AutoPuzzlr

Taking the fun out of puzzles

Andrew Conduff
Connor Maggio
Aneek Mukherjee

Use Case

AutoPuzzlr is a guided-puzzle-solving system to make it easier for users to solve puzzles! The tabletop device will guide users by directing them on where to place a piece, one at a time, until the puzzle is complete. AutoPuzzlr will make it easier and faster to solve puzzles.

We want AutoPuzzlr's MVP to help users with puzzles that are 20"x20" and ~250 pieces.

Serves as a proof of concept for future extensions of this technology.

AutoPuzzlr involves Signals (the CV algorithm) and Software.

High-Level User Requirements

Name	Requirement	Description
End-to-End Suggestion Latency	<4 seconds	AutoPuzzlr must calculate the suggested location of a piece within 4 seconds of recognizing a user tap.
Suggestion Precision	<0.5 inch	AutoPuzzlr must suggest a location within .5 inch of the piece's actual location.
Suggestion Accuracy	90%	AutoPuzzlr must meet the precision requirement for at least 90% of the suggestions.

Technical Requirements - Camera & Projector

Name	Requirement	Description
Camera Field of View	100% of workspace	AutoPuzzlr's camera must be able to see the entire workspace (defined by the 4 legs).
Camera Sensor	Good color identification and contrast 12+ MP resolution	AutoPuzzlr's camera sensor must have high enough resolution and color detection to identify features in <1" puzzle pieces.
Projector Image	100% of workspace	AutoPuzzlr's projector should project an image that covers the entire workspace (defined by the 4 legs).
Projector Brightness	Sufficient to be visible against workspace	AutoPuzzlr's projector requires sufficient brightness and contrast to be visible against the workspace (puzzle pieces and background).

Technical Requirements - Puzzle Solving

Name	Requirement	Description
Tap Detection	<50 ms	AutoPuzzlr must detect a tap and its location in the camera frame within 50 ms.
Piece Identification	<500 ms	AutoPuzzlr must identify the piece being tapped within 500 ms.
Piece Matching	<3 seconds	AutoPuzzlr must identify a probable location for the puzzle piece within 3 seconds.
Response Latency	<50 ms	AutoPuzzlr will display a response animation within 50 ms of a user action (tap) or suggestion.

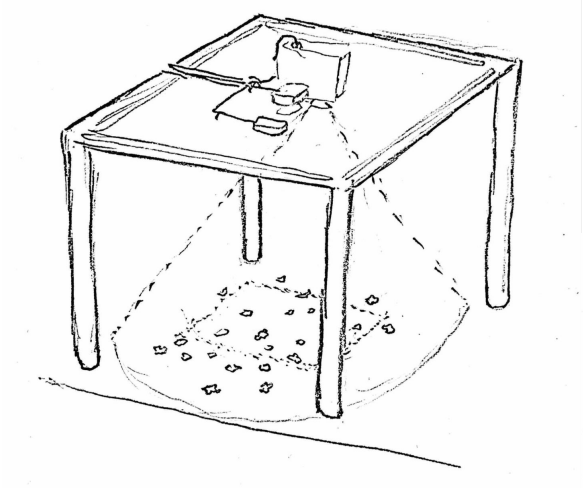
Testing, Verification, Metrics

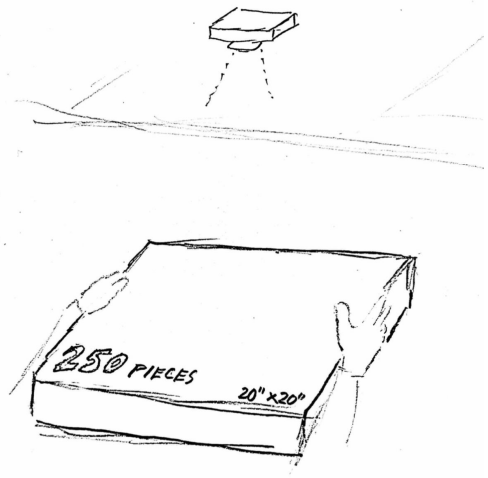
For each requirement we will have a test with tests for multiple data points. As a software dependent project, we will have many tests which use time libraries in the languages that we use (primarily python), to see how long each computation actually takes.

1. Timing individual functions and libraries to compare against our set requirements on slides 2-5 for time based requirements.
2. Distance requirements will be measured by hand from where the AutoPuzzlr says the piece should go and where it should actually go.
3. Testing against smaller puzzles and working our way up. (25-piece, 50-piece, 100-piece, 250-piece)

Solution Approach

1. AutoPuzzlr will consist of a 4-legged platform on a table with our camera and our projector with the ability to project on top of our workspace.

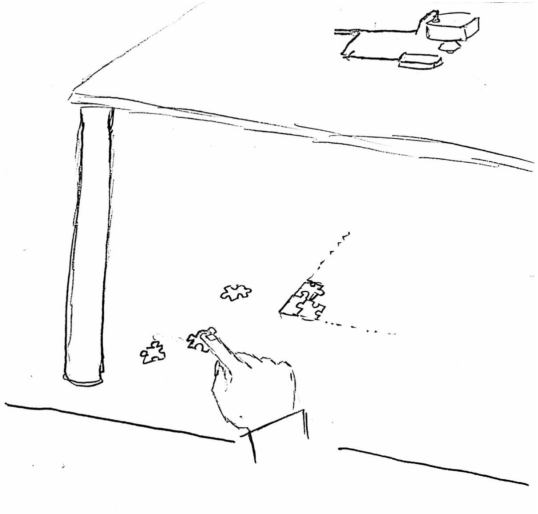




2. The puzzle will start unsolved with all picture sides of the pieces facing up. The user will need to scan a the front of the box or enter a picture of the final result

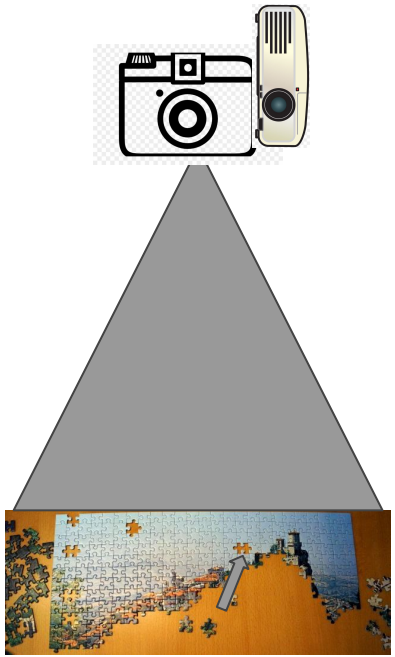
Solution Approach

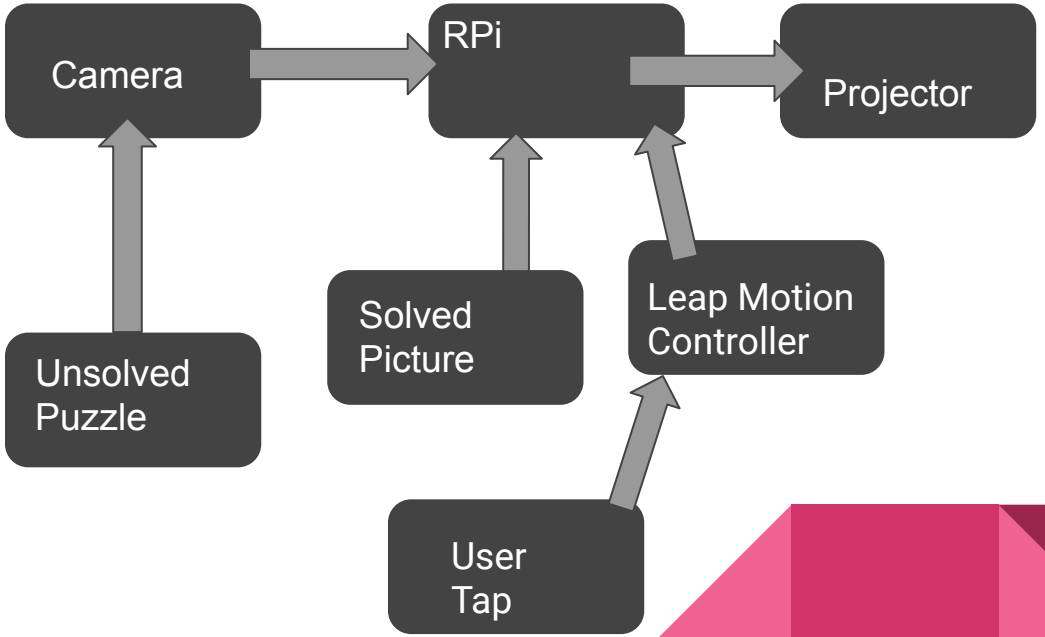
1. Then, the user taps a piece to be solved, and we segment out the individual puzzle piece and extract features from the picture on the piece, and compare them with that of the completed puzzle image.



2. When our algorithm is confident that has found where that piece will go, it will then project an arrow between where that puzzle piece is and where it should go, and this will continue until the puzzle is completed.

Solution Approach - System Architecture

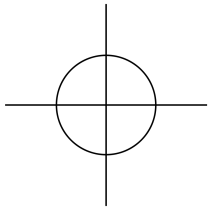




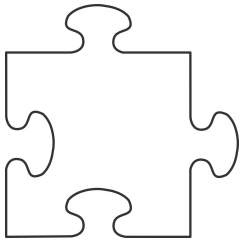
Solution Approach: CV Algorithm Pipeline

User Tap

Interpret Location



Segment out SIngle Piece



Feature extrac of Piece

Match up piece features and orientation

Display

Tasks and Division of Labor

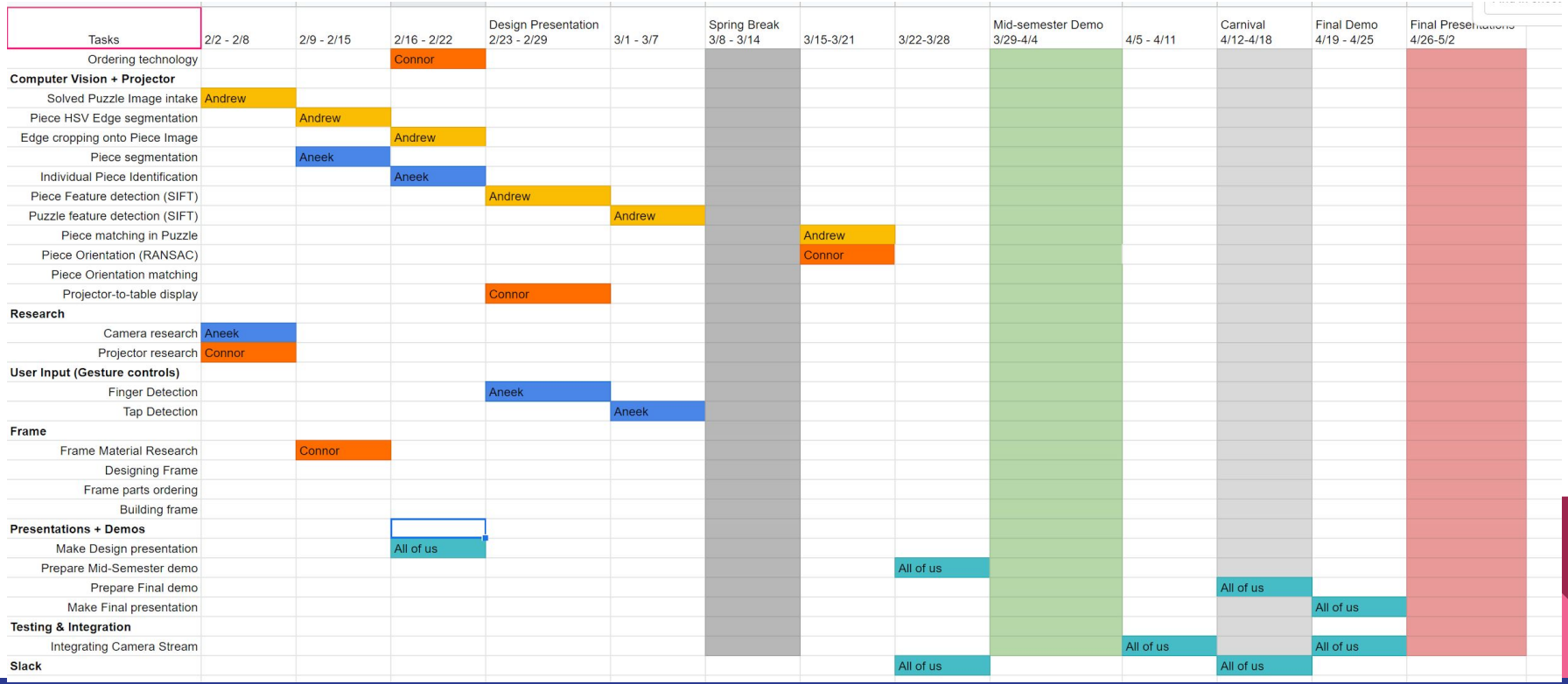
While we will all be working collectively on some tasks:

Andrew will take the lead on the Computer Vision portions

Aneek will take the lead on the Leap Motion Gesture tracking and integration

Connor will head the Projector interfacing as well as general software engineering design for our project.

Gantt Chart



Tasks	2/2 - 2/8	2/9 - 2/15	2/16 - 2/22	Design Presentation 2/23 - 2/29	3/1 - 3/7	Spring Break 3/8 - 3/14	3/15-3/21	3/22-3/28	Mid-semester Demo 3/29-4/4	4/5 - 4/11	Carnival 4/12-4/18	Final Demo 4/19 - 4/25	Final Presentations 4/26-5/2
Ordering technology			Connor										
Computer Vision + Projector													
Solved Puzzle Image intake	Andrew												
Piece HSV Edge segmentation		Andrew											
Edge cropping onto Piece Image			Andrew										
Piece segmentation		Aneek											
Individual Piece Identification			Aneek										
Piece Feature detection (SIFT)				Andrew									
Puzzle feature detection (SIFT)					Andrew								
Piece matching in Puzzle							Andrew						
Piece Orientation (RANSAC)							Connor						
Piece Orientation matching													
Projector-to-table display				Connor									
Research													
Camera research	Aneek												
Projector research	Connor												
User Input (Gesture controls)													
Finger Detection				Aneek									
Tap Detection					Aneek								
Frame													
Frame Material Research		Connor											
Designing Frame													
Frame parts ordering													
Building frame													
Presentations + Demos													
Make Design presentation			All of us										
Prepare Mid-Semester demo								All of us					
Prepare Final demo											All of us		
Make Final presentation												All of us	
Testing & Integration													
Integrating Camera Stream										All of us		All of us	
Slack								All of us			All of us		