
  - Structured?
4. Determine **characteristics of environment** which algorithm is designed to handle (con’t.)

- How can you quantify the complexity of the environment?

- One measure (Cazals and Sbert, 1997):
  - Generate uniformly distributed random rays
  - Compute statistics such as:
    - To measure occlusion:
      - Average number (and standard deviation) of times rays intersect with objects in the environment
    - To measure how large open spaces are:
      - Average length (and standard deviation) of uninterrupted lines
5. Generate a **variety of environments** within this class to test

- For example, can randomly generate locations of obstacles:
  - $y \leftarrow$ Generate random number in $(0,a)$
  - $x \leftarrow$ Generate random number in $(0,b)$
  - $L_1 \leftarrow$ Generate random number in $(0,\text{max\_obstacle\_size})$
  - $L_2 \leftarrow$ Generate random number in $(0,\text{max\_obstacle\_size})$
  - Add obstacle with corner at $(x,y)$ of size $(L_1 \times L_2)$

- To do this:
  - $y \leftarrow$ Generate random number in $(0,a)$
  - $x \leftarrow$ Generate random number in $(0,b)$
  - $L_1 \leftarrow$ Generate random number in $(0,\text{max\_obstacle\_size})$
  - $L_2 \leftarrow$ Generate random number in $(0,\text{max\_obstacle\_size})$
  - Add obstacle with corner at $(x,y)$ of size $(L_1 \times L_2)$

- Can vary approach for different obstacle shapes; can add rotation
6. Determine how many experimental runs are needed to generate statistical significance

- Example:

```
<table>
<thead>
<tr>
<th>Normalized Cumulative Formation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategy IV</td>
</tr>
<tr>
<td>Strategy III</td>
</tr>
<tr>
<td>Strategy II</td>
</tr>
<tr>
<td>Strategy I</td>
</tr>
</tbody>
</table>

Error 0  50  100  150  200  250  300

- Does strategy IV have a statistically different performance than strategy III?
- What about II and III? III and IV?
- How do you tell, in general?
6. Determine how many experimental runs are needed to generate statistical significance (con’t.)

- Obviously, one run is never enough, if there is noise in the system.
  - Might be really lucky, or really unlucky, on a single run
  - What we want to know about is likely performance

- Need to collect sufficient data so that you have confidence that the results are meaningful (i.e., significant)

- In general, the more data you collect, the more confident you can be in your results

- In general, the closer the performance is of two approaches, or the more variation there is in the performance, the more data you’ll need to determine if there is a statistical difference
6. Determine how many experimental runs are needed to generate statistical significance (con’t.)

- Example #1:

Here, we ran 3 experiments with each algorithm. Are we confident that the algorithms perform differently?

Yes! Average performance is very different, and standard deviation on each performance is small.

- Example #2:

Here, we ran 3 experiments with each algorithm. Are we confident that the algorithms perform differently?

No! Average performance is somewhat different, but standard deviation on each performance is large.
6. Determine how many experimental runs are needed to generate \textbf{statistical significance} (con’t.)

• What if we ran more experiments on Example #2?

![Graph showing metric vs algorithms]

Are we confident now that the algorithms perform differently?

Yes! The larger number of samples shows us that these results are coming from two different distributions.
6. Determine how many experimental runs are needed to generate statistical significance (con’t.)

- How to determine the number of runs needed for statistical significance?
- Use a statistical test, such as “Students t test”
- Student’s t distribution is a probability distribution that estimates the mean of a normally distributed population, with small sample size.

Here, the standard deviation is unknown, and has to be estimated from the data.

What we want to do is to determine if the two distributions (from the performance of two different algorithms) are different.
6. Determine how many experimental runs are needed to generate statistical significance (con’t.)

- Compare performance of 2 algorithms as follows:
- Define $\mu_1$ and $\mu_2$ as the means (i.e., averages) of the two distributions
- Define a “null hypothesis”, $H_0$
  - $H_0$: $\mu_1 = \mu_2$ and there is essentially no difference between the two policies.
- Let $\mu_1 =$ mean of group 1; $\mu_2 =$ mean of group 2; $S_1 =$ standard deviation of group 1; $S_2 =$ std. dev. of group 2
  $n_1 =$ # of samples of group 1; $n_2 =$ # of samples of group 2
- Compute the following:
  $$T = \frac{\mu_1 - \mu_2}{\sigma \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$
  where:
  $$\sigma = \sqrt{\frac{n_1 S_1^2 + n_2 S_2^2}{n_1 + n_2 - 2}}$$
6. Determine how many experimental runs are needed to generate statistical significance (con’t.)

• Then, on the basis of a two-tailed test at a 0.01 level of significance, we would reject $H_0$ if $T$ were outside the range $-t_{.995}$ to $t_{.995}$

• To calculate values of $-t_{.995}$ to $t_{.995}$, we need to know the number of degrees of freedom
  – Example: For $n_1 + n_2 - 2 = 250 + 250 - 2 = 498$ degrees of freedom

• Then, we look up these values from a statistical table. For 498 DOF, $-t_{.995}$ to $t_{.995}$ is in the range $-2.58$ to $2.58$.

• If our $T$ calculation is outside this range, then we declare the two algorithms different.

• Repeat this process for each pair of algorithms, if you have more than two.
6. Determine how many experimental runs are needed to generate statistical significance (con’t.)

- When plotting averaged data, need to show “error bars”, which show the standard deviation.
- The error bars provide information on the variation in performance
- Example:

![Chart showing error bars for different systems]
7. Determine a baseline performance to which the new algorithm will be compared

- When developing a new algorithm, how do we assess the quality of the performance?

- We may know actual values of metrics (e.g., time), but how do we know if those answers are good or bad?

- Typically: use a baseline algorithm against which to compare your new results.

- *Baseline algorithm:* an “obvious choice” algorithm, which is the most reasonable alternative approach that you can think of
  - For example:
    - Established alternative approaches in the field (e.g., by others who have solved similar problems in the past)
    - A pure random approach (in cases where we are the first to address this problem)
    - A pure greedy approach (i.e., where the robot does what looks best at the moment)
    - Others …
8. Run multiple experiments, and
9. Collect data over multiple experimental runs

- Here’s where you vary all the parameters that matter, the environment features, the robot starting position, etc.

- Collect data according to your metrics, analyzing whether it is significant, and continuing until you get statistically significant results.
Remember: Key Points of Empirical Robot Control Evaluation

1. Define *important variables/parameters* of the system
2. Determine a *range of values* for variables/parameters to be tested
3. Define *metrics* to be used for evaluating system
4. Determine *characteristics of environment* which algorithm is designed to handle
5. Generate a *variety of environments* within this class to test
6. Determine how many experimental runs are needed to generate *statistical significance*
7. Determine a *baseline performance* to which the new algorithm will be compared
8. Run multiple experiments, varying starting position of robot randomly
9. Collect data over multiple experimental runs, and *average data*, also maintaining standard deviation