Introduction

Maze generation is an area of computational geometry and algorithm design, with applications ranging from game development to procedural content generation. Traditionally, mazes have been generated using various algorithms that create intricate pathways and dead ends, challenging the solver to find the correct route from the start to the finish. The typical resources we could find on maze generation focus on static images or predefined patterns. However, an unexplored side of this field involves dynamic content, such as videos.

Unlike static images, videos provide a sequence of frames that capture motion and changes over time, which can be leveraged to create more complex mazes. Despite the potential, there has been limited research on this topic (only one academic paper, which was not published).

Maze generation from images usually involves algorithms like Reaction-Diffusion [1] to get patterns, A* or Kruskal algorithm to turn it into a solvable maze [find something] ……

But when it comes to videos, the process becomes more complex. Videos consist of a series of frames, each representing a static image at a different moment in time. The temporal dimension adds a layer of complexity, as the maze generation algorithm must now account for motion, changes in the scene, and possibly varying levels of detail across frames. This dynamic nature requires advanced techniques in image processing, such as optical flow analysis [x] or frame differencing to extract meaningful patterns that can be translated into maze structures.

The paper we found [2] on this subject explores the use of video data to create mazes by analyzing motion and changes within the frames. The approach involves detecting edges and features in the video frames, tracking their movement over time, and then using this information to construct a maze that evolves with the video content.

We could imagine using such generation processes in advanced video game scenarios where the environment adapts dynamically to the player's actions.

Related works

Technical details

All codes described
Multi-threading for efficiency

Modifications and improvements

Time comparisons
Differences with the paper Amal sent

Observation and future works

Use of CNN for the remapping, as evoked in [2]

Conclusion

References

[1] le papier sur RD
[2] Moving maze