Mobile robot navigation using evolving neural controller in unstructured environments

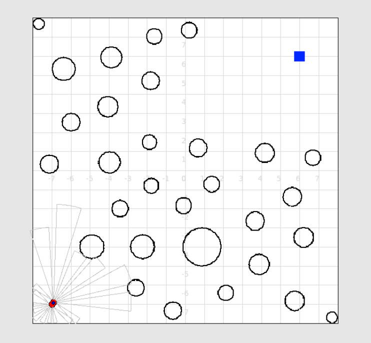
Awhan Patnaik, Khimya Khetarpal, Laxmidhar Behera
awhanp, khimya, lbehera @iitk.ac.in

Department of Electrical Engineering
Indian Institute of Technology Kanpur, India

March 14, 2014

Problem Statement

- **Autonomous** mobile robot navigation
- **Unknown unmapped** environment
- **Partially blind robot** $\Longrightarrow$ **sonar proximity sensors** facilitating limited range
- **Avoid obstacles** while navigating towards a **stationary target**



Target(blue square) AND Obstacles (black circles or any shapes)

Related Work

- Classical Methods:
  - Roadmap Technology
  - Cell Decomposition Method
  - Artificial Potential Field Method
- Search Algorithms
  - Voronoi Diagram
  - A* Search Algorithm
  - D*, Field D* Search Algorithm
- Heuristic Approaches
  - Fuzzy Logic Control
  - Artificial Neural Network
- Other Approaches
  - Vector Force Field
  - Vector Field Histogram

Behavior Based Navigation - I

- Navigation comprises of: **Basic Behaviors**
  - Obstacle Avoidance
  - Target Seeking
  - Wall Following
- A hierarchial decompostion: **Divide and Conquer**
  - Hard switching between controllers
  - Blending of behaviors

Behavior Based Navigation - II

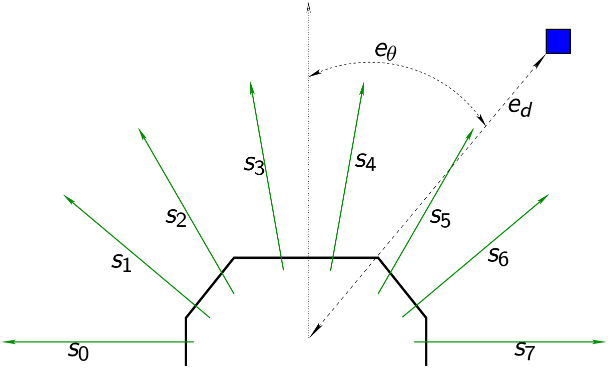
- Predefined Behavior: **How the switch is done?**
  - Supervisor chooses between pre-defined behaviors
  - Learning to switch

- Switching between controllers: **Shortcomings**
  - Pre-defined behaviors
  - Additional behavior supervisor/arbitration mechanism
  - Chattering $\Longrightarrow$ Performance Degradation

This Work

- The real challenge - **implicit learning**
- Evolutionary Learning - **Genetic Algorithm**
  - **Single controller** to accomplish both target seeking and obstacle behavior
  - **Real coded** genetic algorithm that uses **polynomial mutation** and **simulated binary crossover**
  - The genotype is a **multi-layered neural network** with a **tan sig(.)** activation function.

The Robot - Sensors - Error - Target



$s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7$ are 8 **sonar** proximity sensors.
each with 5 meters range and 15 degrees view.

e_θ is angle error between robot heading and target.
e_d is distance error between robot and target.

Objective function

$$\underset{\mathbf{w}}{\text{minimize}} \, [\text{target farness} + \text{obstacle nearness}]$$

$$\text{such that}:$$

$$\text{no collisions occur}$$

$$\text{robot reaches target}$$

robot is controlled by a 3 layer MLP parameterized by weights $\mathbf{w}$

a possible formulation

$$\underset{\mathbf{w}}{\text{minimize}} \frac{1}{T} \sum_{t=0}^{T} \left(e_\theta(t) + e_d(t) + \exp\left(-s_l\right) + \exp\left(-s_f\right) + \exp\left(-s_r\right)\right)$$

such that :

$$\alpha\beta \left(e_\theta(t_f) + e_d(t_f)\right) = 0$$

$s_l = \min\{s_0, s_1, s_2\}$ left worst sensor reading
$s_f = \min\{s_3, s_4\}$ front worst sensor reading.
$s_r = \min\{s_5, s_6, s_7\}$ right worst sensor reading.

T is the maximum time steps allowed for robot simulation.
t_f is the time at which simulation ends i.e. final time.

minimizing target farness

$$\sum_{t=0}^{T} \left(e_\theta(t) + e_d(t)\right) \quad \Longrightarrow \text{ more penalty for facing away}$$
$$\Longrightarrow \text{ more penalty for staying away}$$

minimizing obstacle nearness

$$\sum_{t=0}^{T} \exp\left(-s_{\mathrm{l}}\right) \implies \text{more penalty for obstacle proximity on left}$$
$$\sum_{t=0}^{T} \exp\left(-s_{\mathrm{f}}\right) \implies \text{more penalty for obstacle proximity up front}$$
$$\sum_{t=0}^{T} \exp\left(-s_{\mathrm{r}}\right) \implies \text{more penalty for obstacle proximity on right}$$

more the sensor reading s the smaller the $\exp(-s)$ value

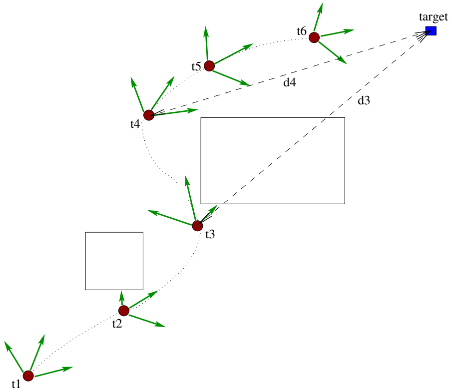
minimizing $\sum \exp(-s)$ promotes obstacle avoidance.

constraint

$\alpha\beta\left(e_\theta(t_f) + e_d(t_f)\right)$ assigns a penalty based on **final position** of robot wrt target.

scaling factor α is 1 for robots that do not **hit obstacles** but 3 others. Thus robots that hit get 3 times the penalty.

β is 0 for robots that **reach target**, 1 otherwise. Thus robots that reach target bear no constraint penalty.



Robot position at 6 time instants.

Green arrows represent left, front and right sensor readings.

d3 and d4 distance to target at time instants t3 and t4

`(other distances not shown for clarity)`

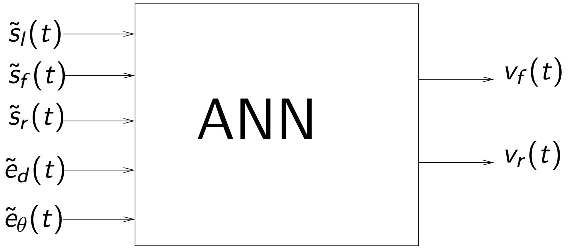
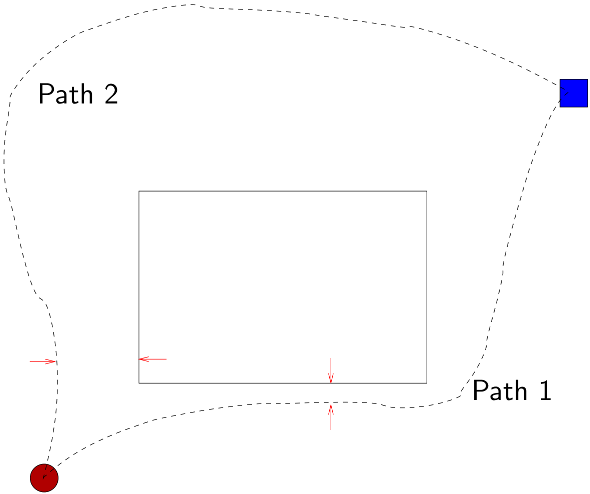


Figure: 3 layer feed forward ANN controller. 5 inputs and 2 outputs. $\tilde{s}_l(t)$, $\tilde{s}_f(t)$ and $\tilde{s}_r(t)$ represent the average obstacle presence to the left, front and right of the robot. $\tilde{e}_\theta(t)$ and $\tilde{e}_d(t)$ are the normalized distance and angle error. Tilde on the input variables denote that these are the normalized values. 1 hidden layer with 10 neurons. Output neurons are linear and hidden neurons use a scaled tanh() non-linear activation function.

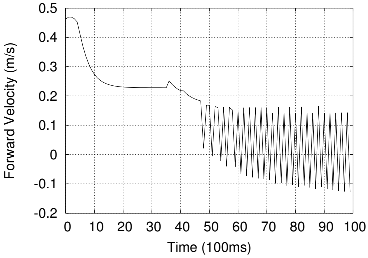
Origin of conflict in objectives

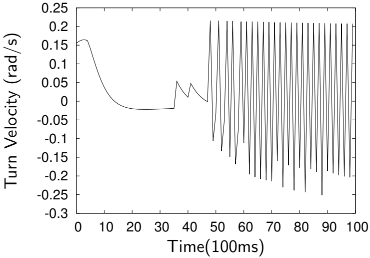


Path 2 has greater safety margin but travels a longer path.

Simulation speed up techniques.

- If robot **hits** an obstacle it is killed off.
- If robot motion is **oscillatory** it is killed off.





- If robot motion is very **slow** it is killed off.

Training

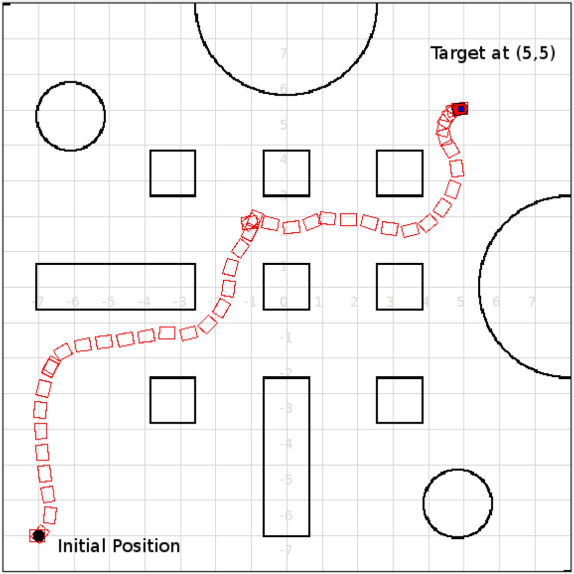
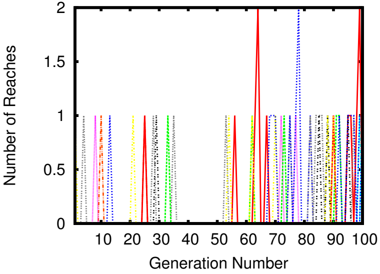


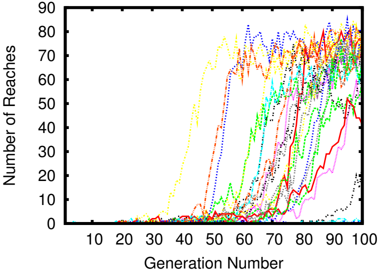
Figure: Training Environment

Necessity of Constraint

- Number of Generations: 100
- Population Size: 100
- Runs: 30
- Objective: $\sum_{t=0}^{T} e_d(t)$

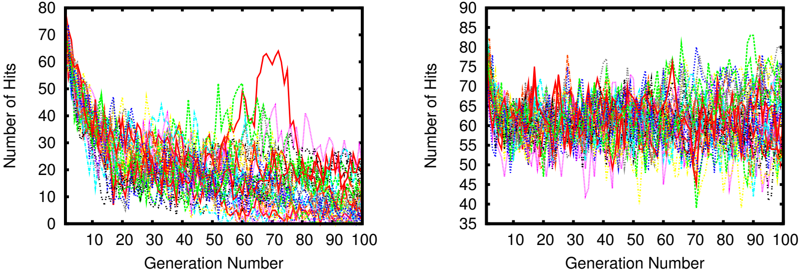


(a) Training with no constraints



(b) Training with constraints

Figure: Plot of number of reaches vs. generations



(a) Training with constraints

(b) Training with no constraints

Figure: Plot of number of hits vs. generations

Variation in objectives

Case 1: $\sum_{t=0}^{T} e_d(t)$

Case 2: $\sum_{t=0}^{T} e_\theta(t) + e_d(t)$

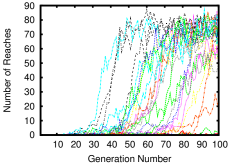
Case 3: $\sum_{t=0}^{T} e_\theta(t) + e_d(t) + \exp\left(-s_f\right)$

Case 4: $\sum_{t=0}^{T} e_\theta(t) + e_d(t) + \exp\left(-s_f\right) + \exp\left(-s_\mathrm{l}\right) + \exp\left(-s_\mathrm{r}\right)$

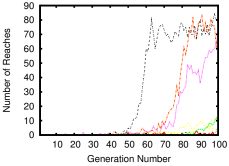
Metrics for Evaluation

- Number of Reaches
- Number of Hits
- Proximity
- Front Clearance

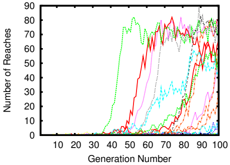
Results & Discussion



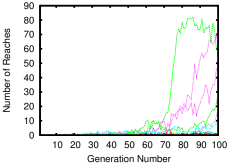
(a) $\sum_{t=0}^{T} e_d(t)$



(b) $\sum_{t=0}^{T} (e_\theta(t) + e_d(t))$

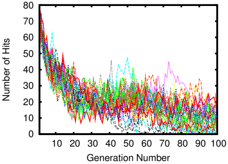


(c) $\ldots + \exp(-s_f)$

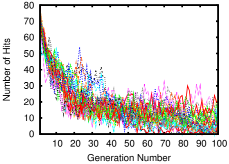


(d) $\ldots + \exp(-s_l) + \ldots$

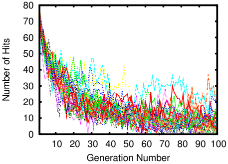
Figure: Plot of number of reaches vs. generations



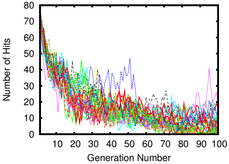
(a) $\sum_{t=0}^{T} e_d(t)$



(b) $\sum_{t=0}^{T} (e_\theta(t) + e_d(t))$

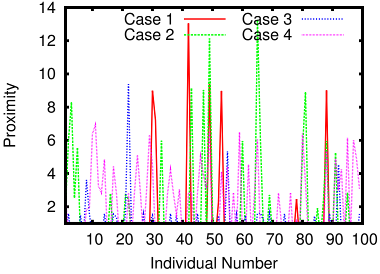


(c) $\ldots + \exp(-s_f)$

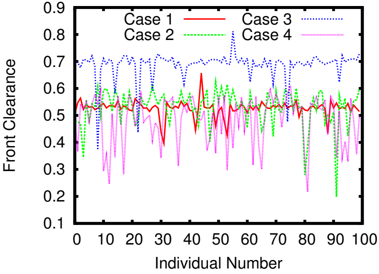


(d) $\ldots + \exp(-s_l) + \ldots$

Figure: Plot of number of hits vs. generations



(a) Proximity metric for final population of best fit run for all cases



(b) Front clearance metric for final population of best fit run for all cases

Validation

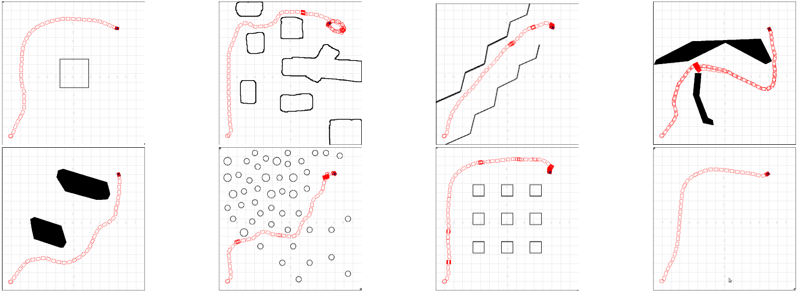


Figure: Validation results across different environments

An overview of the criterion is described here.

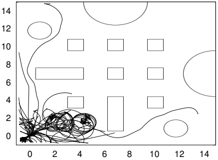
- **Success:** Reach the exact coordinates, or
  Rotate around the target ,if not stop, or
  Oscillate around the target
- **Close:** Navigates without a hit, and
  Moves towards the target,
  but does not reaches the target accurately
- **Hit:** Hits an obstacle and stop
- **Failure:** Oscillates far away from the target, or
  Passes very closely from the target and does not navigate back

Environment	Case 1	Case 2	Case 3	Case 4
Training Environment	Success	Success	Success	Success
empty	Success	Success	Success	Close
one	Success	Success	Success	Failure
multi	Close	Success	Success	Failure
pathway	Close	Hit	Close	Success
multi2	Hit	Close	Hit	Failure
circ2	Hit	Hit	Hit	Close
variable	Hit	Hit	Hit	Success
cave	Hit	Success	Hit	Success
parallel objects	Success	Hit	Close	Failure
random	Hit	Hit	Close	Failure
streched objects	Success	Hit	Close	Failure

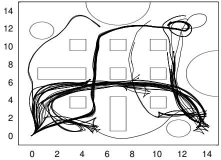
Case 3 is the best, though not by a far margin.

Case 1 next best.

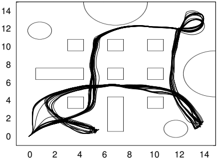
Case 4 which has the most constraints registers the most failures but it should be noted that it never hits an obstacle. The target reaching behavior for Case 4 is most poor though.



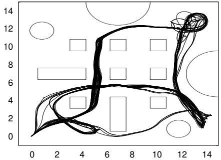
(a) Gen 1



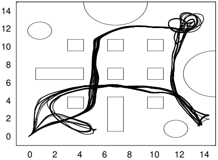
(b) Gen 20



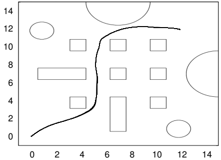
(c) Gen 40



(d) Gen 60



(e) Gen 80



(f) Gen 100

Figure: Evolution of robot trajectories as the generations progress.

Conclusion and Future Works

- single controller multiple *conflicting* tasks
- evolution of *implicit* switching/co-ordination behaviour
- complexity of objective function vs performance $\Longrightarrow$ simple is better
- blending of behaviours
- multi-objective optimization