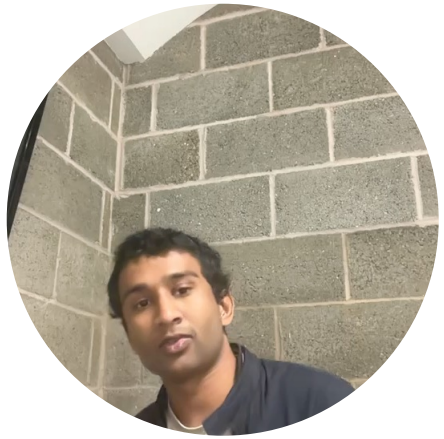
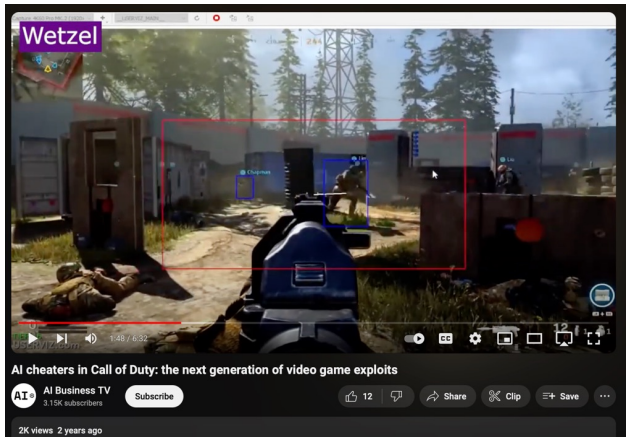
General Video Game AI: Learning from screen capture

Authors: Kamolwan Kunanusont, Simon M. Lucas, Diego Pérez-Liébana



Inspiration for this paper



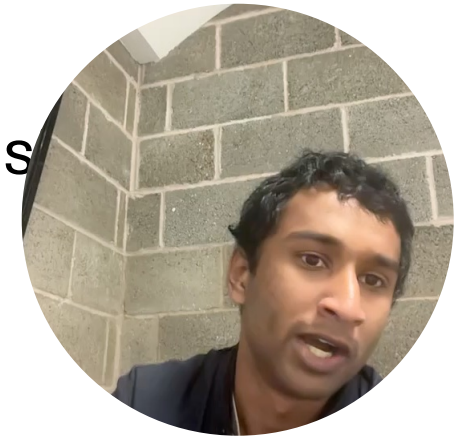
Personal fascination in producing an AI to mimic complex movement and AIM, whilst being undetectable to anti-cheat software

Use inputs only humans can see
(screen capture [not included in the paper but sound])
To create outputs (keyboard + mouse)

No special knowledge

Algorithm 1

- Real(prep)
- Input : (‘bsize’) [the size of the predefined image], screen captured image
- Output : preprocessed image as a 2d array
- Steps
- Capture the current screenshot
- Resize (based on ‘bsize’
- Normalise
- Extend (with padding if nessecary to meet the smallest image s
  allowed)

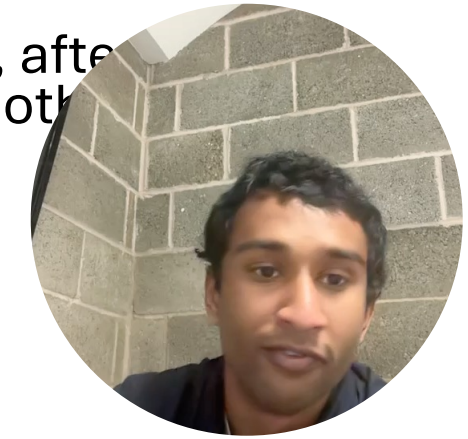


Algorithm 2

- Gen(prep)
- Input: grid representing game states, mapper to associate colors with different types of sprites
- Output : output is a preprocessed image in a 2d array
- steps
- Create new image (matching dimensions of this grid)
- Color assignment (for each sprite type in each grid cell, check if a previous color has been assigned, if not assign a new color + update mapper
- Normalise
- Extend
- Return (for use in ML algorithm)

Algorithm 3

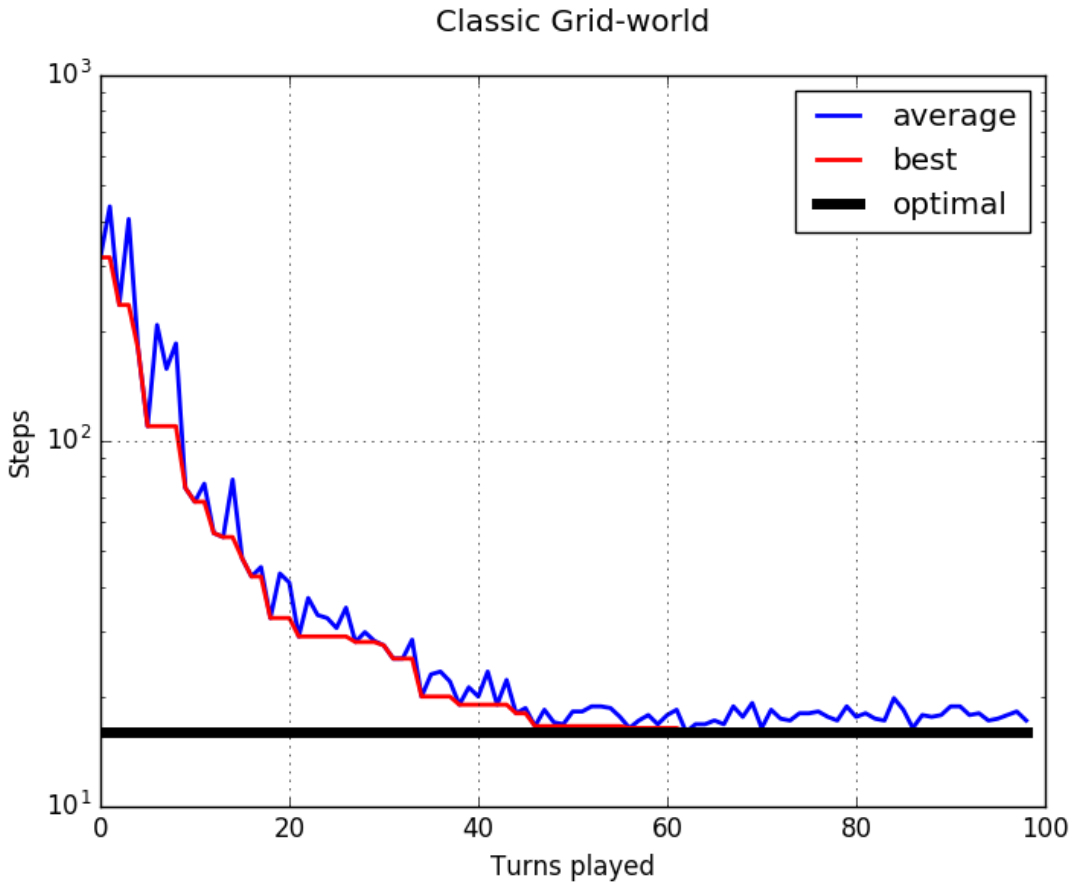
- Input : bsize, grid
- Otuputs: Action
- Steps
- Determine if Real(prep) [for GUI] or Gen(prep) [for no GUI] will be used,
- Initalise experience + model [uses ‘expPool’ to store game states, ‘exp’ actions and outcomes, and state-action value function ‘q’]
- For first image , it stores the image and selects a random action ‘act’
- After the AI, updates current experience with reward, it applies Q-learning, it picks random actions and then updates the Q-value based on the rewards
- Action decision using (1 – e) where e is the probability of a random action, afte image is fed an image, an action is chosen based on the models outputs, oth a random action is chosen.
- Process repeats



Deep Q-network

- Q-value is the expected reward based on an action
- The goal is to maximise action to reward
- Uses the bellman equation to recessively update the Q-value
- As its impractical to store q-values for every action, function approximators are used to store inputs and outputs for all available actions in a that state

Results


Classic Grid-world
10^3
10^2
10^1
Steps
0
20
40
60
80
100
Turns played
average
best
optimal

Future

- Undetectable AI, playing video games with superior mechanics than humans.
- Big problem for the video game industry