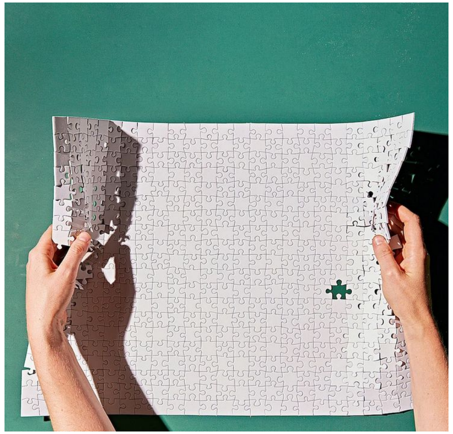
Jigsaw Puzzle Sorter

Computer Vision Project

Tanisha Basrai, Richard Jeong, Krishaan Patel

Background

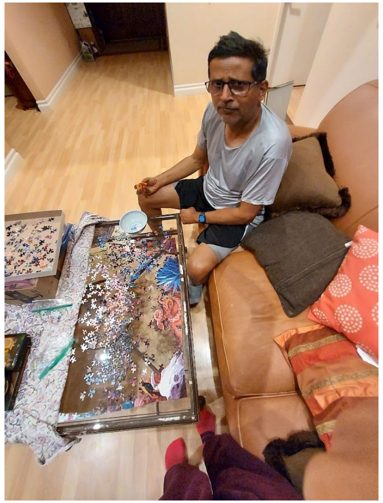
- Jigsaw puzzles are common worldwide activity
  - $6.69 billion in 2022
- Jigsaws are about *finding patterns*
  - Visual cues: color, shape
  - Sometimes only shapes
- How do you start a jigsaw puzzle?
  - Common approach: sorting edge/corner pieces from non-edge pieces
    - Grueling process for large puzzles



"Incomplete White Piece Puzzle"

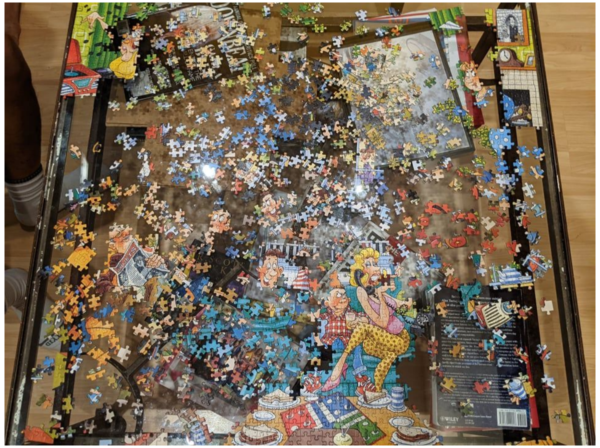
source: https://yetch.store/products/incomplete-white-puzzle

Real life puzzler (clearly upset)

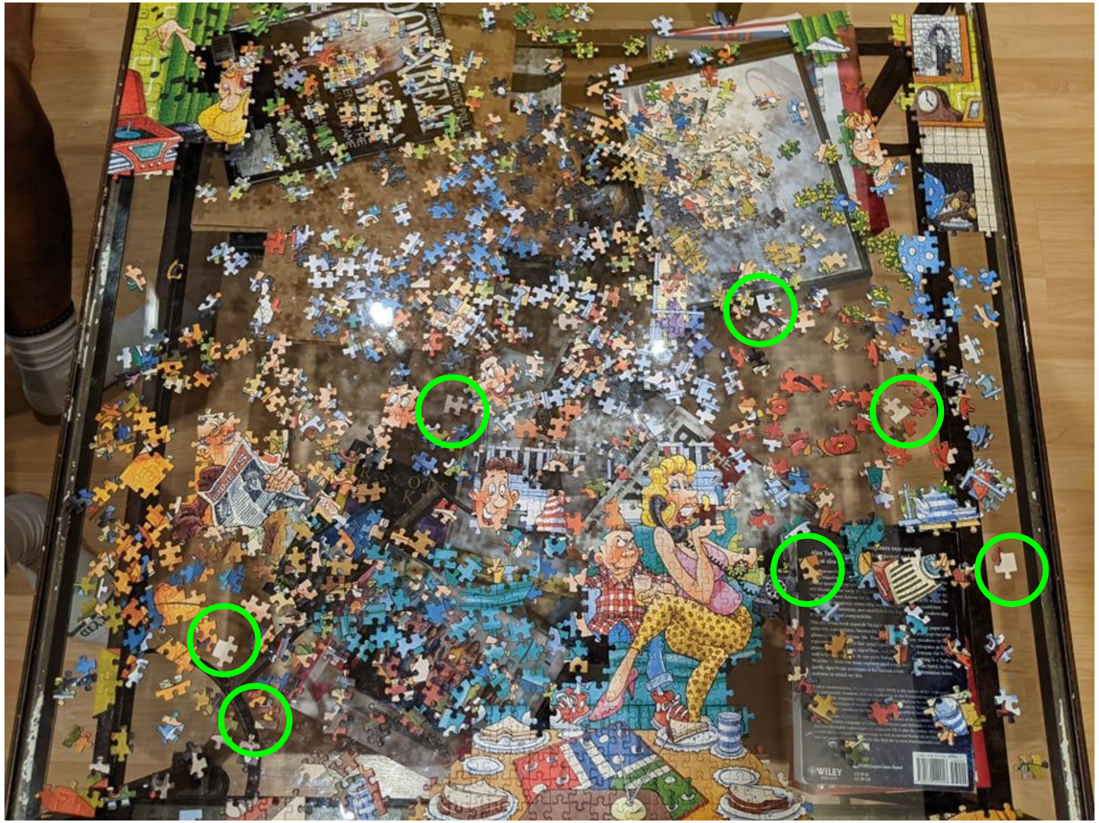


Problem

- Time consuming
  - 1000+ pieces can take *hours* to sort
- Visual and physical clutter
  - Pieces easily lost
    - Physically (dropped, eaten, etc)
    - Visually
- Loss of joy!
  - Human error is frustrating
  - Jigsaws should be fun



Quick! Can you find the any of the missing edges?



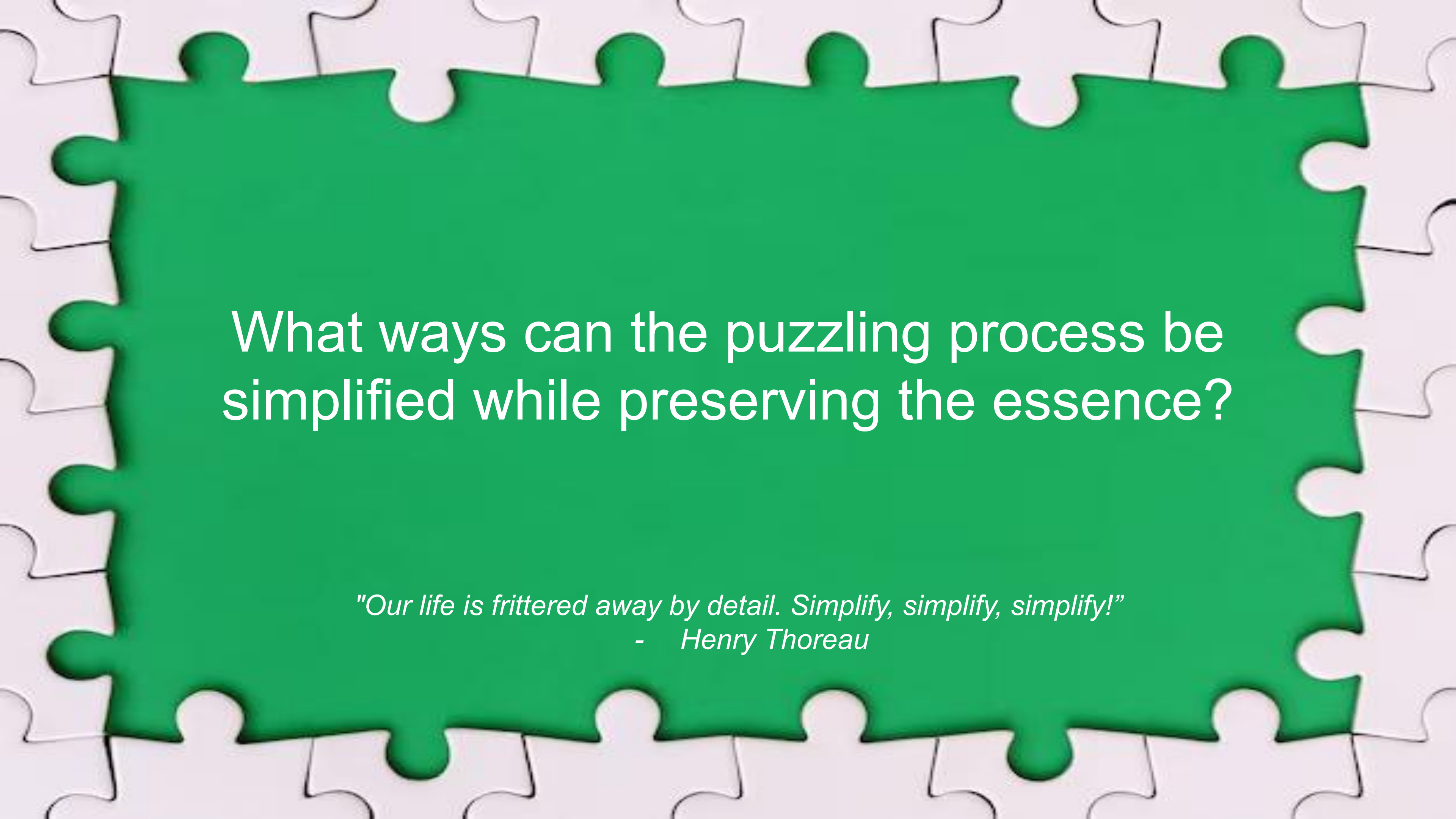
Current Solutions

- Private, personal projects
    - Not publically nor widely accessible
- Jigsaw puzzle solvers
    - What’s the point?
- In short: there are none

Worlds hardest jigsaw vs. puzzle machine (all white)

Stuff Made Here

4.19M subscribers



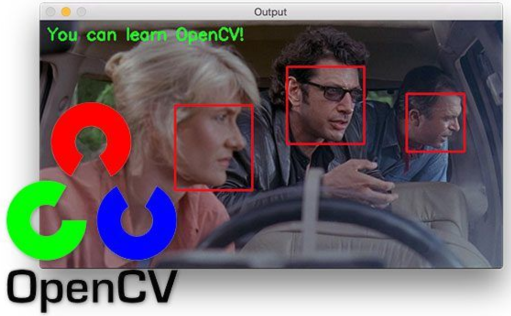
What ways can the puzzling process be simplified while preserving the essence?

"Our life is frittered away by detail. Simplify, simplify, simplify!"

- *Henry Thoreau*

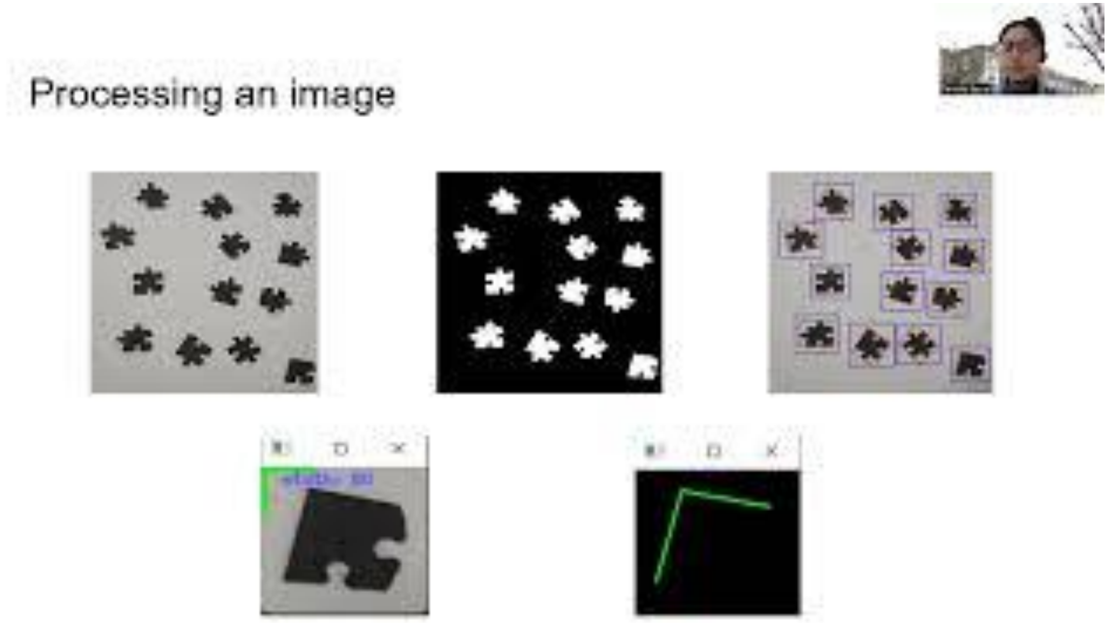
Our Solution

- Automate sorting corner, edge, and non-edge pieces
    - Give user classifications that are
        - Quick
        - Correct
- Most of puzzling process remains intact
    - Lower margin for human error
- Use computer vision
- OpenCV
    - Computer vision framework
    - Popular for image processing
    - Used for edge detection, Hough line transform



Tanisha is travelling… here is a pre-recorded presentation

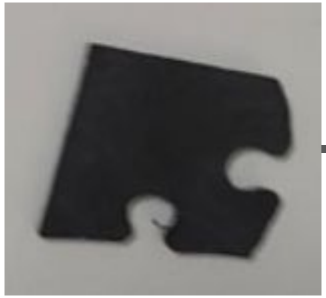
https://youtu.be/TpNnJKYKb0A



Processing an image

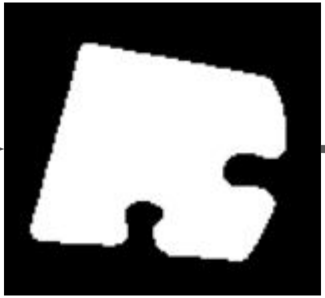
Piece Classification Overview

Raw image



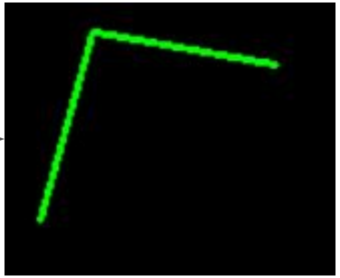
1) Grayscale
2) Blur
3) Threshold

Thresholded Image

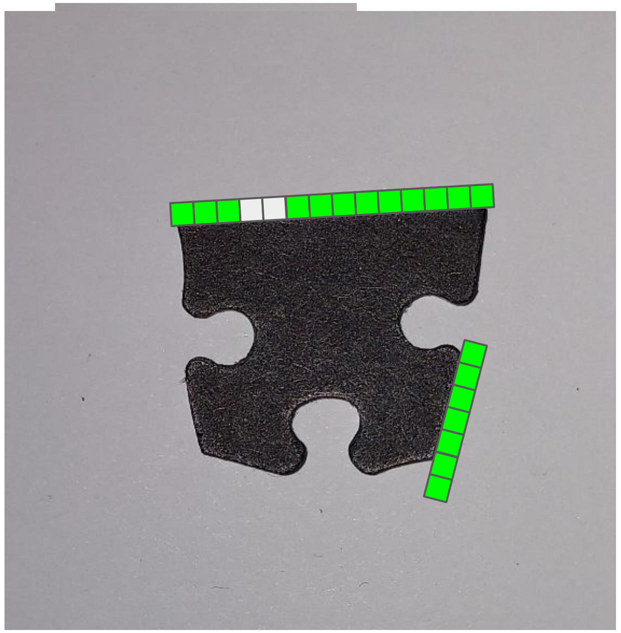


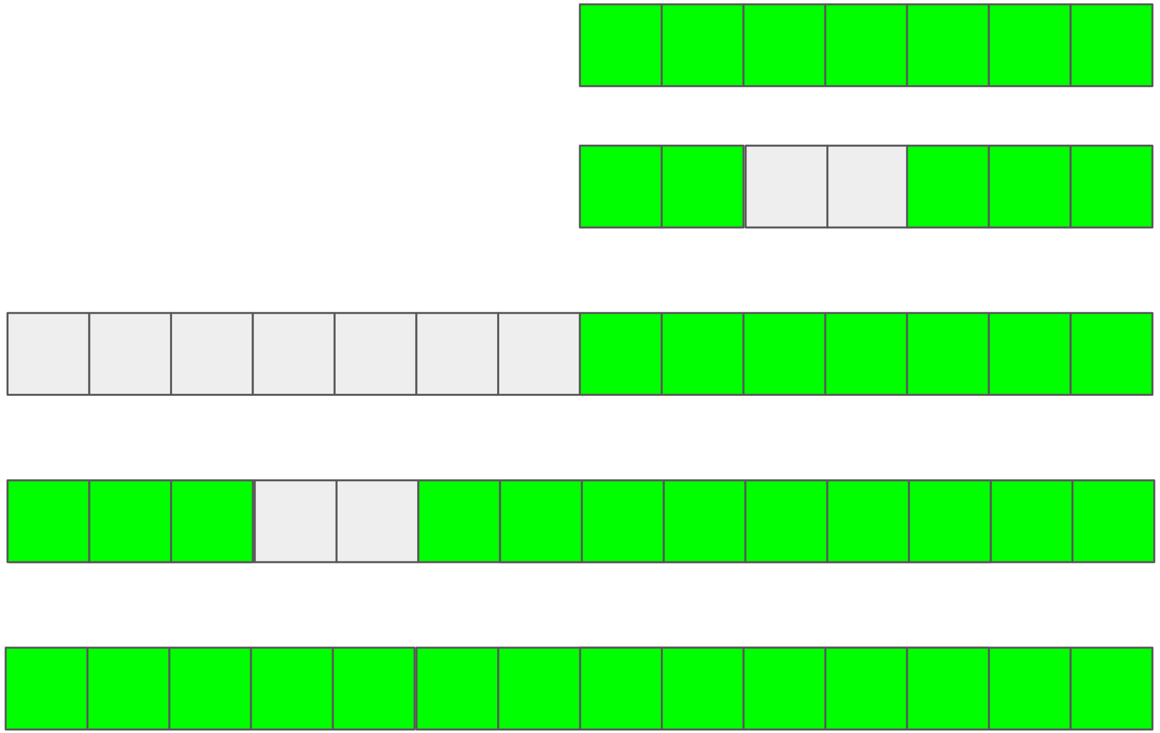
1) Define width
2) Define resolution
3) Detect straight lines

Detected Straight Lines



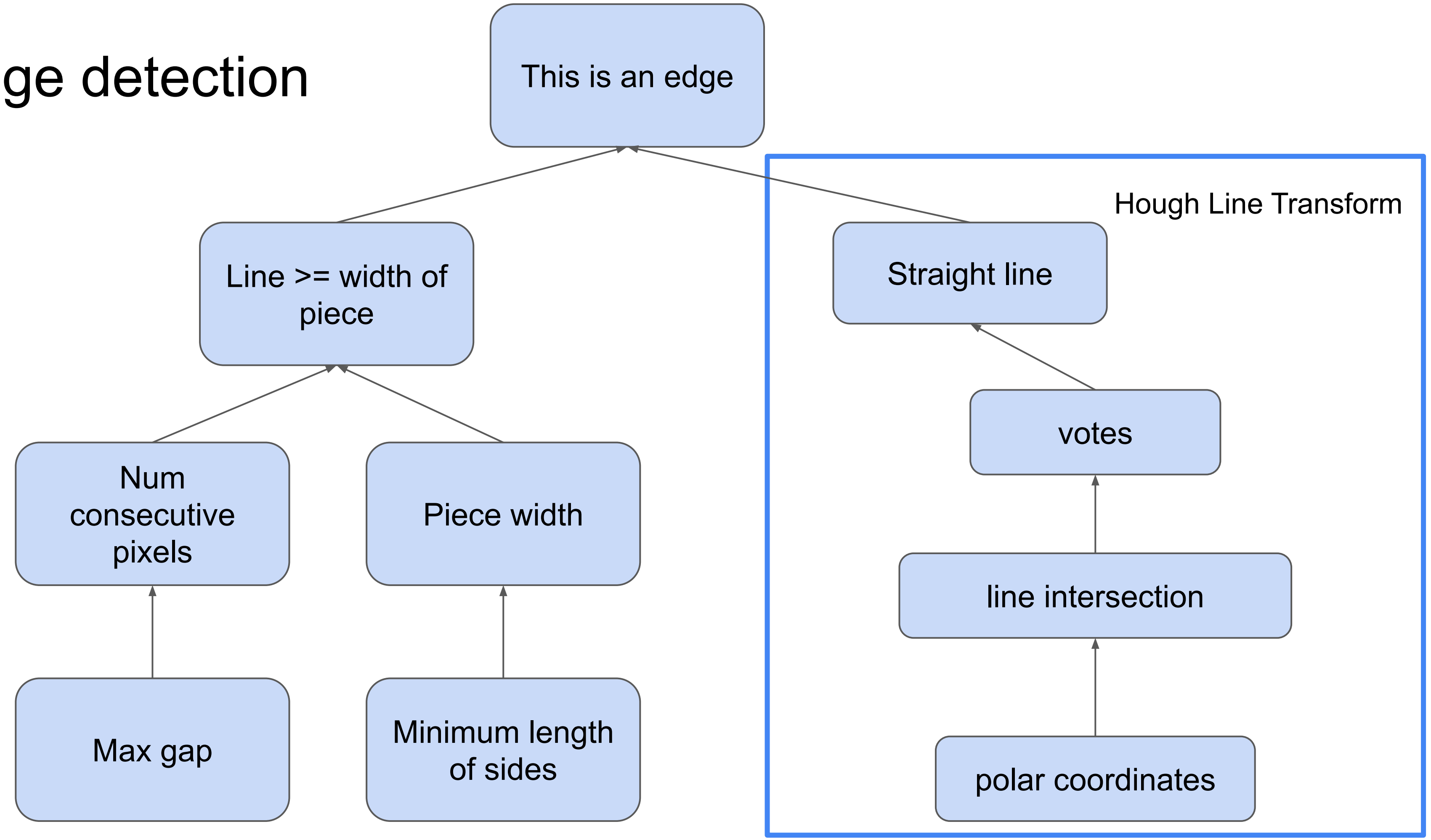
Defining an edge



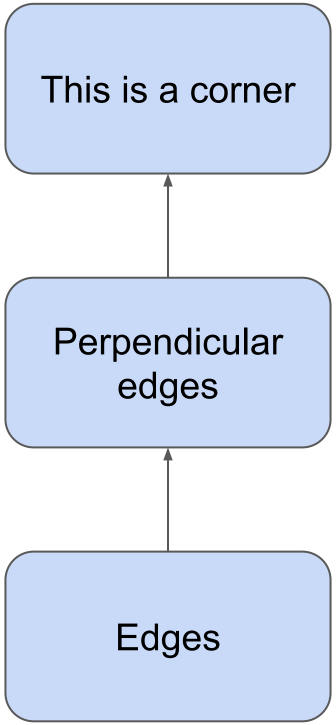


Which is an edge?

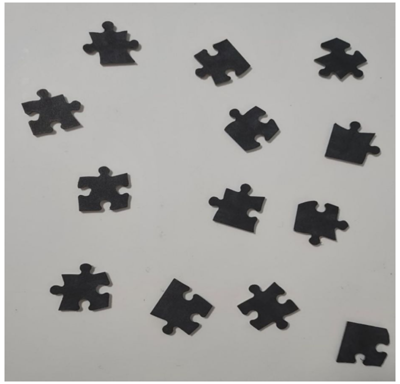
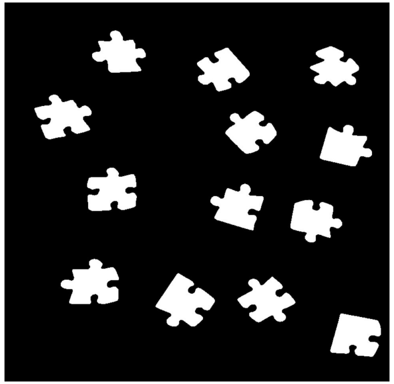
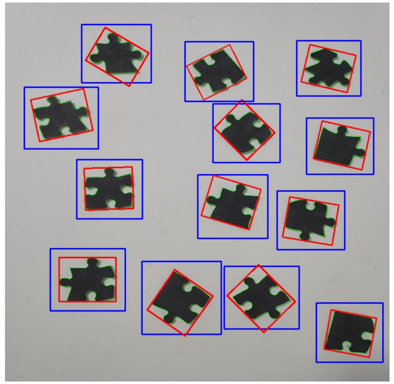
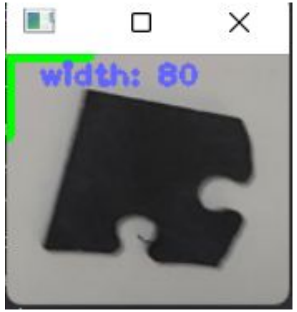
Edge detection

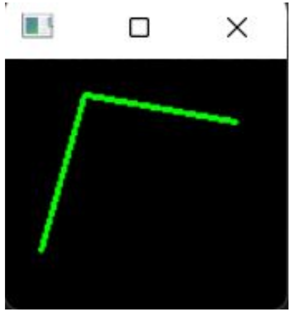


Corner detection

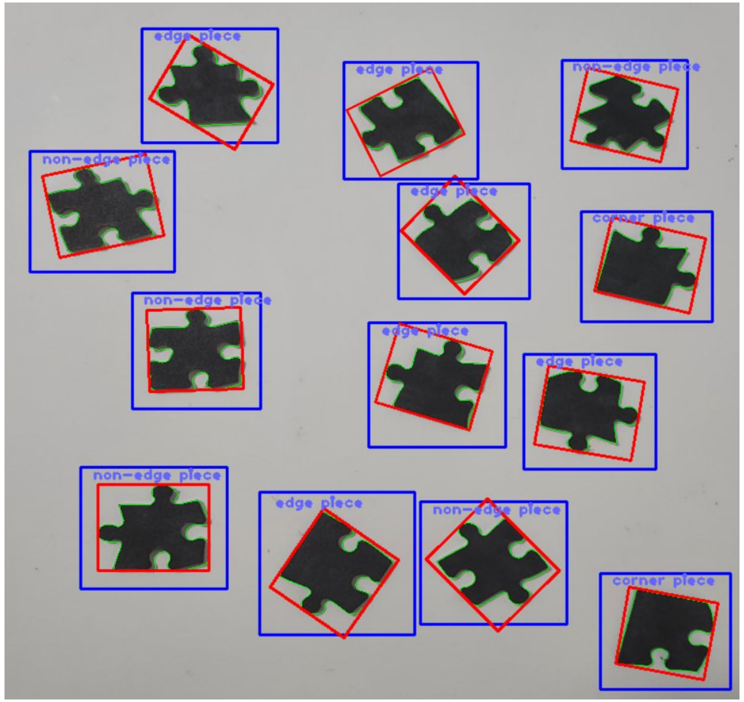


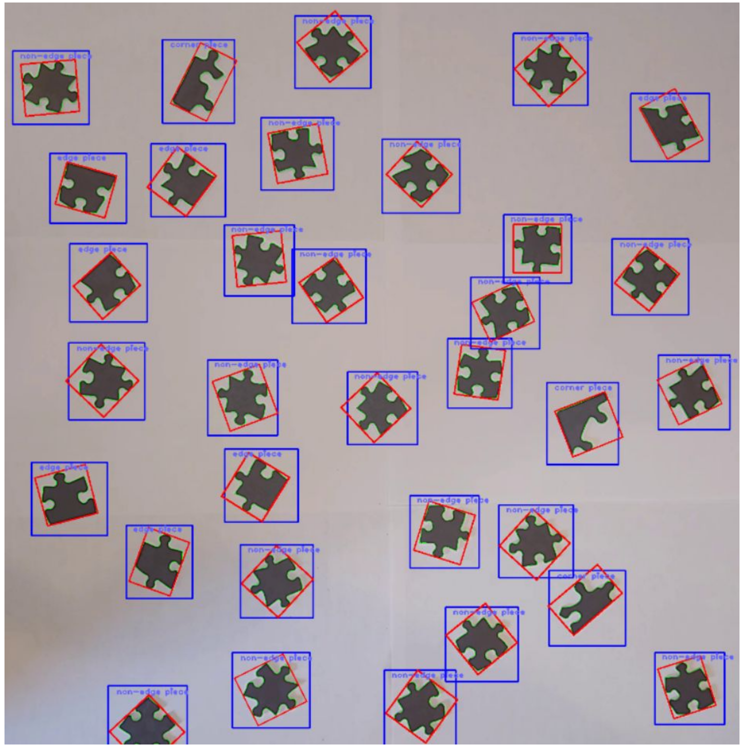
Processing an image

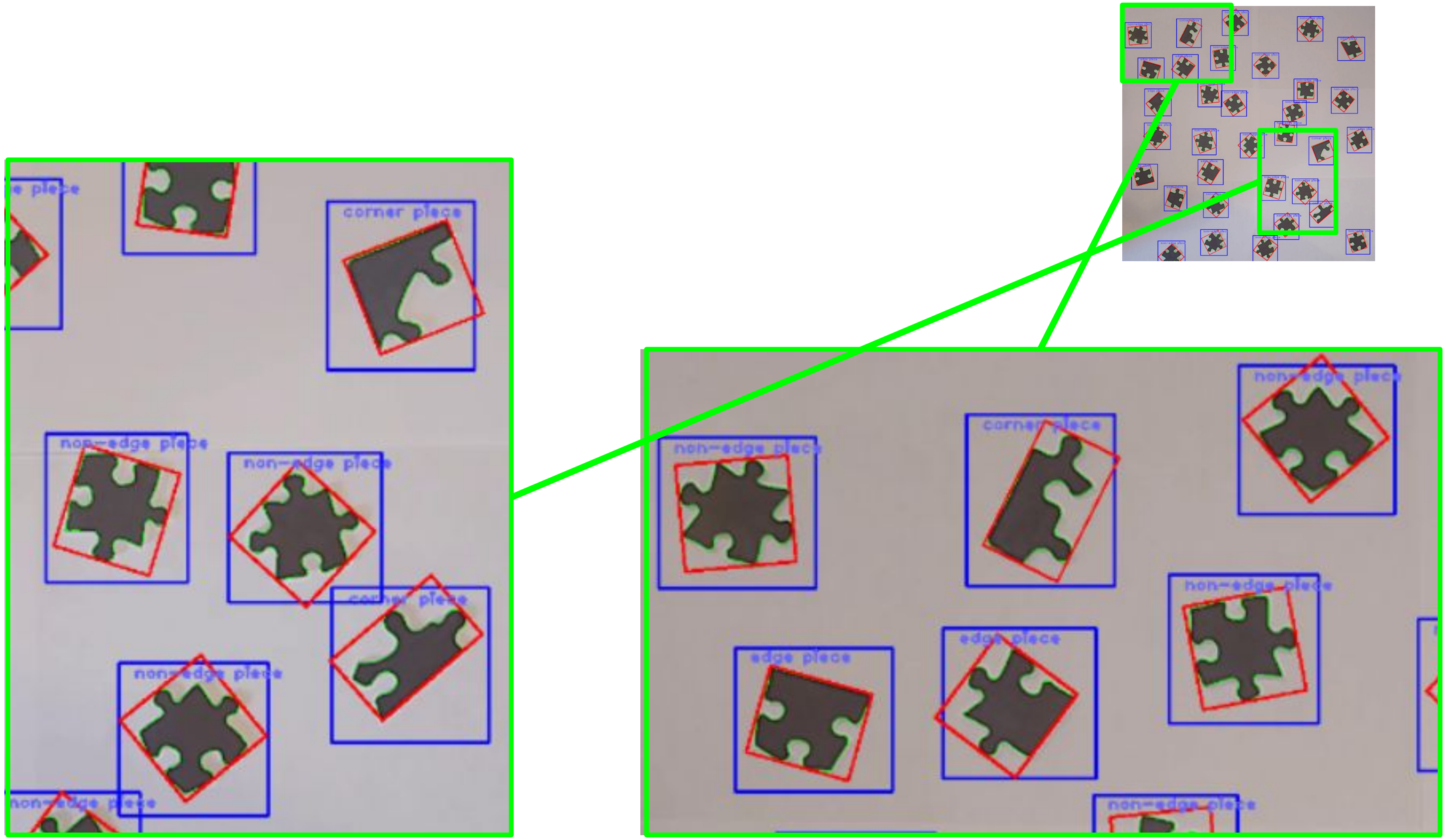







Results

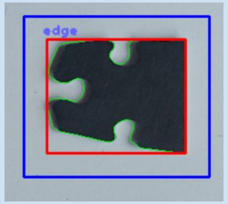

edge piece
edge piece
non-edge piece
non-edge piece
edge piece
corner piece
non-edge piece
edge piece
edge piece
non-edge piece
edge piece
non-edge piece
corner piece

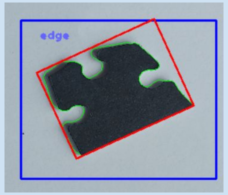


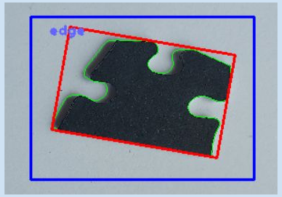

corner piece
non-edge piece
non-edge piece
corner piece
non-edge piece
non-edge piece
non-edge piece
non-edge piece
corner piece
non-edge piece
edge piece
edge piece
non-edge piece

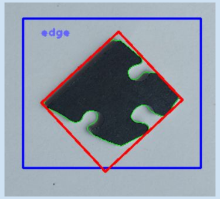
Tests

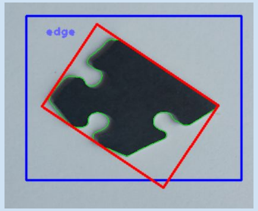
Edge Rotations



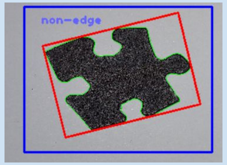


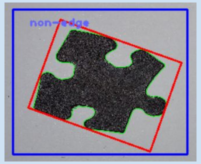


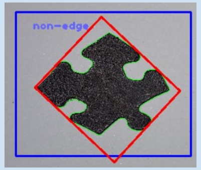


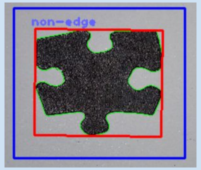


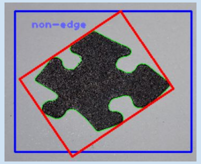
Non-Edge
Rotations Check

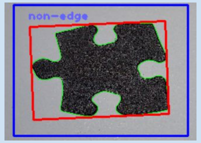




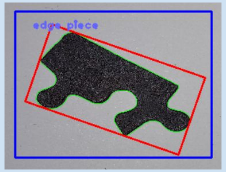


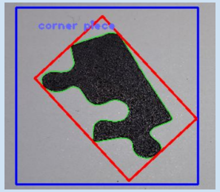


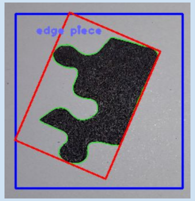


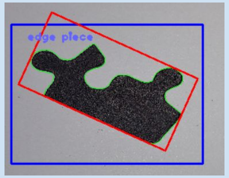


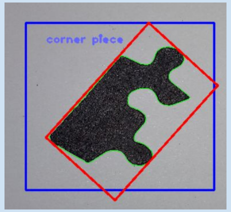
Funky Corner
Rotations Check

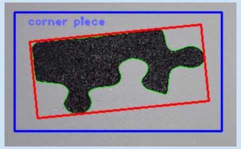












Evaluations

- Correctness: Good
  - False positives: Acceptable
  - False negatives: Undesired
- Robustness: Poor
  - Bad lighting
  - Non-puzzle pieces
  - Some piece shapes unclassifiable
    - Very small edges
    - Oddly shaped
    - Not solid color
    - Not contrasting color to background

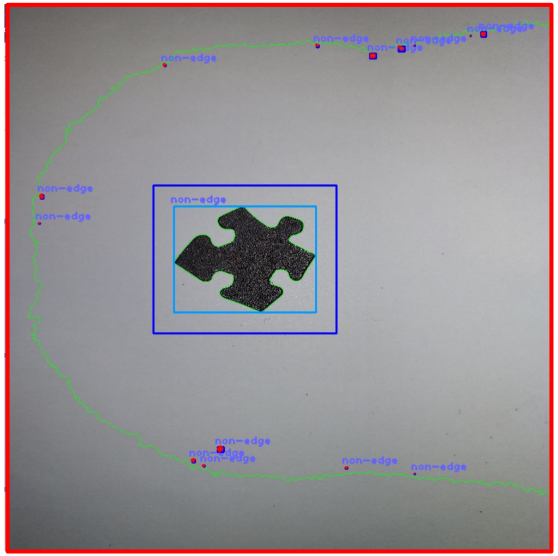
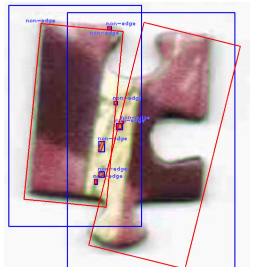
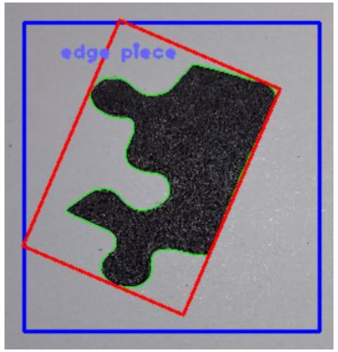


Image with poor lighting (shadows and artifacts)



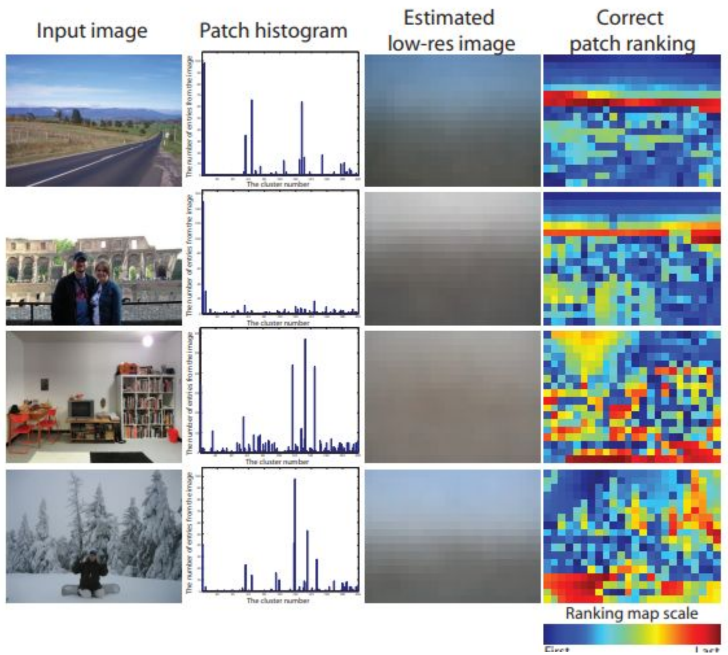
Colored piece (not highly contrasting, has image)

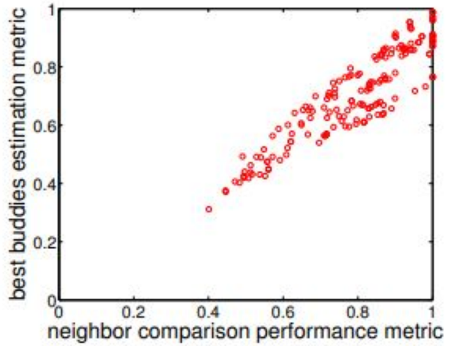


Irregular piece (very small edge)

Comparisons

- State of the art implementations:
  - Greedy Square Jigsaw Puzzle Solver (Ben-Gurion University)
    - Greedy solver (novel part compatibility & solution estimation measures)
  - Puzzle Machine (Stuff Made Here)
    - Embedded system to scan individual pieces
  - Probabilistic Jigsaw Puzzle Solver (MIT)
    - Probabilistic approach using graphical models

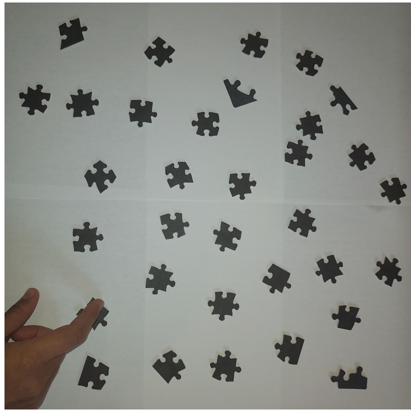


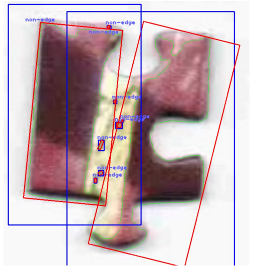


Moving Forward

Flask
web development,
one drop at a time

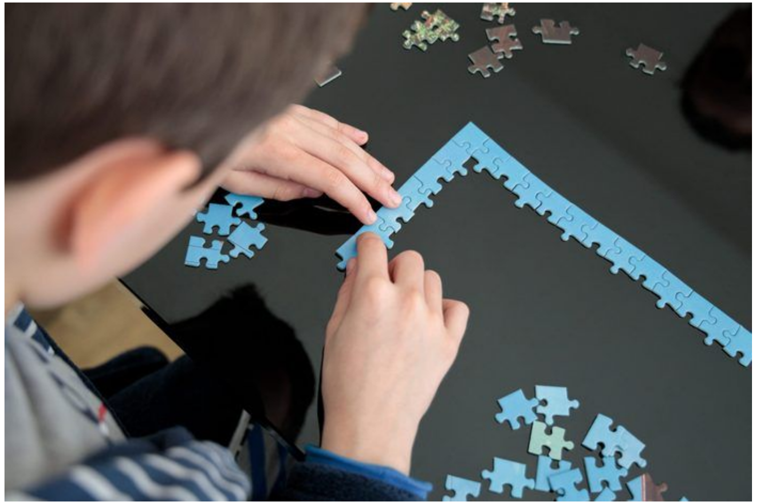
- Website for ease of use
  - Using **Flask**
    - Microframework to integrate python code into html/css/js
- Implement machine learning
  - Only process puzzle pieces and ignore other objects
  - Better video processing
  - More robustness
- Testing Scalability
  - Have only tested on sets of max 30-40 pieces

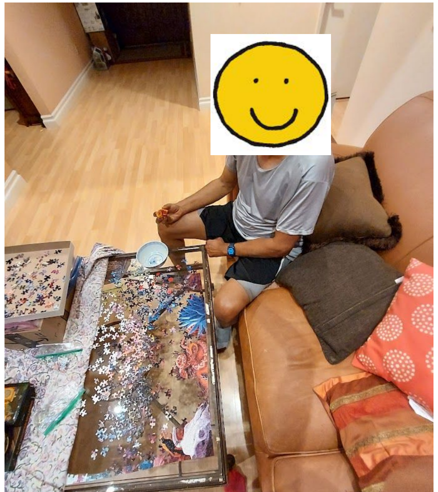




Conclusion

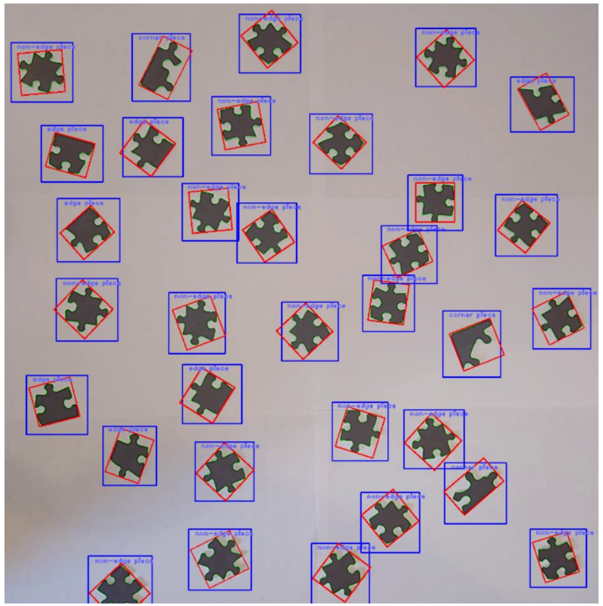
- Overall Success
  - Achieved high accuracy in classification
- Implications
  - Aid those with visual impairments
  - Decrease barrier of entry





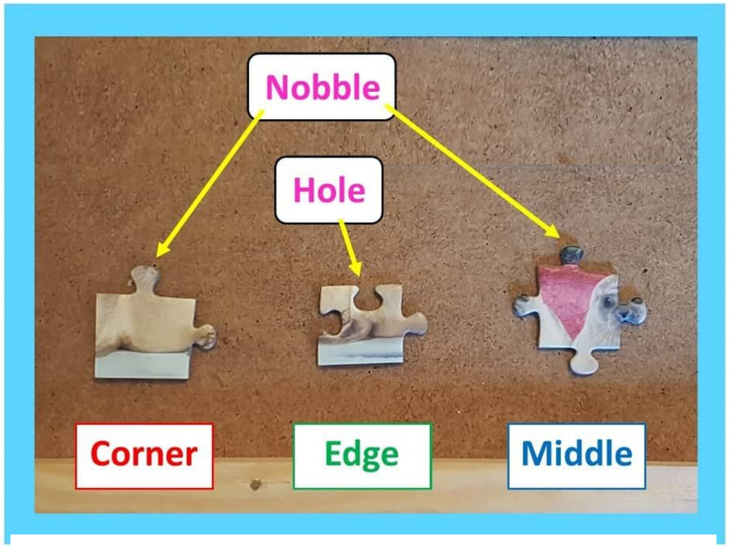
Evaluations

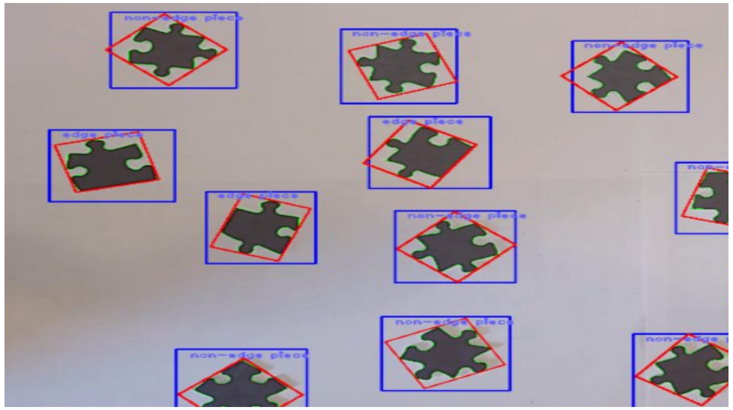
- What types of pieces do we test?
  - Test on standard and irregular shaped pieces
- How do we test?
  - Requires a stable video feed
  - Randomly sample 20-30 pieces
    - Swap pieces
  - Orientations/angles
- Main Takeaways
  - Shapes & Sizes
  - Sensitivity to lighting conditions
  - Ability to identify multiple pieces at once
  - Processes constant visual input (live-feeds, recordings)



Evaluations (cont.)

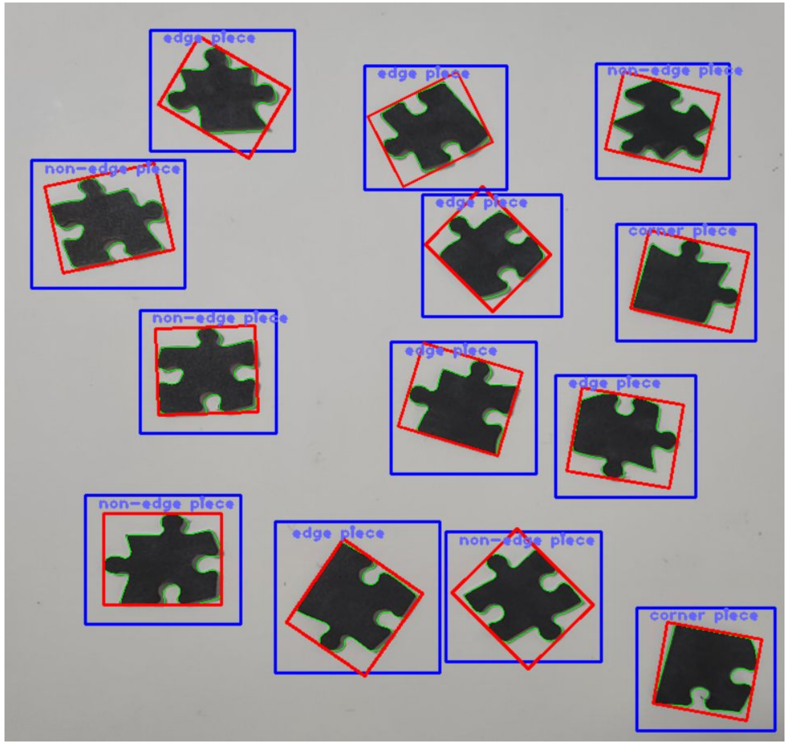
- Analyzing Our Results
  - **Main Focus:** Accuracy of identifying the correct piece type
  - Achieved an accuracy of **96%** on test dataset
- Reduced processing time
- Automated tagging & labeling process





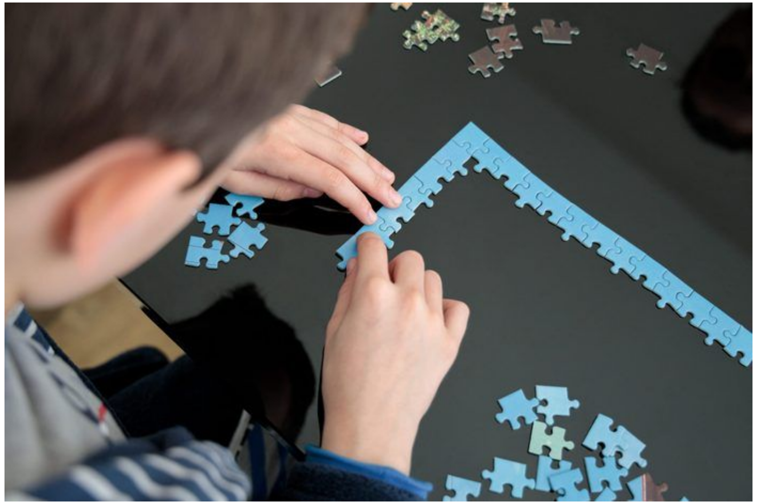
Comparisons (cont.)

- Comparison to state of the art:
  - Greatly reduced processing time
    - Results are quick & clear
  - Outperforms some existing methods
  - Existing methods require additional hardware or extensive setup
- Limitations
  - Machine Learning
  - Scalability
- Strengths
  - Ease of access
  - Real-time processing
  - Flexibility



Conclusion

-
- Future plans for the project
  - Website to improve ease of access
    - Currently in development!
      - Flask: running python code using html/css/java
  - Incorporate machine learning



```
<div class="container">
  <div class="carousel-caption">
    <h1>One more for good measure.</h1>
    <p>Cras justo odio, dapibus ac facilisis in, egestas eget quam.
    .</p>
    <p><a class="btn btn-lg btn-primary" href="#" role="button">
  </div>
 </div>
</div>
</div>
<a class="left carousel-control" href="#myCarousel" role="button" data-slide="prev">
  <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
  <span class="sr-only">Previous</span>
</a>
<a class="right carousel-control" href="#myCarousel" role="button" data-slide="next">
  <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
  <span class="sr-only">Next</span>
</a>
</div><!-- /.carousel -->

<!--Featured Content Section-->
<div class="container">
  <div class="row">
    <div class="col-md-4"></div>
    <div class="col-md-4"> <h2> FEATURED CONTENT </h2>
    <div class="col-md-4"></div>
```

References

- Worlds hardest jigsaw vs. puzzle machine (all white)
- A fully automated greedy square jigsaw puzzle solver
- A probabilistic image jigsaw puzzle solver
- https://medium.com/@tomasz.kacmajor/hough-lines-transform-explained-645feda072ab