Edith Cowan University

CSG2341
Intelligent Systems

Assignment 1B

Saucers Part 2

Martin Ponce
Student 10371381

Tutor: Philip Hingston

October 1, 2015

Contents

1 Introduction

This report examines fuzzy logic using Sugeno style inference, and its practical use in a video game called Saucers. Saucers is a multiplayer game where each player indirectly controls their flying saucer using a fuzzy logic controller. The saucers meet on a battle space, a rectangular xy plane, with the purpose of destroying each other. The walls of this space cannot be travelled through, and will cause the saucer to ricochet off the wall when hit.

Up to twelve saucers spawn at the start of the game, and each saucer begins with equal amounts of energy. This energy is consumed as they fly around, fire their auto aiming cannon mounted on a rotating turret, or use their shield to protect themselves from energy blasts. When a saucer is hit by an energy blast, energy is depleted. The amount of energy depletion depends on the amount of energy committed to firing, and how far away it was fired. Energy blasts lose energy the further they travel.

Saucers cannot stop flying and will always consume energy. However, the speed of a saucer can be controlled. The slowest speed consumes the least amount of energy, while the fastest speed consumes the most. The saucer's heading can also be controlled, and can turn left or right in any direction. There is however, a small energy penalty for turning.

Each saucer is equipped with multiple sensors to provide inputs for fuzzy logic. There is a sensor for all current enemy saucers, providing information about their distance, their direction in relation to the player, their current heading in relation to the player's heading, their current speed, their current energy level, and their ID number. Similarly, there is a sensor for power ups that randomly appear in the battle space, providing an energy boost to the first saucer that touches it, and a sensor for all energy blasts. These sensors provide the same information as the enemy saucer sensor. There is also a sensor providing information about the player's own energy level.

When a saucer loses all of its energy, it disappears from the battle space and loses. Each game round has a limited time, and if multiple saucers are still alive at the end of the round, it results in a tie for that spot in the ladder. The goal of this report is to design a fuzzy logic controller so that it's saucer will be the last saucer remaining during the end of a game round.

2 Idea

The main tactic for this controller is fly defensively in order to conserve energy and survive until there are a two enemies left. Turns for the most part will focus on the energy blast sensor, in order to dodge as many of them as possible. When there are only two enemy saucers left in the arena, turning will focus on the enemy saucer sensor, to track them down and shoot at them from a close distance.

However, when any power ups spawn nearby, the goal is to move straight to the power up, ignoring any energy blasts. If any close energy blasts close to the player while attempting to retrieve a power up, the player will raise the shield. The rate of fire will be kept to a minimum for most situations to conserve energy, unless a nearby power up spawns, or if there is only two remaining enemy saucers left. In these cases, the rate of fire will increase, to either deter enemy saucers from retrieving the power

up, or attempt to destroy them.

Speed will also be kept at the minimum for most situations, only increasing speed when necessary for dodging, or when a power up spawns nearby, to try and get to it first. Speed will also be increased when there are only two remaining enemy saucers left, attempting to destroy them before the timer runs out.

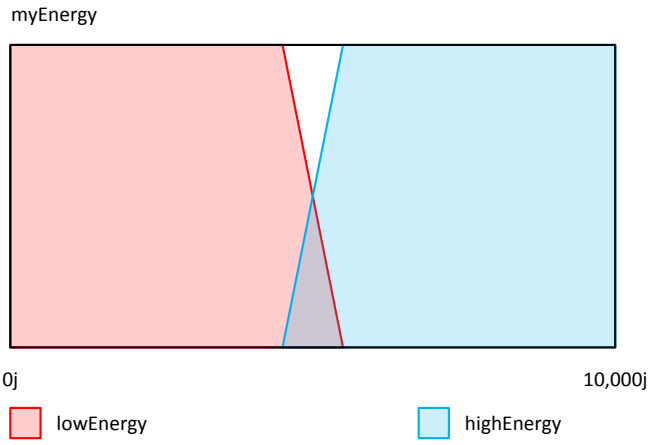
This controller attempts to implement these strategies with turning, speed, shields and firepower, with the primary goal of flying defensively by dodging energy blasts and retrieving nearby power ups. When there are only two enemies left, the saucer will begin to fly offensively, increasing speed and firepower, and attempt to destroy the enemies before the timer runs out.

3 Input linguistic variables

3.1 myEnergy

The linguistic variable *myEnergy* is the player's energy level and determines whether or not the player has *lowEnergy* or *highEnergy* remaining. The universe of disclosure for *myEnergy* is between 0 joules and 10,000 joules, the amount of energy that all saucers begin with.

Figure 1: *myEnergy* fuzzy sets



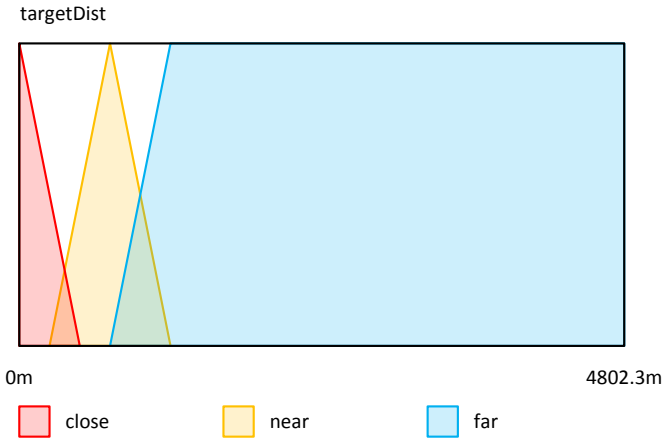
3.2 Target variables

The sensor returns a list of all the enemy saucers currently in the battle space. This controller only considers the closest saucer as the target, and ignores all other saucers in the list.

3.2.1 targetDist

The linguistic variable *targetDist* is the distance from the player to the target. The universe of disclosure for *targetDist* is between 0m and 4802.3m, and the fuzzy sets are defined as *close*, *near*, and *far*.

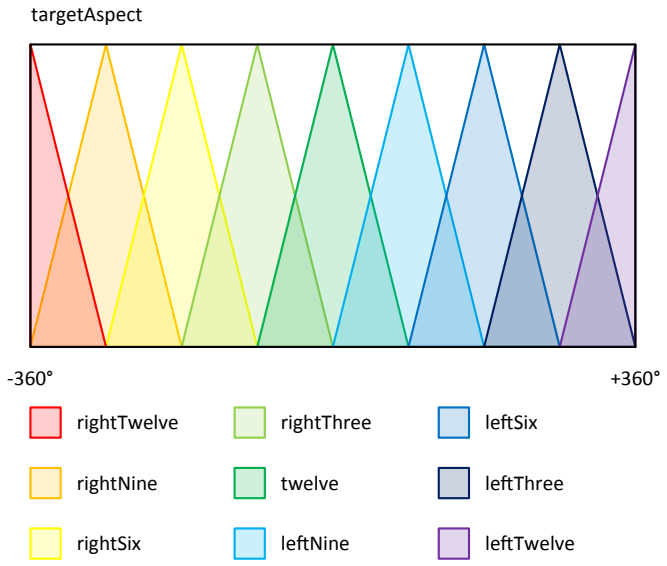
Figure 2: *targetDist* fuzzy sets



3.2.2 targetAspect

The linguistic variable *targetAspect* is the direction of the target in relation to the player. The universe of disclosure for *targetAspect* is between -360° and +360°. Positive values rotate to the left, and negative values rotate to the right. The fuzzy sets selected relate to clock positions, similar to what fighter pilots might call out in combat. There are three twelve o'clock positions due to the two revolutions between -360° and +360°.

Figure 3: *targetAspect* fuzzy sets

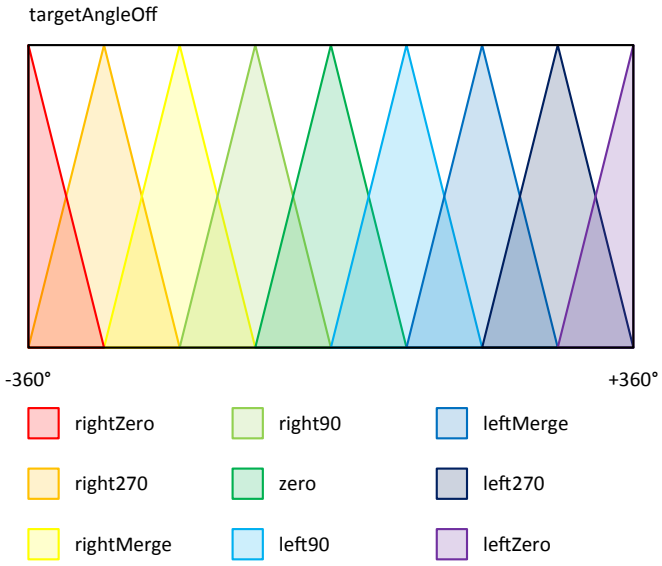


3.2.3 targetAngleOff

The linguistic variable *targetAngleOff* relates to the target's current heading, in relation to the player's current heading. For example, if the target is heading towards the player perpendicularly from the right, the target's angle-off would be +90°. Similarly, if the target has the exact same heading as the player, the target's angle-off would be 0°. Again, positive values rotate to the left, and negative values rotate to the right. The universe of disclosure is between -360° and +360°. The fuzzy

sets selected mimic clock positions, similar to *targetAspect*, however are named with degree values. A merge is when the player and target have opposite angle-off's, ie. 180°. In this situation, if the player and target were in front of each other, they would facing each other, and would be about to directly pass each other, in a "merge".

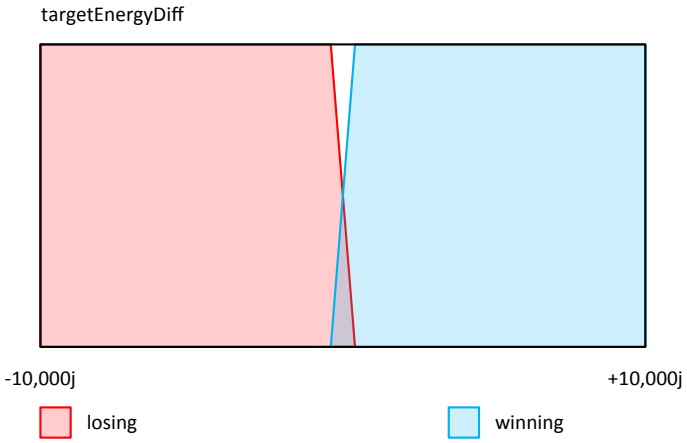
Figure 4: *targetAngleOff* fuzzy sets



3.2.4 targetEnergyDiff

The linguistic variable *targetEnergyDiff* relates to the difference between the player's energy and the current target's energy. The universe of disclosure for *targetEnergyDiff* is between -10,000j and +10,000j. The fuzzy sets selected for this linguistic variable are *losing* and *winning*.

Figure 5: *targetEnergyDiff* fuzzy sets



3.3 Blast variables

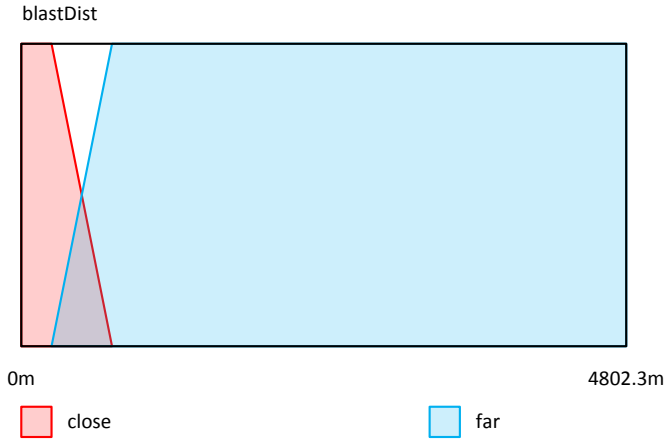
The sensor returns a list of all energy blasts currently in the battle space. This controller only considers the closest energy blast to dodge, and ignores all other

blasts in the list.

3.3.1 blastDist

The linguistic variable *blastDist* is the distance from the player to the energy blast. The universe of disclosure for *blastDist* is between 0m and 4802.3m, and the fuzzy sets are defined as *close* and *far*.

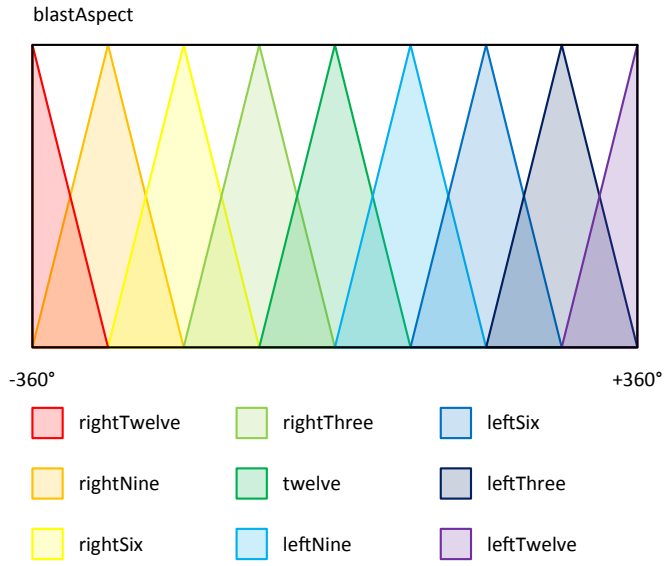
Figure 6: *blastDist* fuzzy sets



3.3.2 blastAspect

The linguistic variable *blastAspect* is similar to *targetAspect*, but relates to the direction from the player to the energy blast. Similar fuzzy sets, based on the clock analogy have been used.

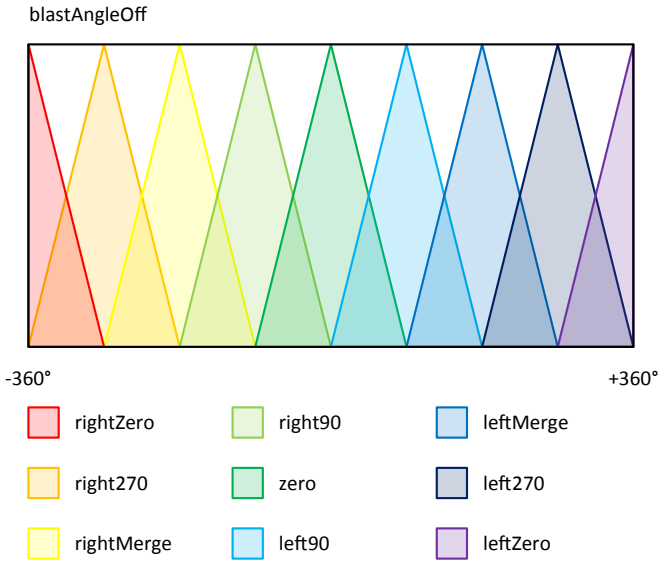
Figure 7: *blastAspect* fuzzy sets



3.3.3 blastAngleOff

The linguistic variable *blastAngleOff* is similar to *targetAngleOff*, but references the current heading of the energy blast in relation to the player's heading. Similar fuzzy sets have also been used.

Figure 8: *blastAngleOff* fuzzy sets



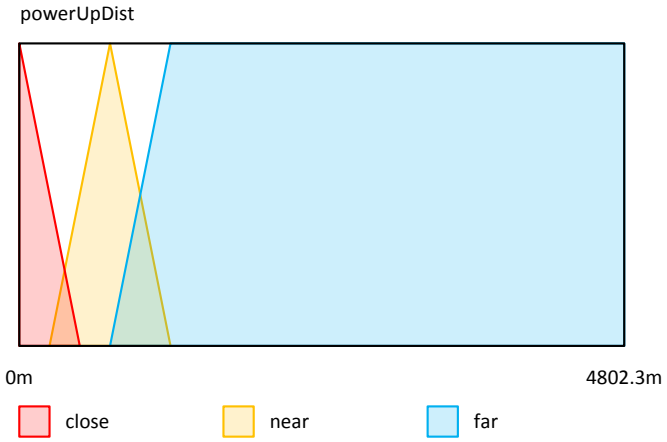
3.4 Powerup variables

The sensor returns a list of all powerups that currently exist in the battle space. To conserve energy, this controller only reacts to powerups only if they are nearby, and assumes that powerups that are far would most likely be consumed by an enemy by the time the player arrives.

3.4.1 powerUpDist

The linguistic variable *powerUpDist* is the distance between the target and the powerup. Similar to *targetDist*, the universe of disclosure is between 0m and 4802.3m. Similar fuzzy sets have also been used.

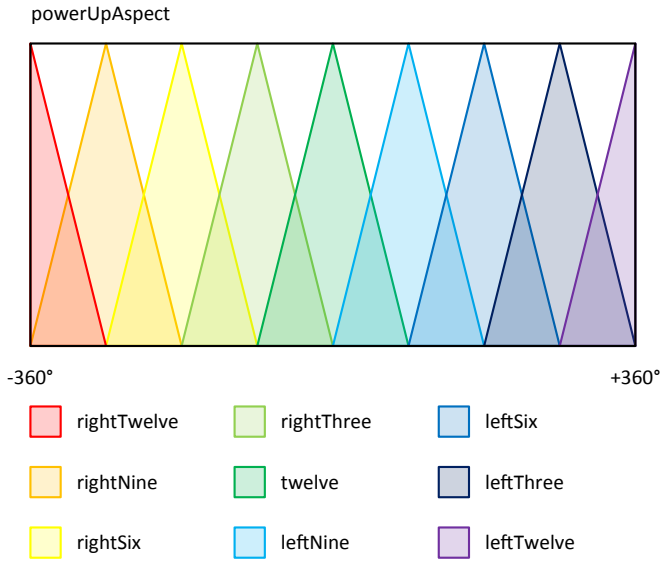
Figure 9: *powerUpDist* fuzzy sets



3.4.2 powerUpAspect

The linguistic variable *powerUpAspect* is the direction of the powerup in relation to the player. Again, the clock analogy has been used to determine the fuzzy sets. However, the angle-off of the powerup is not considered, since it is stationary and does not change its heading.

Figure 10: *powerUpAspect* fuzzy sets



4 Output linguistic variables

Due to the fact that the fuzzy logic API does not provide a method to prioritize rules, defensive and offensive versions of some output linguistic variables have been defined. Boolean variables are defined, which are then tested to determine which version of the rule is fired, during a given situation.

For example, the defensive turn rule only considers the blast linguistic variables as input, whereas the offensive turn rule considers both target and blast linguistic variables.

4.1 Defensive rules

4.1.1 defensiveTurn

The *defensiveTurn* linguistic variable defines which heading the saucer will take, in degrees, according to the rules that govern defensive turning. The linguistic variables used as input for *defensiveTurn* are: `layerVar = blastDist`, `rowVar = blastAngleOff`, and `colVar = blastAspect`, creating a 3D rule matrix. Note that r and l represent right and left in the table.

For example: IF (close) AND (rightSix) AND (zero) THEN (-90)
In other words, IF the energy blast is close, AND it is behind the player, AND it's heading towards the player, THEN turn right for 90°.

Table 1: *defensiveTurn close*

	rTwelve	rNine	rSix	rThree	twelve	lNine	lSix	lThree	lTwelve
rZero	0	0	-90	0	0	0	+90	0	0
r270	+90	0	0	+180	+90	0	0	+180	+90
rMerge	0	0	-90	0	0	0	+90	0	0
r90	+90	-180	0	0	-90	-180	0	0	+90
zero	0	0	-90	0	0	0	+90	0	0
l90	-90	0	0	+180	+90	0	0	+180	-90
lMerge	0	0	-90	0	0	0	+90	0	0
l270	-90	-180	0	0	-90	-180	0	0	-90
lZero	0	0	-90	0	0	0	+90	0	0

Table 2: *defensiveTurn far*

	rTwelve	rNine	rSix	rThree	twelve	lNine	lSix	lThree	lTwelve
rZero	0	0	0	0	0	0	0	0	0
r270	0	0	0	0	0	0	0	0	0
rMerge	0	0	0	0	0	0	0	0	0
r90	0	0	0	0	0	0	0	0	0
zero	0	0	0	0	0	0	0	0	0
l90	0	0	0	0	0	0	0	0	0
lMerge	0	0	0	0	0	0	0	0	0
l270	0	0	0	0	0	0	0	0	0
lZero	0	0	0	0	0	0	0	0	0

The far rules have zero values so that the player does not react and maintains current heading when the energy blast is far, and is not a threat.

4.1.2 defensiveSpeed

The *defensiveSpeed* linguistic variable defines how fast the saucer will travel, according to the rules that govern defensive speed. The linguistic variables used for input for *defensiveSpeed* are: `layerVar = blastDist`, `rowVar = blastAngleOff`,

and `colVar = blastAspect`, creating a 3D rule matrix.

For example: IF (close) AND (rightSix) AND (zero) THEN (125)
In other words, IF the energy blast is close, AND it is behind the player, AND it's heading towards the player, THEN speed is 125.

Table 3: *defensiveSpeed close*

	rTwelve	rNine	rSix	rThree	twelve	lNine	lSix	lThree	lTwelve
rZero	50	125	125	125	50	125	125	125	50
r270	50	125	50	125	50	125	50	125	50
rMerge	50	125	50	125	125	125	50	125	50
r90	50	125	50	125	125	125	50	125	50
zero	50	125	125	125	50	125	125	125	50
l90	50	125	50	125	125	125	50	125	50
lMerge	50	125	50	125	125	125	50	125	50
l270	50	125	50	125	50	125	50	125	50
lZero	50	125	125	125	50	125	125	125	50

Table 4: *defensiveSpeed far*

	rTwelve	rNine	rSix	rThree	twelve	lNine	lSix	lThree	lTwelve
rZero	50	50	75	50	50	50	75	50	50
r270	50	50	50	75	50	50	50	75	50
rMerge	50	50	50	50	75	50	50	50	50
r90	50	75	50	50	75	75	50	50	50
zero	50	50	75	50	50	50	75	50	50
l90	50	50	50	75	75	50	50	75	50
lMerge	50	50	50	50	75	50	50	50	50
l270	50	75	50	50	50	75	50	50	50
lZero	50	50	75	50	50	50	75	50	50

4.2 Offensive rules

4.3 Neutral

4.3.1 getPowerTurn

5 Learnings

6 Conclusion