MC906 - Trabalho 3

Victor Andrietta (RA 187835) e Guilherme Dumas Peres (RA 173711)

Introdução

Problemas nem sempre podem ser modelados com conjuntos crispy, alguns problemas representam situações nas quais uma solução pode estar em vários conjuntos ao mesmo tempo com um grau de pertinência diferente, a partir desse pensamento testamos a aplicabilidade da logica fuzzy na solução de um problema escolhido.

O problema consiste em desenvolver um controlador fuzzy para jogar flappy bird simplificado; Portanto, o resultado da defuzzyficação consiste na escolha de pressionar ou não a tela em cada momento, de forma a maximizar a longevidade do pássaro.

Foi utilizada como base a versão aberta do flappy bird presente no repositorio **freegames**. Esta versão foi modificada para implementar a tomada de decisão de movimentação do pássaro usando lógica fuzzy.

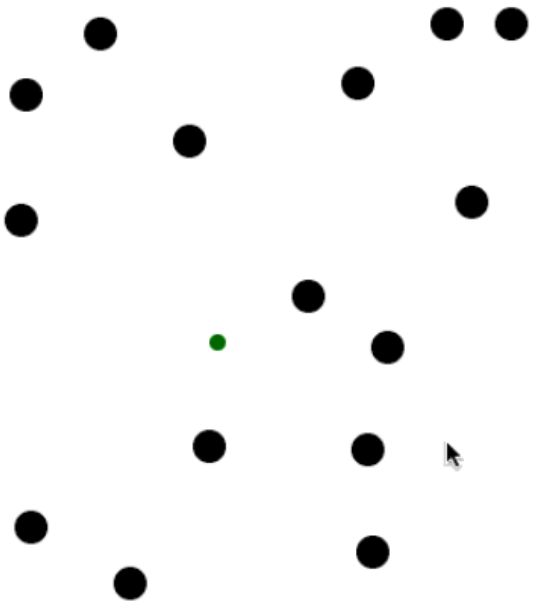

Figura 1: Simplified Flappy Bird

Trabalho proposto

Para desenvolver o projeto optou-se por usar Python como linguagem de programação apoiando-se na biblioteca SciKit-Fuzzy que contém uma infinidade de algoritmos para modelar fuzzy logic e em bibliotecas auxiliares para modelar o ambiente e as interações onde ocorrem as computações.

Tais algoritmos foram usados concretizar o modelo desenvolvido pelo grupo, que consiste de uma bolinha com diâmetro *10* deslocando-se *-5* unidades no eixo y a cada *50* milesegundos (turno) e vários obstáculos de diâmetro *20* gerados em posições aleatórias deslocando-se *-3* unidades no eixo x no mesmo turno em que o pássaro se move

tudo ocorrendo num grid de tamanho *400* por *400* com um alcance de *-200* a *200* tanto no eixo x quanto no eixo y e com um threshold de distância do pássaro entre os obstáculos de 5 unidades. Para fazer a bolinha se deslocar na direção positiva do eixo y uma função tap é acionada, após diversas computações, movendo-se *30* unidades no eixo y a cada tap.

Restrições e especificidades

Uma das especificidades do programa advém diretamente da biblioteca SciKit-Fuzzy que possui uma documentação ruim, confusa e nebulosa tornando a modelagem para o problema complicada. A especificidade em questão é que o pássaro não costuma ir para a parte negativa do eixo y e por isso têm a tendência de manter-se no meio do grid, mas isso será explicado detalhadamente mais tarde.

Materiais e métodos

Antecedentes(fuzzyfication)

Os antecedentes são usados para calcular os valores fuzzy que serão necessários para computações a cada turno. A funcao escolhida para modelar a pertinencia dos dados que alimentam os antecedentes foi a triangular (*trimf*).

- **wall**: proximidade do pássaro(bolinha menor) com a parede, possui uma range de *-200* a *200* no eixo y e é modelado como uma função triangular. O objetivo aqui é que o pássaro fique longe das paredes.
- **proximity**: calcula a soma das distâncias da bolinha menor para os obstáculos, considerando que foi dado um tap, ou seja, que o pássaro deslocou-se *+30* no eixo y e essa mesma soma considerando que não foi dado um tap, ou seja, que o pássaro deslocou-se *-5* unidades no eixo y. A soma que for maior faz com que proximity receba um valor entre *0* e ela. Esse antecedente é modelado como uma função triangular e o objetivo aqui é que o pássaro fique o mais longe possível de onde existir mais obstáculos.
- **no_tap_ball_threat**: possui uma range de *0* até proximity_range(que é uma distância segura do pássario até o obstáculo maior dentro de um threshold predefinido) é modelado como uma função triangular e o objetivo aqui é obter uma variável para avaliar o quão ruim é se o tap não **for** pressionado.
- **tap_threat**: é semelhante ao antecedente acima a diferença aqui é que o objetivo é obter uma variável para avaliar o quão ruim é se o tap **for** pressionado.

Notemos que as funções de pertinência acima foram geradas através do método *automf*, da biblioteca sugerida, que faz essa tarefa automaticamente.

Gráficos das fuzzificações

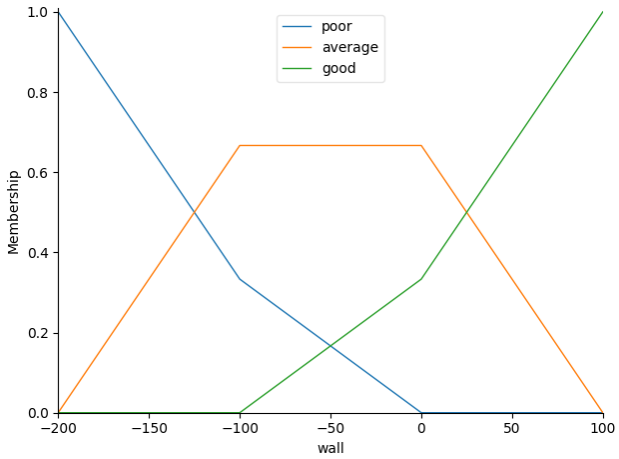

Figura 2: wall

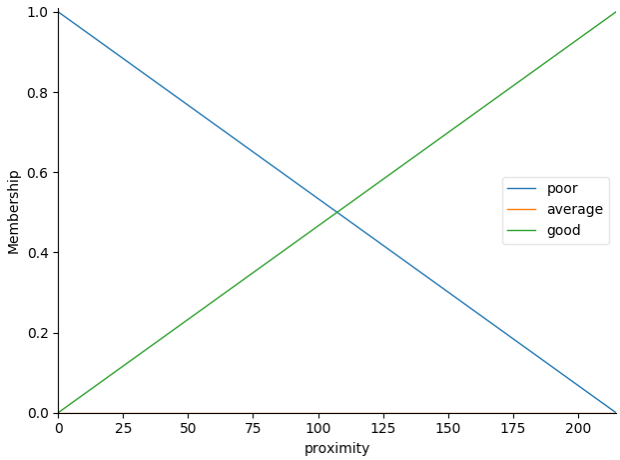

Figura 3: proximity

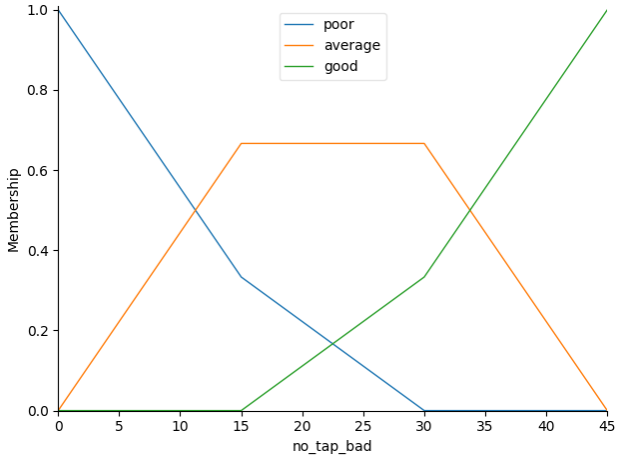

Figura 4: no_tap_ball_threat

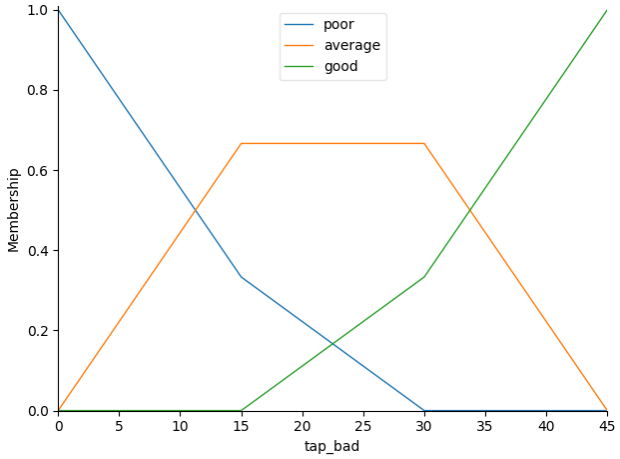

Figura 5: tap_threat

Consequentes(defuzzification)

O nível de discretização utilizado para a classificação dos parâmetros para a tomada de decisão na defuzyficação foi dividido em 3 intervalos (*"poor"*, *"average"*, *"good"*).

- **poor**: a decisão gerada pelo parâmetro que alimenta o antecedente implica em um posicionamento no próximo turno que expõe o pássaro a perigo de colisão.
- **good**: a decisão gerada pelo parâmetro que alimenta o antecedente implica em um posicionamento no próximo turno que não expõe o pássaro ao perigo de colisão(dentro de um threshold).
- **average**: a decisão gerada pelo parâmetro que alimenta o antecedente implica em um posicionamento no próximo turno tem uma chance de expor o pássaro ao perigo de colisão(dentro de um threshold).

Gráfico do output fuzzy

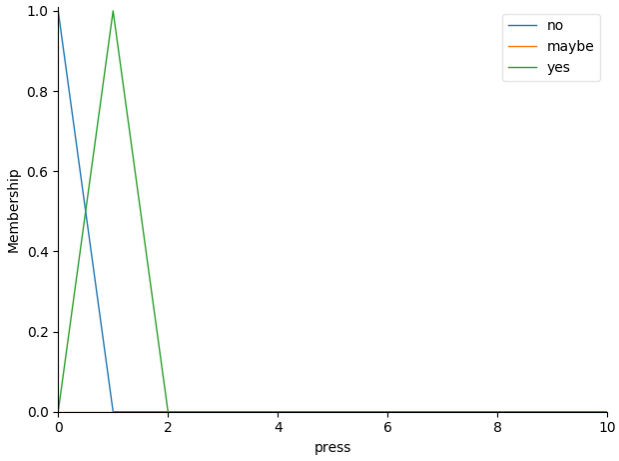

Figura 6: press

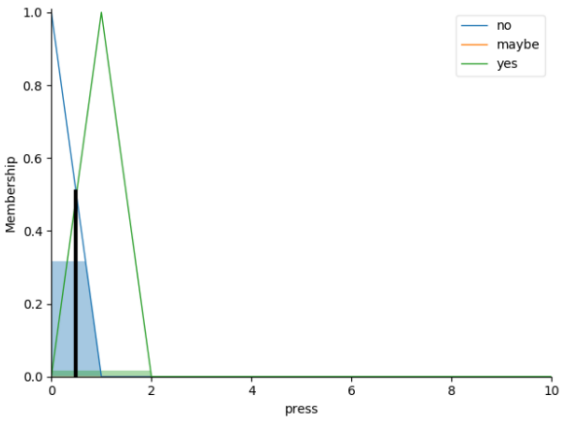

Figura 7: press_sample_centroid

Regras modeladas

- **Proximidade das paredes(Figura 2)**: essa regra foi dividida em 2 variáveis *wall_rule_1* e *wall_rule_2*. A primeira estipula que se a proximidade com a parede debaixo(chão) for *"poor"* o *press*(Figura 6) deve ser *"yes"*, ou seja, se o pássaro estiver muito próximo dela a função *tap*(*press*) deve ser acionada(*press["yes"]*), já a segunda avalia o quão próximo o pássaro está da parede de cima(teto), se ele estiver muito próximo(*"good"*) o tap(*press*) não deve ser acionado(*press["no"]).*

- **Proximidade dos obstáculos**: essa regra é dada pela variável *proximity_rule* e basicamente estipula que se a proximidade com os obstáculos(Figura 3) é menor com o *tap*, então ele deve ser acionado, caso contrário, ele não deve ser acionado. Essa regra tem precedência menor em relação à regra de iminência de colisão (descrita abaixo) e a regra de proximidade de paredes, ela será apenas levada em conta se as duas regras citadas retornarem "good"ou "average"em sua classificação.

- **Tomada de decisão de acordo com iminência de colisão**: essa regra, assim como a primeira que explicamos, foi dividida em 2 variáveis *no_tap_bad_rule* e *tap_bad_rule*. A primeira estipula que se a proximidade com o obstáculo que é calculada através da seguinte função: *threatening_balls_antecedents*, for *poor*(Figura 4) sem o tap(*press*), então *press*(Figura 6) deve ser acionado(*press["yes"]*), em outras palavras, se o pássaro estiver muito próximo do obstáculo ele deve pular com o intuito de ficar longe dele. A segunda é o contrário, se a proximidade com o obstáculo for *"poor"* dando o tap(*press*), então *press* não deve ser acionado(*press["no"]*), com o intuito de ficar longe dele.

Modelo de inferência

O modelo de inferência adotado foi o Mamdani no qual a saída é um conjunto fuzzy que pode ser defuzzificado, os métodos mais usados para defuzzificação são: centroid of area, bisector of a area, mean of maximum, smallest of maximum e largest of maximum, não vamos entrar em detalhes sobre todos esses métodos, apenas sobre o método do centroid que foi o escolhido pelo grupo. Em tal método agregam-se as funções antecedentes e o resultado é o centro da área sobre a curva.

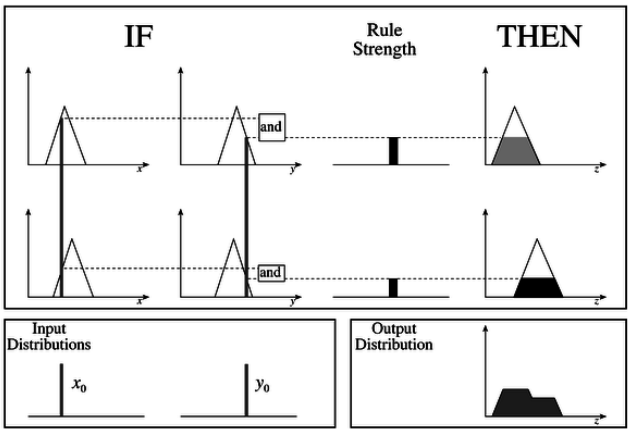

Figura 8: Mamdani inference model

Testes

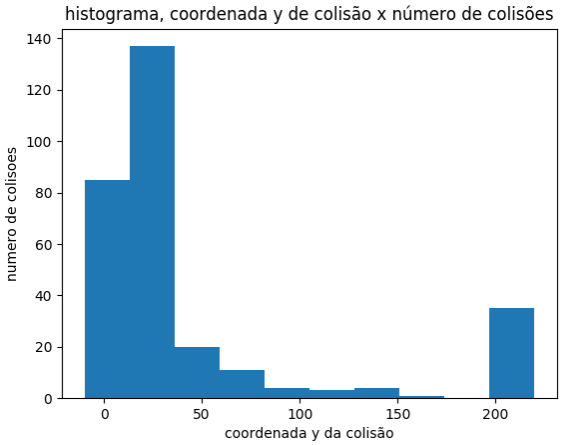

Figura 9: Histograma

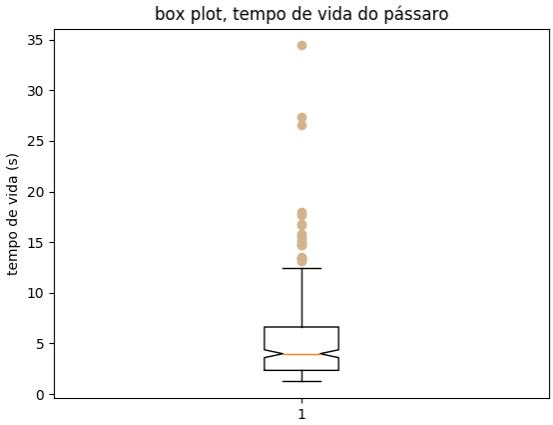

Figura 10: Box plot

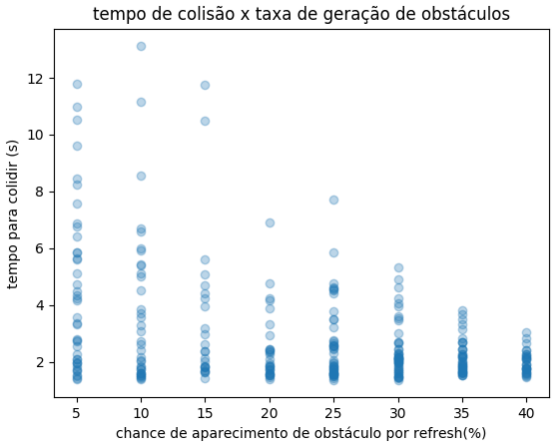

Figura 11: Tempo de colisão por geração

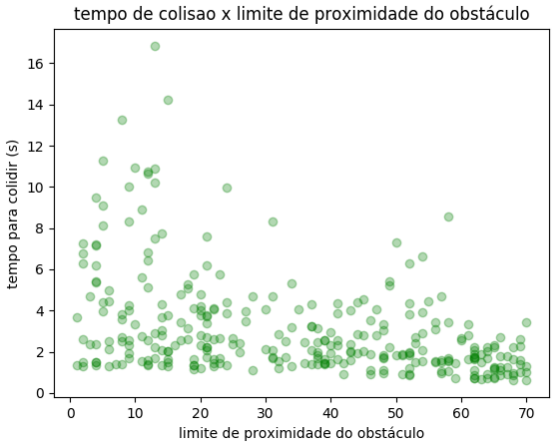

Figura 12: Tempo de colisão por proximidade

Resultados e Discussões

Os gráficos foram gerados testando resultados obtidos a partir de 300 instâncias diferentes em que foram variados os parâmetros utilizados no gráfico e os outros mantidos fixos, preenchidos com valores especificados na descrição do problema, como citado anteriormente.

- **Figura 9**: Esse gráfico têm como objetivo mostrar o número de colisões que ocorreram numa coordenada y. Olhando atentamente podemos observar no histograma que a solução para o problema está tendenciosa, pois a maioria das colisões ocorreram acima da coordenada *0* e nenhuma ocorreu abaixo dela. No mais podemos observar também que o pássaro prefere ficar entre as coordenadas *0* e *50* ao invés de ficar muito em cima ou muito embaixo, provavelmente isso ocorre por conta das restrições que colocamos através das regras nas quais o limita a ficar muito próximo do teto ou muito próximo do chão, portanto sua tendência é ficar no meio como podemos comprovar. Outro detalhe é que algumas decisões tomadas pelo pássaro resultam em sua morte iminente, assertado pela coordenada *200* que mostra *40* colisões ocorridas, decerto que alguns cálculos realizados pelo controlador fuzzy o faz acreditar que a melhor escolha é dar o tap priorizando, dessa forma, algumas ações sobre outras.
- **Figura 10**: como sabemos o boxplot considera dados como outlier através da regra de *1.5 IQR*. Podemos ver pela mediana (aprox *4s*) que aproximadamente *50%* dos dados têm tempo de sobrevivência maior que esse valor e *50%* sobrevivência menor, essa divisão não está proxima a uma divisão ótima entretanto o modelo tem um desempenho bom no melhor caso (aproximadamente *12,5s*). O pior caso obtido foi perto de um segundo o que faz sentido pois as bolas sempre aparecem no canto direito da tela, então a colisão tem chance de ocorrer apenas após o obstáculo chegar ao meio da tela (posição do pássaro).
- **Figura 11**: Esse gráfico busca ilustrar a relação do número de obstáculos gerados e o tempo de sobrevivência do pássaro. É evidente que quanto mais obstáculos são gerados por unidade de tempo menos o pássaro sobrevive, esse raciocínio faz muito sentido, uma vez que, se há muitas bolas no grid ele não vai conseguir desviar de todas, pois seu espaço é mais limitado. Para verificarmos essa convicção basta olharmos para o gráfico e para imagem abaixo.

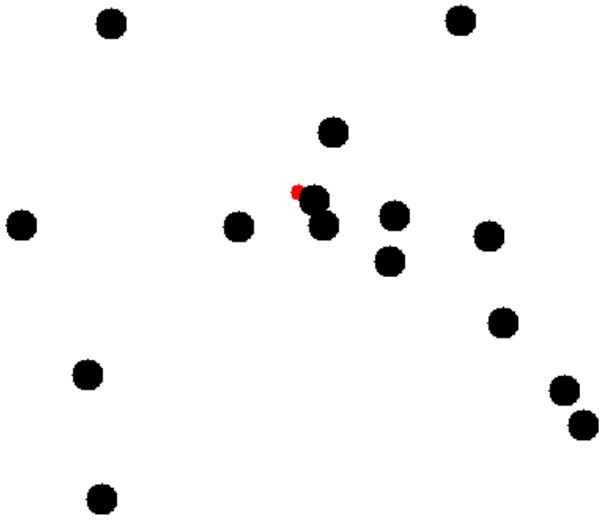

Figura 13: Poucos obstáculos

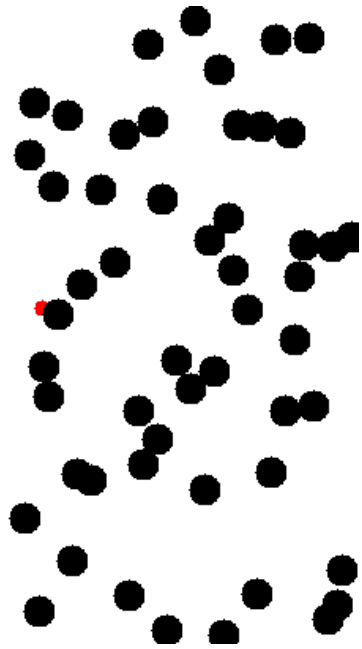

Figura 14: Muitos obstáculos

- **Figura 12**: Esse gráfico tem por objetivo ilustrar a relação entre a proximidade com um obstáculo e o tempo para colidir com ele, observando-o notamos que quanto mais longe menos tempo levamos para chocar, por

outro lado quanto mais perto menos tempo. Esse resultado acontece por conta do tempo de reação, se o pássaro está muito perto da bola o controlador fuzzy tem menos turnos para realizar suas computações e decidir o que fazer, logo o tempo para colidir vai ser maior se comparado com o caso no qual o pássaro está mais longe.

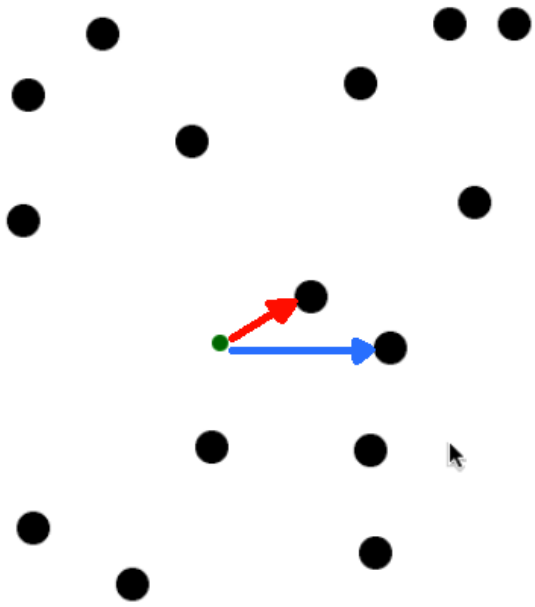

Figura 15: Mais perto, mais longe

Tabela de Correlação

1000 experimentos foram analisados variando todos os seguintes parâmetros a fim de obter o coeficiente de correlação entre cada um deles e o tempo de vida do pássaro

Correlação					
posição y inicial	chance aparição de obstáculo	diâmetro obstáculo	velocidade do obstáculo	v_y pássaro	threshold proximidade
0,10	-0,037	-0,13	-0,6	-0,026	0,01273

- **posição y inicial**: faz sentido o coeficiente ser positivo, com o pássaro mais pra cima, geralmente é possível tomar mais decisões até que o chão seja atingido, embora isso não seja sempre verdade pois depende da proximidade dos obstáculos e também da do teto.
- **chance aparição de obstáculo**: a chance de aparição de obstáculo deve ser realmente inversamente proporcional ao tempo de vida do pássaro, mas era esperado que a relação fosse mais próxima de -1
- **diâmetro do obstáculo**: conforme maior o diâmetro do obstáculo, maior a chance de colisão, então a correlação negativa está dentro dos conformes, entretanto era esperado que o coeficiente fosse mais próximo de *-1*
- **velocidade do obstáculo**: se o threshold de proximidade limítrofe é maior o pássaro tem uma mmovimentação mais limitada, o coeficiente é negativo de acordo com o previsto
- v_y **pássaro**: Conforme mais negativa a velocidade em y do pássaro menos tempo há para tomada de decisão até que o pássaro colida com o chão, a correlação obtida demonstra isso.

- **threshold proximidade**: se o threshold de proximidade limítrofe é maior o pássaro tem uma mmovimentação mais limitada, no entanto ele também tende a ficar mais longe dos obstáculos, assim como esperado obtemos um valor postivo mas não próximo de *1*.

Conclusão

A proposição do projeto de encontrar a solução para um problema usando lógica fuzzy é desafiadora, com isso em mente foi necessário encontrar um assunto pertinente e entendê-lo nesse contexto. Ao desenvolver o projeto foram necessárias adaptações para facilitar seu desenvolvimento tal como a divisão do código em pacotes, refatoração de algumas funções para adequá-las ao nosso projeto e o desenvolvimento do código em si.

Ao rodarmos o programa pela primeira vez já nos deparamos com alguns problemas que não esperavamos como o comportamento estranho do pássaro, mas ao conhecer melhor a biblioteca e o modo como ela funciona mudanças foram realizadas com o intuito de torná-lo o mais fuzzy possível e acretidamos piamente que a modelagem e o código condizem com os objetivos de desenvolver esse projeto.

No final, modificamos os parâmetros e fizemos outras mudanças no código com o propósito de extrairmos estatísticas e comparar os resultados, isso foi feito através de tabelas e gráficos dispostos ao longo do projeto e por fim realizamos uma análise crítica dos resultados para embasarmos nossos argumentos e finalizamos aqui nosso trabalho.