EE 586L Kinetetris

Hengyue Liu, Jason Pang, Yihao Xia

February 20, 2016

1 Abstract

Our project aims to develop a real-time image processing based Tetris game that can be competitive for 1-2 players. Essentially the player(s) can begin and play Tetris using hand gestures and arm motions, within a bounding box, that are visually captured using either a camera or a Kinect, and the game ends when the blocks are stacked above a certain height for the player. In the 2 player competitive mode, an additional element of random difficulty is added when clearing lines on one side generates random block drops on the other.

2 Description of the system

Our system includes (per player) a DSP board, camera, speakers, and a screen. The cost of a DSP board is around $30 [1], Kinect costs around $70 (used) to $115 (new), monitors can vary in price from $15 to $100+, and speakers can also vary from $20 to $100+, depending on the output quality (e.g. monitor resolution, sound quality). Then there are various cables to link the DSP board with the other components. Altogether this makes for a rather expensive prototype if one were to want to market the game. One interesting possibility for the system would be as part of an arcade game, where it would serve as something for groups of people to play together casually, and the hardware cost could then be covered by the arcade owner, since it seems unlikely that masses of people would want to own so much hardware (which also makes it difficult to port around).

Here is a simple flowchart for the prototype of the whole system:

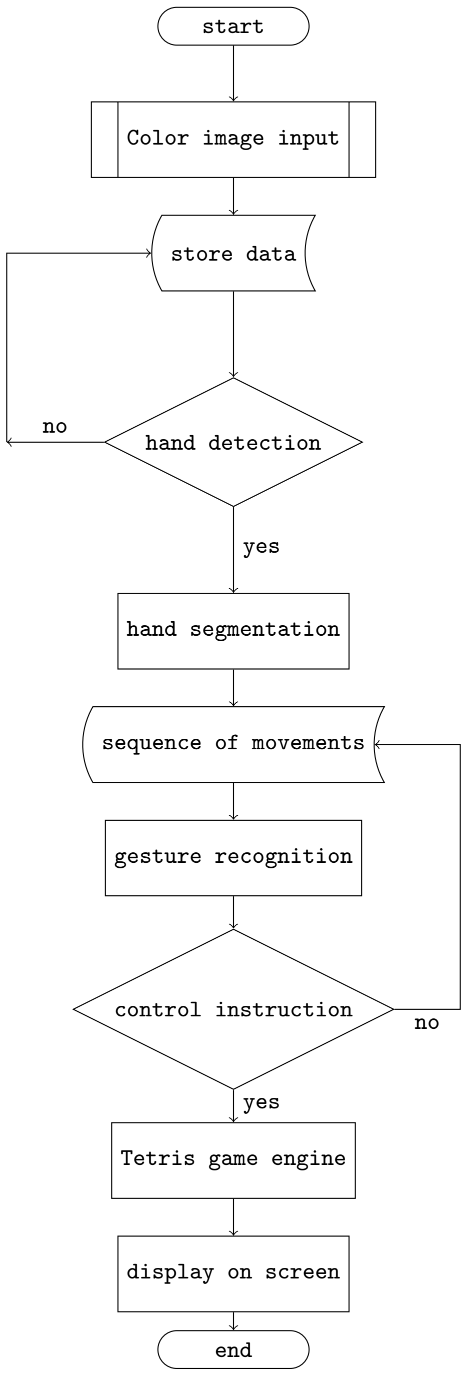

Figure 1: game system flowchart

Here is a simple flowchart for the game engine:

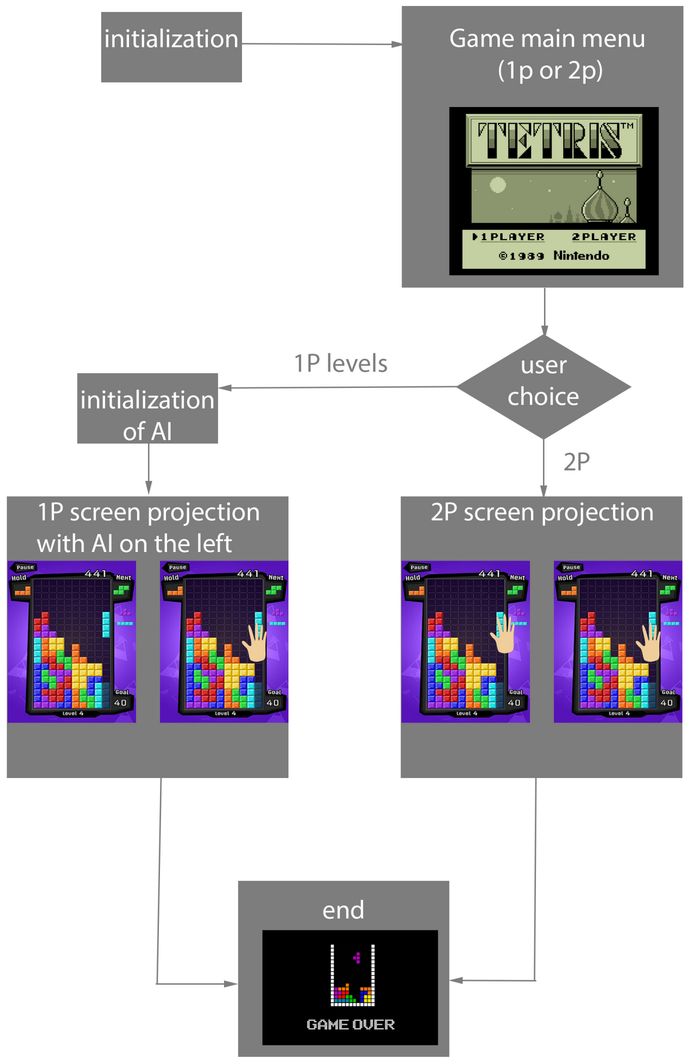

Figure 2: game engine flowchart

Game description: two simultaneous games of Tetris in 10x20 block grids on a split screen. Whenever a player clears a certain number of lines (e.g. clearing a total of 10x lines), random blocks get dropped into the opposing player field after a brief warning delay. In between the two screens is a center margin, where the stats from each screen will track the number of lines cleared. This region will also have a cartoon image of what each current players gestures are being read as. For example, a regular hand sign will indicate the default non-action move, a clockwise arrow will indicate rotation to the right, a counter-clockwise arrow will indication rotation to the left, and a flashing down arrow will indicate a block drop. The horizontal position of the block will be tracking the player hand positions. If we play over a desk (as a backup option) instead of in front of a screen, then the player horizontal hand position will instead indicate whether to move the blocks left or right (as well has how fast, scaling faster the longer the hand is displaced from the center). When a block is placed the exceeds the maximum grid height on either side, the game ends and the opposing side is victorious.

3 Description of Possible Algorithms

If we want to include a computer opponent, we could also write a Tetris solving algorithm [2]. The algorithm would attempt to search (to a certain depth level) the ideal move for a particular tetronimo (4 block piece) within calculation restraints, and perform the move based on that. The speed or delay of the algorithm, and the depth of the search would scale with the difficulty level of the AI.

- Hand detection

  We will use a color camera, so the players can have their hands on a desk or somewhere with single-colored background. Under ideal lighting, we could use envelope detection to get the outline of the hand, and track that. Alternatively, we could track the center of the hand, as well as the 5 fingertips, to make gesture recognition easier.If we have the player play over the output screen, we will need to also segment out their body. A promising approach is to use multi-feature fusion and template matching [3]. The template matching aspect is more vital for gesture recognition, as below.

- Gesture recognition

  For hand gesture recognition step, two techniques are used. One is dissimilarity measure, another is template matching [4]. We keep track of the hand variations between frames, the hand movement of one certain complete gesture will give a unique pattern of variations, then use that as a template. By calculation the similarity of this pattern, compare it to the template, if it is acceptable (within the thresholds), then the sequence of hand movement can be matched to certain gesture. While we dont know how viable it is, with enough training data, we might be able to obtain a robust template for gesture recognition.

The hand gestures recognition contains both spatial and temporal gesture segmentations, to determine where and when the gesture occurs. We need to establish a framework to segment the hand gesture spatially and temporally. To recognize the hand gesture, we need to extract the features of different gestures by employing some pattern recognition algorithm and then build a gesture models for incoming gesture recognition.

4 Complexity Analysis

The system will need to handle images at a rate that balances detail and memory capacity/ processing speed.Were going to aim for a rate of 15 frames per second at first, and then tweak it up if the system runs perfectly well, or lower it if the system struggles. Most of the memory will probably be taken up by a buffer (or buffers) that stores all the image data that are processed or going to be processed. We also will need to use the processed results to recognize gestures, swipes, etc., as moves, and write game code that runs at a realistic pace. In addition, we will probably have to write a game AI for computer players, and this AI will likely take up a fair amount of computational power to perform searches for the best moves (e.g. up to a depth of 2 moves ahead), and then perform these moves at a particular speed to simulate player difficulty. We might also incorporate random AI errors to simulate occasion human behavior.

In addition, a robust gesture recognition will likely require additional non-gesture recognition to prevent false recognition, for example if a movement resembles more than one gesture, we may need to either ignore the movement until the player makes the movement clearer , or allow for weighting so that the system will perform only the closest similarity gesture in the case of a majority (e.g. 80% similar to one gesture and 20% similar to another).

5 Major Challenges

Optimization, how to omit unintended moves, how to deal with the sub-gesture problem, how to recognize multiple hand gesture.

Sub-gesture Problem: false detection of gestures that are similar to parts of other longer gestures.

One major challenge that we are likely to encounter is: how fast can we run the game? In order to make the system more sensitive to intricate player movements, we have to incorporate more data, such as in the form of more frames per second. However, this will make the system more memory and processor intensive. On the other hand, if we dont try to obtain a system thats as fast as possible, players might notice or experience a lag in the gameplay, which would be very undesirable. Besides, allowing multiple candidates in the game increase computational time require us to further optimize the algorithm.

Another foreseeable challenge is figuring out not just how to recognize proper moves (such as moving blocks left/right, dropping blocks, rotating them), but also how to exclude unintentional moves (such as an arm movement that is part of a natural sway, or drifting position from fatigue, or ambiguous movements that could be meant for one type of move but is actually intended to signify another type of move). Various factors of the background would cause problems including changing illumination, low-quality video (tradeoff of calculation time), temporary occlusion, and background clutter.

Different users can perform movements at different speeds, so another challenge is figuring out the start and end of a gesture. When a user makes a move, for example swiping right for moving one block to right, at the end of this gesture, the hands will return to its original position afterwards, i.e. moving left. So, we can consider this change in frames to be the end of the gesture. In other words, if we assume the user has one or more preferred default position(s) then any return to default indicates that an action choice such as swiping has just been completed.

6 Modeling Tasks

We need to model the computer AI speed and how to control it to simulate real-time behavior (e.g. a novice computer would take a long time to make moves, while a master computer would make Tetris decisions quickly).

7 Human Factors

User evaluation will be based on a few things, focusing on balancing the challenge of the game with its enjoyment. For the challenge aspect, we want to be able to scale the difficulty for all skill levels (from easy to master). This is the most important part, because a game that is too difficult to play will be hard to enjoy because of the lack of progress, but a game that is too easy to play will be boring. Thus we will probably develop difficulty tiers and test it out by having various skill level players evaluate whether those tiers are suitably difficult and enjoyable (not just for themselves, but also for what they believe is suitable at those particular levels).

Another important human factor is that gesture recognition must be suitable for the vast majority of players (unfortunately there are cases that the system probably cannot be played, for example by someone who doesnt have any hands). This means that what the majority of humans recognize as a gesture should also be recognized as a gesture by the algorithm, regardless of any other variation in gestures between different people. Also, not just gesture variation between people, but also gesture variation for a single person over time (due to fatigue or other factors).

8 Training

We will need to design various levels of computer AI and optimize their behavior within certain standards. Training data set can be obtained by having various volunteers of different skill levels play the game, and in addition to measuring their performance against AI or other people, also request their feedback for how appropriate they felt the game and/or AI skill levels to be.

9 Rough schedule

(1) video input and data extract

(2) real time gesture recognition

- hand detecting
- gesture modeling
- gesture recognition

(3) game design

(4) optimization (might take a long time)

(5) quality of life improvements (visual/ audio quality improvements, user interface improvements, appeal)

10 Final Test Set-Up

Demo scenario- camera either facing the players, or pointed down at a tabletop with marked boundaries. We will first aim for a scenario in which the players play in front of a screen, with the camera observing the players hand before the screen. The camera input is (in terms of pixels), 480 height by 720 width, so the we can divide the halves as having two 480x240 Tetris games on the left and right side (with 24x24 pixel blocks), and the center 480x240 region displaying player statistics.

Output- split screen with Tetris game and feedback on both sides, speakers for music/sound effects.

11 Board

Need: DAVINCI TMS320DM6437 EVM, as well as a camera (Color TeleCamera, model number NCK 41CV), a screen to display the game, and probably speakers for background music and sound effects.

12 References

References

[1] `http://www.ti.com/product/tms320c6713b/samplebuy`.

[2] `http://totologic.blogspot.com/2013/03/tetris-ai-explained.html`.

[3] Z. Shujun L. Yun, Z. Lifeng. A hand gesture recognition method based on multi-feature fusion and template matching. In *2012 International Workshop on Information and Electronics Engineering*, pages 1678–1684. Procedia Eng., 2012.

[4] Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition with kinect sensor. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 759–760. ACM, 2011.