Super TuxKart Ice Hockey Image Agent

Weikang Cai
Eric Culbertson
Jaspreet Kang
Shelby Patzer

1 Introduction

Super TuxKart is a an open-source kart racing game which features a 2 vs 2 ice hockey mode. The goal of this game is for the karts to push a puck through the enemy goal. Through this project an image-based agent was designed to detect the puck's location from in-game images, and a controller was fine-tuned to act based on the agent's detections.

2 Approach

2.1 Controller

The initial controller relied on the puck's true location which was transformed into player screen coordinates using the kart's camera view and projection. This is so that once the model was completed, the puck's actual state could be replaced with the predicted location. The first controller determined acceleration based on distance from the puck and steer based on the angle from the kart's front to the puck. Another strategy involved one aggressor and one goalie; however, this track proved ineffective as the primary objective is to score points and not to defend. Lastly, the controller that worked best had similar logic to the first version, but included a few extra components. First, a "correcting" mode was added which reverses the kart if it is in danger of passing the puck. Additionally, the angle between the kart and the goal was used so that the kart tried to nudge the puck towards the enemy goal. It was further improved by having one player take a slightly more aggressive stance than the other in the beginning of a round.

Controller	Avg Points / Game
Initial Controller	0.6
Aggressor and Goalie	0.71
Final Version	0.89

Table 1: Performance of Controllers for 100 Games with Different Puck Locations and Team Placements. Puck image calculated manually from game state.

2.2 Data Generation

Using our controller, 16 matches were simulated between our custom agent and image jurgen agent, with each match initialized with a random starting puck location and velocity to ensure a robust set of training data was generated. From these matches, around 30,000 sets of images, ground truth puck locations (in screen coordinates), and segmentation masks of the puck were generated. The segmentation masks were then used to create binary classification labels to indicate whether the puck was present in the image or not. This data was split 80-20 for training and validation.

2.3 Model

Three types of models similar to HW3/HW4/HW5 were trained and tested:

1. Aim Point Prediction (similar to HW5):
    - Model Architecture: Fully-convolutional network design with 3 down-convolutional layers followed by 3 up-convolutional layers, and skip connections between each pair. Each down-convolutional layer consisted of batch normalization followed by one 3x3 convolution and ReLU. A 1x1 convolution along with a Softmax operation was applied to the last up-convolutional layer to predict 2D normalized "screen" coordinates of the puck.
    - Loss: During training, the L1 loss was used.

2. Point-based Object Detection and regression (similar to HW3 and HW4):

    - Model Architecture: Similar to the architecture used above for aim point prediction, except two outputs were predicted: a binary label indicating whether the puck is present in the image or not, and 2D puck coordinates. A 1x1 convolution along with a Softmax operation was applied to the last up-convolutional layer to predict 2D normalized "screen" coordinates of the puck. Additionally, a global average pooling layer followed by a linear layer was applied to the last up-convolution layer to generate a binary label for puck classification.
    - Loss: During training, multiple losses were optimized: Binary Cross Entropy loss for puck classification, and L1 Loss for puck coordinate regression. In instances where the puck was not in the image, the L1 loss was multiplied by zero.

3. Image segmentation and regression:

    - Model Architecture: Fully convolutional network with 3 down convolutional layers followed by 3 up-convolutional layers. A 1x1 convolution along with a softmax operation was applied to the last up-convolutional layer. Additionally, a spatial argmax was taken over the heatmap to produce the predicted model puck coordinates.
    - Loss: Loss was calculated over the heatmap via binary cross entropy and predicted coordinates via mean squared error loss. The presence of the puck was determined by the strength and size of the predicted heatmap, but no puck classification loss was computed directly.

Each model was trained for 50 epochs using the Adam optimizer with a learning rate of 0.005 and a batch size of 50 images. The results of each model are shown in the table below. Furthermore, image augmentation methods such as color jittering and horizontal flips were applied to prevent overfitting, and a "Reduce on Plateau" learning rate scheduler was used. After using each model to run games, it was discovered that the point-based object detection model performed much better.

Model Type	Measure	Train	Valid
Aim Point Prediction	loss	0.07	0.25
Point-based Detection	accuracy	99.7 %	98%
Point-based Detection	loss	0.011	0.015
Image Segmentation	accuracy	95 %	96%

Table 2: Model Performance on the Training and Validation Sets

2.4 Modifying the Controller

Once the model was trained, the controller was modified to utilize the model's predictions for the pucks location. One issue that arose was how to handle noisy predictions. One attempt to handle noisy predictions was to take the average over the last 5 predictions to determine whether the puck was on screen. If the model makes an errant prediction that the puck is offscreen, the last detected coordinate of the puck is used by the controller. If the model consistently predicted that the puck was offscreen, we modified the controller to drive in wide circles to search for the puck. A further issue, was that on occasion the agent directed the puck away from the opponent's goal. These issues were addressed in the final version of the controller. We also added drift (when the kart needs to make big turns), nitro (when the kart and puck are aligned), fire (when the kart is near the puck), and a rescue strategy to handle situations where the agent is stuck against a wall or inside the soccer net.

3 Conclusion

While many strategies were attempted, the tactic that worked best included a model that performed point-based object detection to locate the puck. Many versions of the controller were also tested, and the best one included the modifications listed above. While this project did not reach the goal of averaging one goal per game, many different designs were tested. After a final image detection architecture and controller was chosen, the image agent played against four opposing AI agents, and the final image agent averaged 0.75 goals per game.