1. Goals/Background
   ○ The high level goal of this project is to be able to use a camera (e.g. a webcam; not using a DROID phone in this case) to capture live screen images from a game being played on an external system, then use edge, color, or other feature detection algorithms to construct a computationally simplified representation of the playing field, decide on a course of action, and send the chosen command(s) back to the external system in real time.
   ○ The game selected for this is Tetris Attack, a simple puzzle game that involves horizontally swapping blocks to clear them from the playing field by aligning 3 (or more) like-featured blocks horizontally or vertically; a losing condition is encountered when the ever-rising stack of blocks reaches the top of the playing field. An example game screen (taken directly from the screen output, rather than from a camera) is shown in Figure 1.

![Figure 1. Example Tetris Attack playing field.](image)

   ○ At a bare minimum, any candidate algorithm should be able to “play” the game in real time, clearing at least some number of blocks from the playing field before a losing condition is encountered at the slowest possible game settings.
   ○ As an ideal state, an image processing algorithm should be fast enough to allow the gameplay logic to react quickly enough to “play” the game indefinitely at the highest “difficulty” – the fastest possible scrolling rate with the shortest latency between clearing blocks from the field.
   ○ Intermediate benchmarks of performance exist in the form of scrolling rates at the losing condition, scores achieved (based on numbers of blocks cleared), and performance relative to several pre-existing game-playing algorithms provided by the game itself.
   ○ Since a camera will be used instead of directly capturing screen output, the algorithm must be tolerant of conditions such as skew/rotation and adverse lighting conditions.

2. Work to be Done
   ○ Identify and characterize the playing field.
     • The Tetris Attack field is always a 12x6 array of blocks, all of which have seven defined shape+color combinations, and empty space, superimposed upon a dimmed background image. The algorithm must be able to identify these blocks (by their shape, color, or some combination of the two), determine their relative position in the playing field, and represent this in a simple, compact data structure.
   ○ Identify the cursor.
     • In a similar vein to identifying the playing field, the algorithm should also be able to locate
the current position of the control cursor on the playing field.

- **Evaluate and act.**
  - With information about the current state from the image processing portion of the algorithm, the gameplay algorithm should select a block to switch and send inputs to the external system to work toward achieving this (in this case, delivering keypresses over a network using an open-source keyboard/mouse virtualizer called Synergy: [http://synergy-foss.org/](http://synergy-foss.org/))

3. **References**