TOWARDS SOLVING VISUAL PUZZLES USING VISUAL KNOWLEDGE REPRESENTATIONS IN SOAR

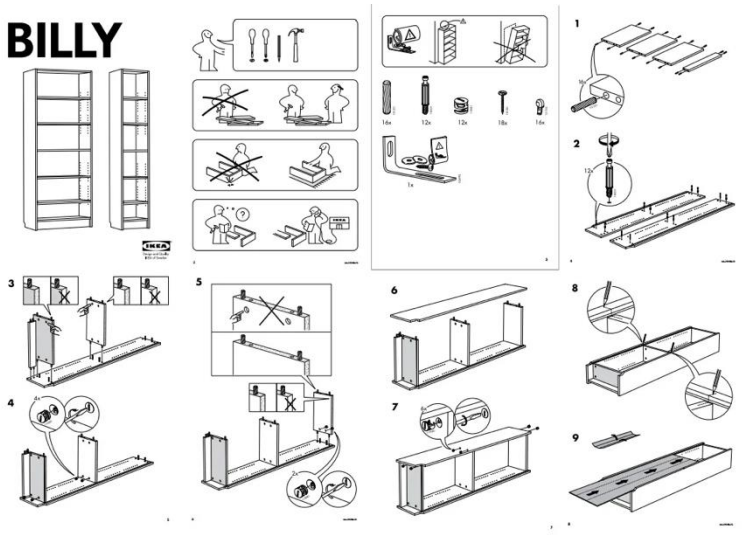
JAMES BOGGS

UNIVERSITY OF MICHIGAN

INTRODUCTION

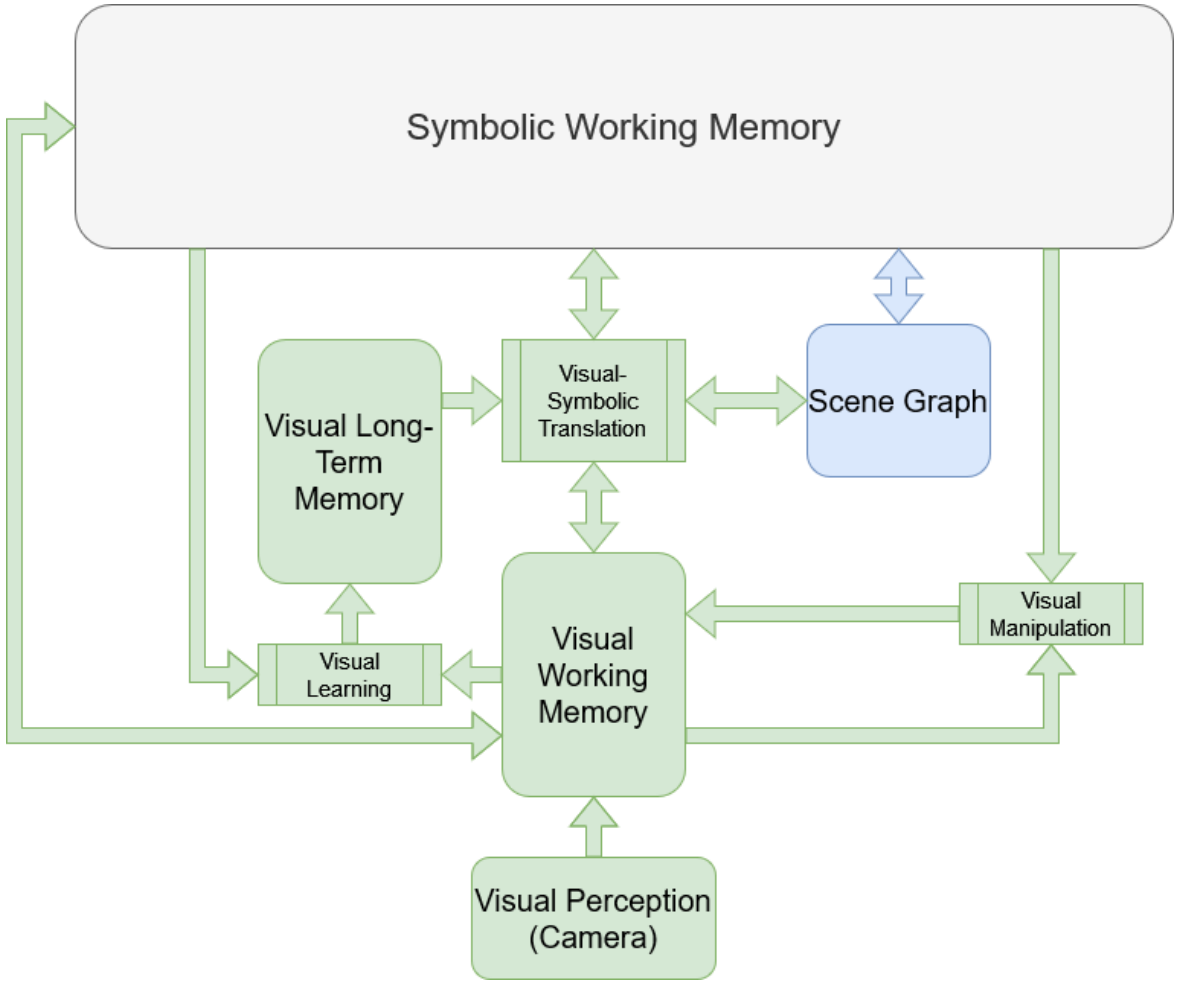
MOTIVATION

- Many tasks in the real world benefit from or even require visual reasoning



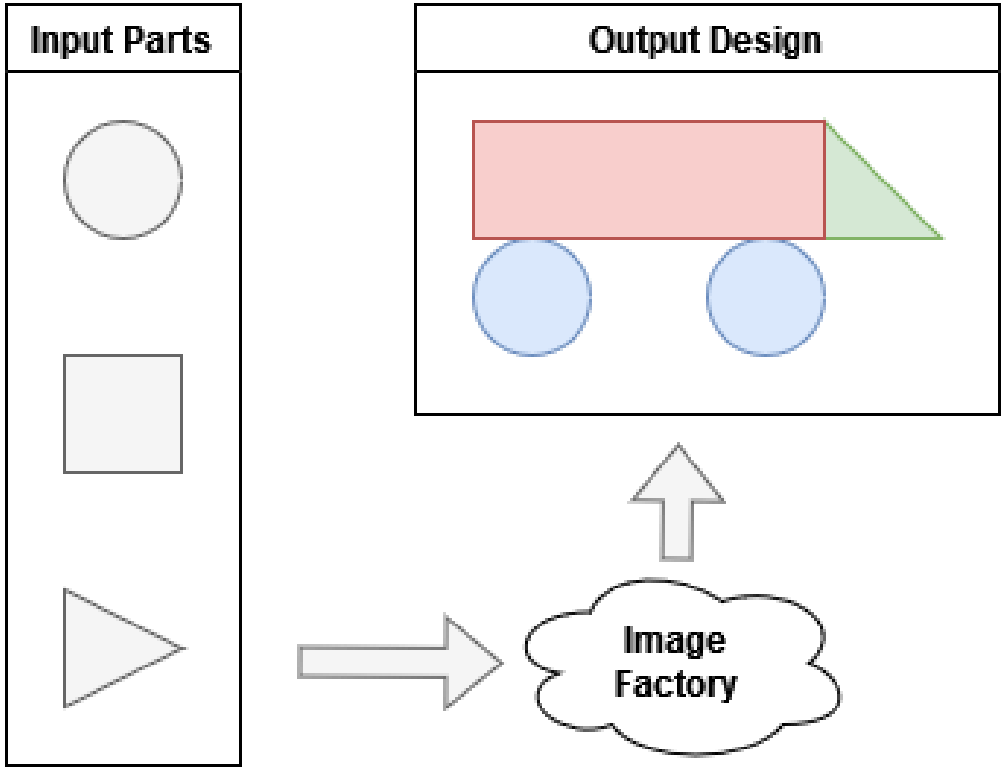
GENERAL RESEARCH FOCUS

Extend Soar to include visual knowledge and reasoning by creating new visual memories and associated reasoning operations



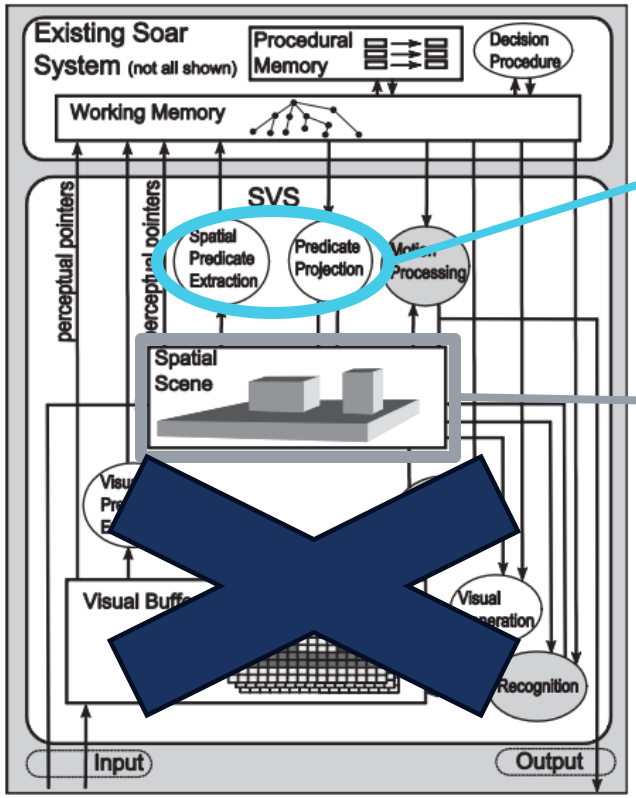
OVERVIEW OF CURRENT WORK

- Proof-of-concept agent in simple domain: "Image Factory"
  - Fundamental task: Create output design by modifying and combining input parts
- New visual-symbolic object representation in SVS

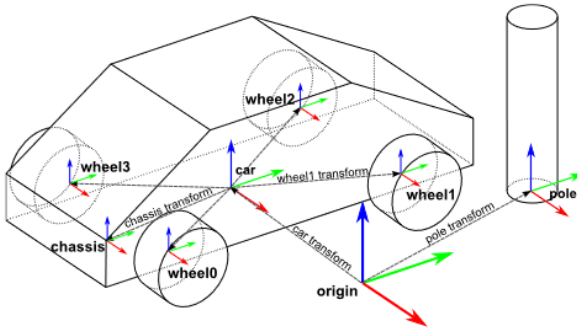


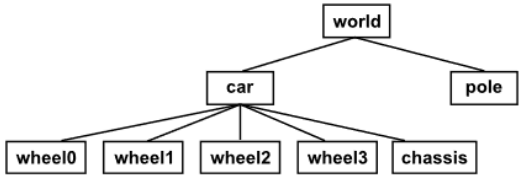
BACKGROUND

CURRENT SPATIAL-VISUAL SYSTEM




```
(S1 ^svs S3)
  (S3 ^command C3 ^spatial-scene S4)
    (C3 ^extract E2)
      (E2 ^a A1 ^b B1 ^result R7 ^status success ^type intersect)
        (A1 ^id car ^status success ^type node)
        (B1 ^id pole ^status success ^type node)
        (R7 ^record R17)
          (R17 ^params P1 ^value false)
            (P1 ^a car ^b pole)
```

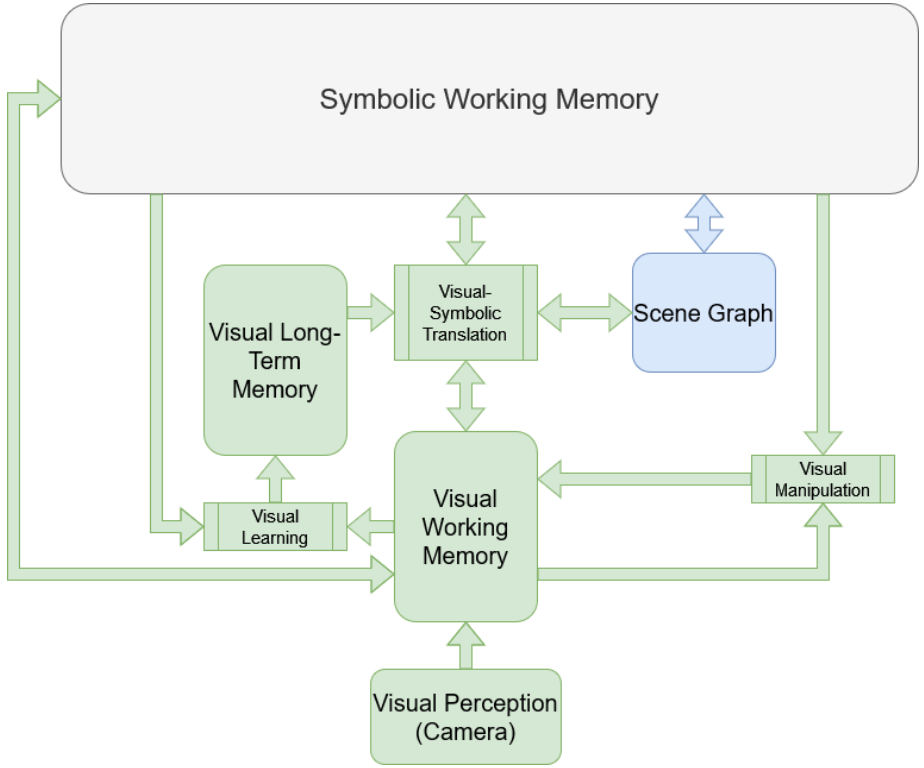




Laird 2012, "The Soar Cognitive Architecture"

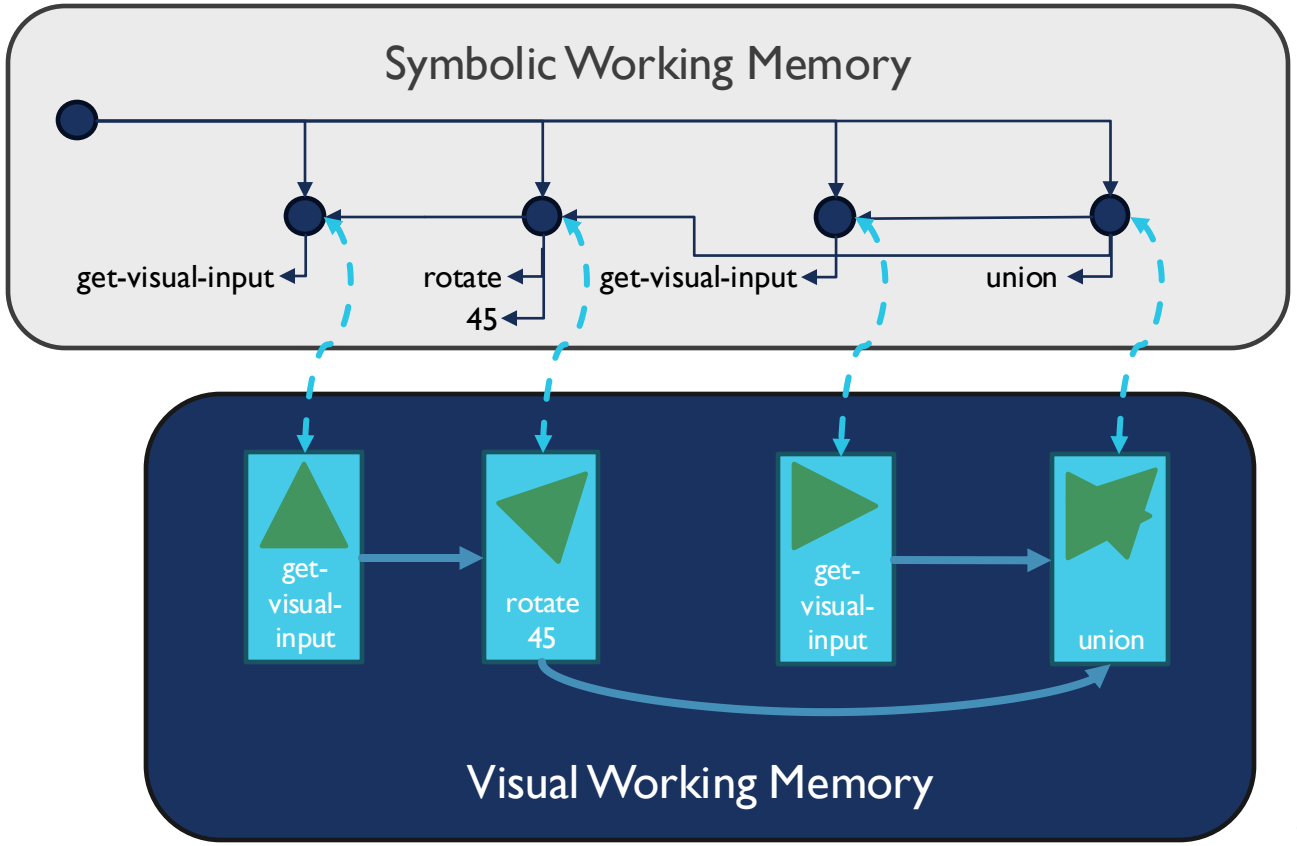
STRUCTURE OF SVS2

- Three new visual memories:
- Visual Working Memory: holds task-specific visual knowledge and reasoning operations
- Visual Long-Term Memory: holds task-independent, long-term visual knowledge
- Visual Input Buffer: very short term memory storing most recent raw visual percepts



VISUAL WORKING MEMORY

- Maintains current state of visual reasoning
- Graph of visual operations (VOps) which can generate & manipulate visual knowledge
- VOps + their inputs & outputs connected to symbolic WM



CURRENT WORK

IMAGE FACTORY DOMAIN

- Objective: given input parts, design a factory which uses machines to assemble the inputs into a specified output product
- Parts & product are both given as images
- Machines can:
  - Generate an input part
  - Scale or rotate an input part
  - Stack two input parts with a relative translation
- Solution should be given as a graph indicating how machines should be connected

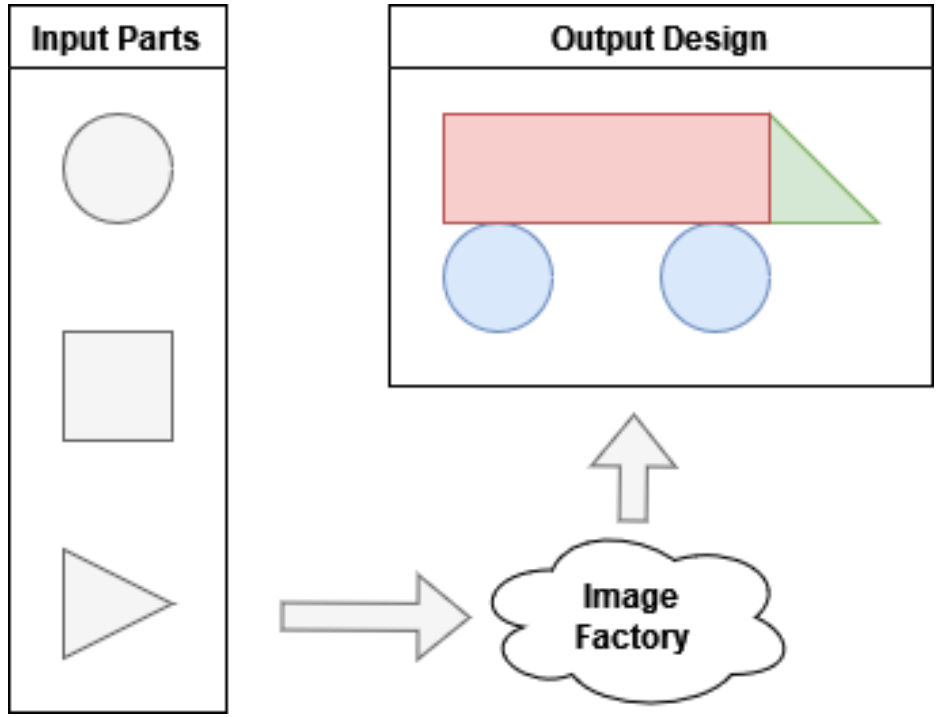
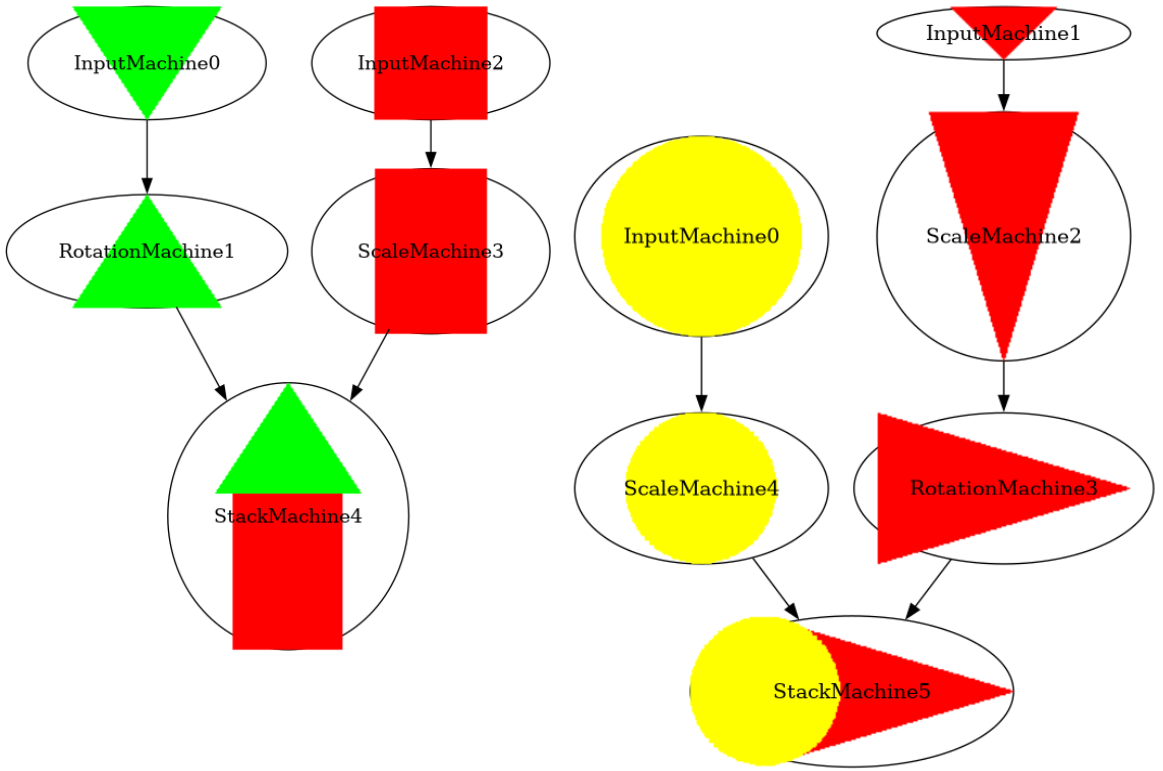


IMAGE FACTORY: SOME SIMPLIFICATIONS

Image Factory domain as described is extremely complex, so some simplifications were made for this work:

- Only two input images
  - Unique shapes and colors
- No color changing machine
- Limited factory size
- Limited values for scaling and rotation



OBJECT REPRESENTATION

- Fundamental to the domain is *objectness:*
  - there are multiple discrete components which should be treated separately
- Must be able to represent *objects* in addition to images
- Object representation is visual-symbolic: should maintain both visual-spatial & symbolicized features
- Currently, uses hand-crafted feature set, including:
  - Image, mask, height, width, location in original image, color, number of sides, angle of minimal rectangle

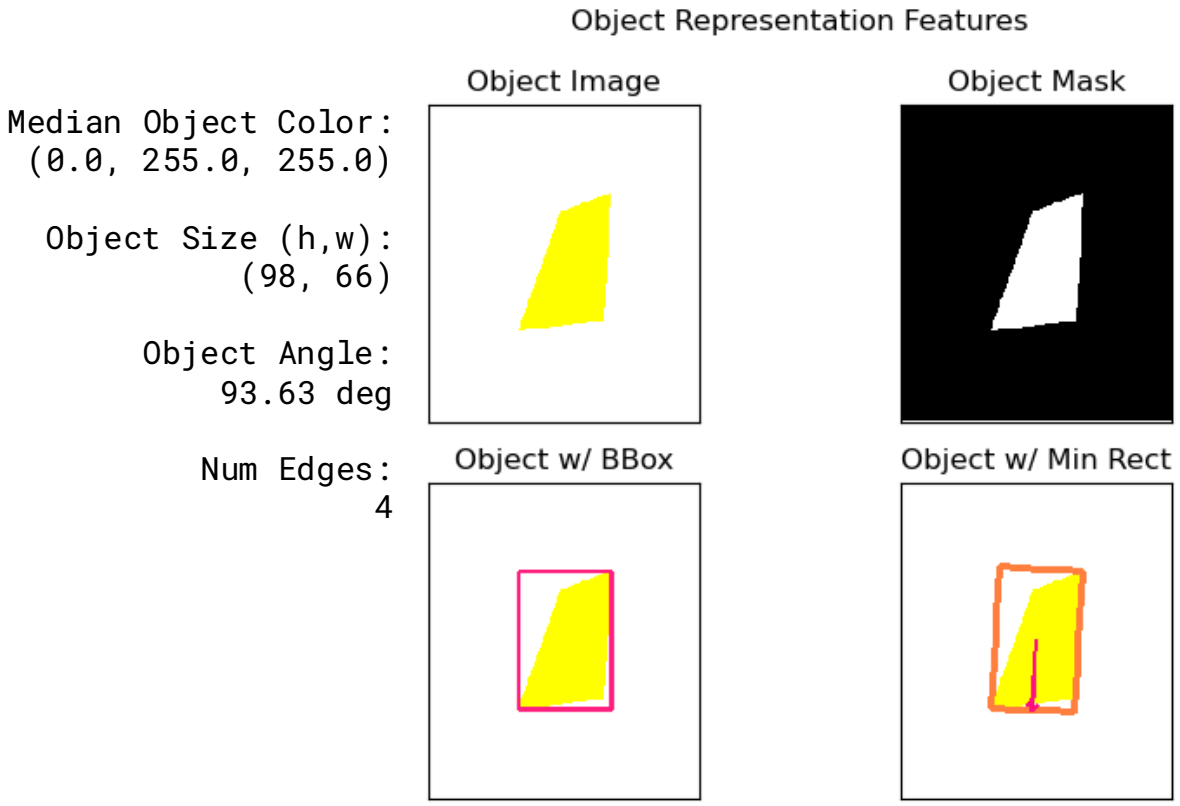
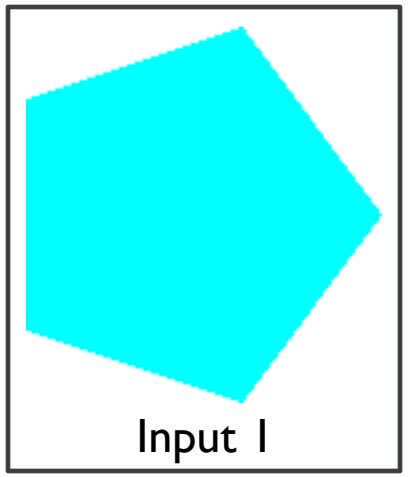
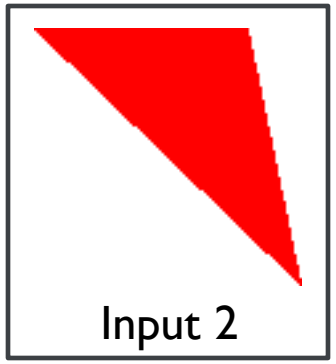


IMAGE FACTORY AGENT

- Get input parts and desired product





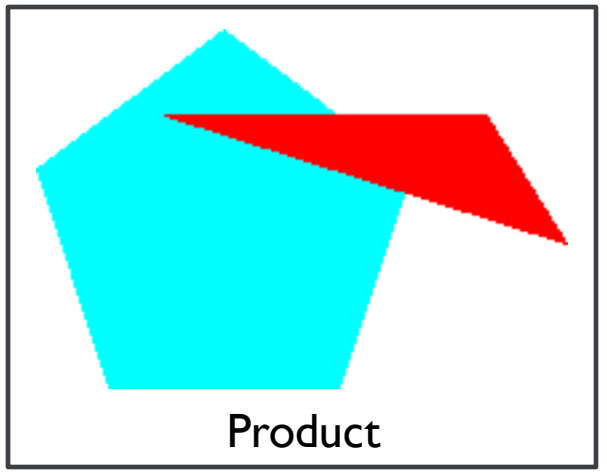


IMAGE FACTORY AGENT

- Get input parts and desired product
- Segment product image by color to get individual objects

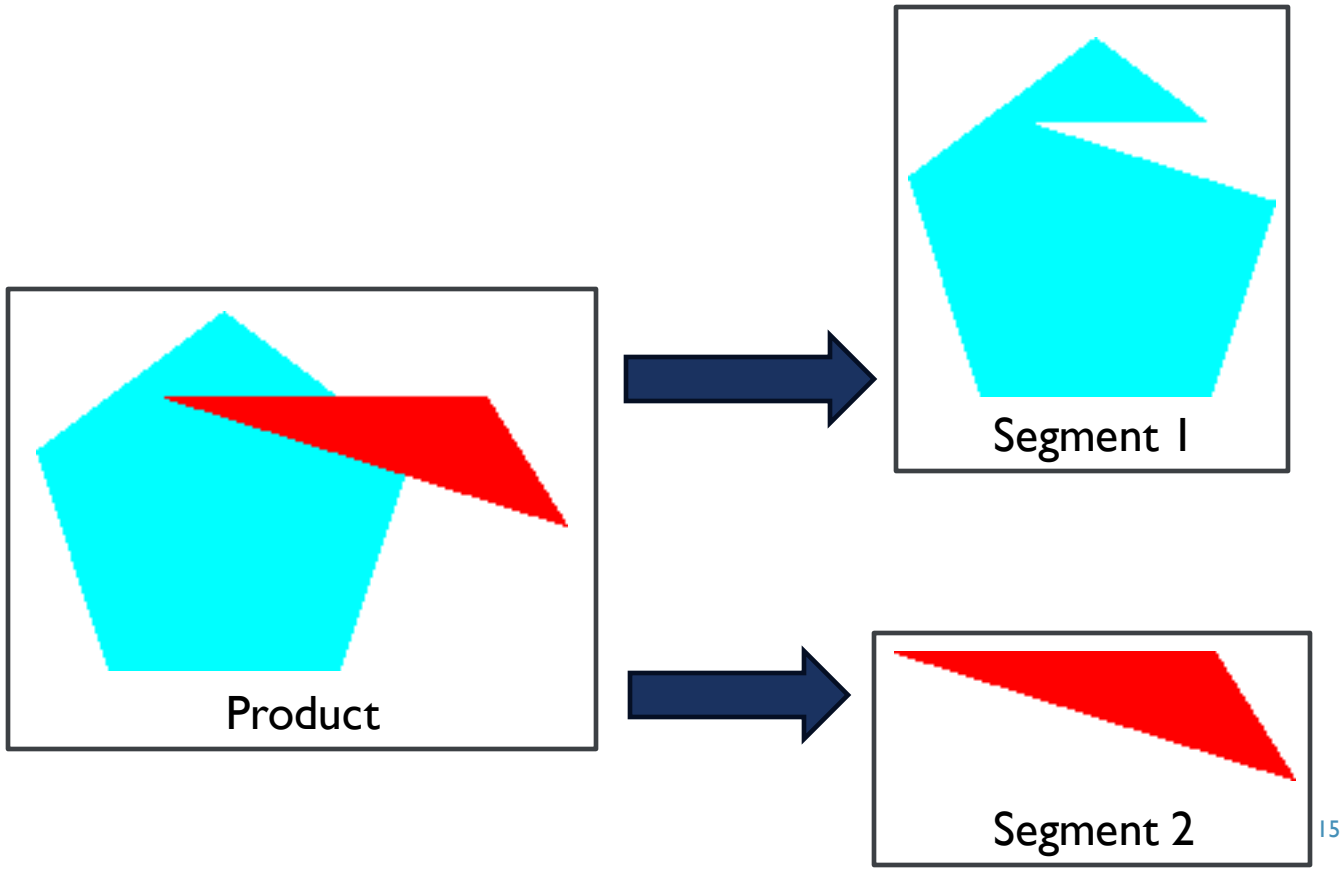


IMAGE FACTORY AGENT

- Get input parts and desired product
- Segment product image by color
- Since there's no color changing, pair same-colored inputs and segments together

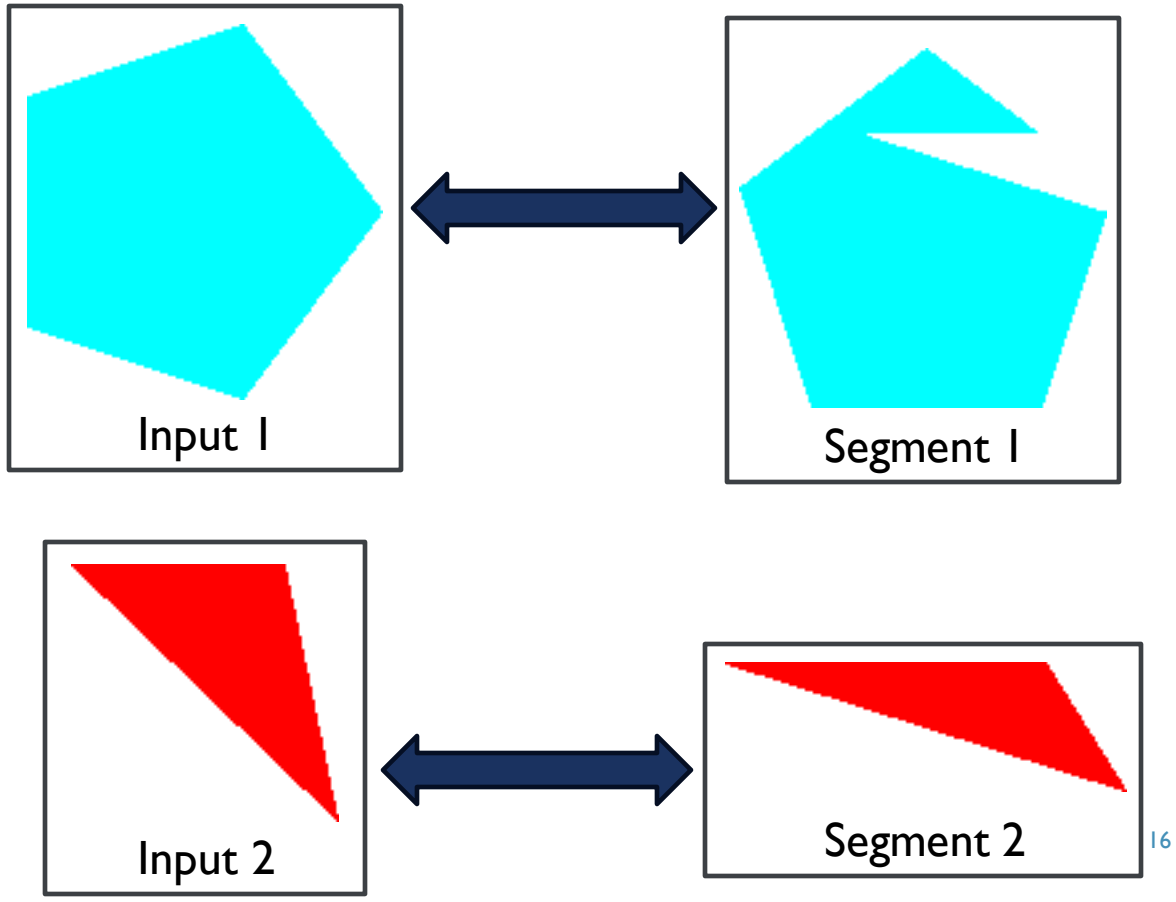
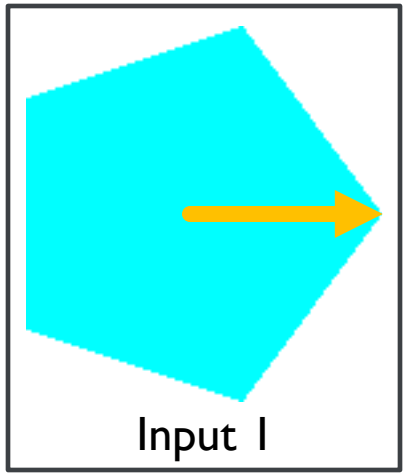


IMAGE FACTORY AGENT

- Get input parts and desired product
- Segment product image by color
- Pair inputs and segments
- Compare input and segment in a pair to get difference in angles and height & weight ratios



Input 1

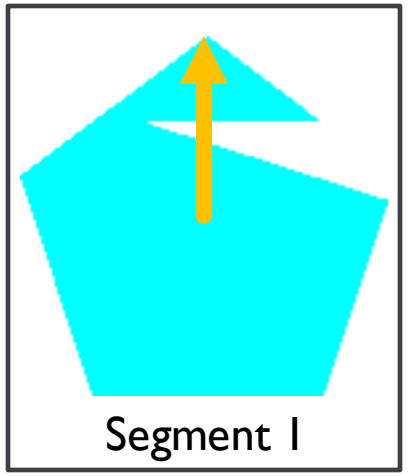
Object Angle:
180 deg

Object Size:
(141, 134)

Δ Angle:
-90 deg

H-ratio:
1.0

W-ratio:
1.0



Segment 1

Object Angle:
90 deg

Object Size:
(135, 138)

IMAGE FACTORY AGENT

- Get input parts and desired product
- Segment product image by color
- Pair inputs and segments
- Compare input and segment
- Attempt a scaling or rotation operation on input part, guided by results of comparison

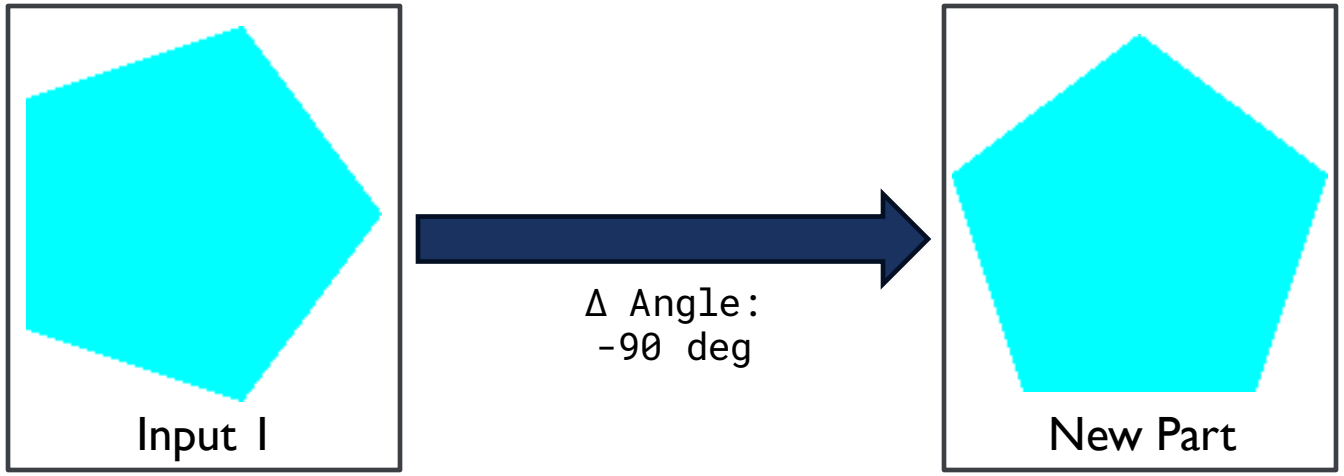


IMAGE FACTORY AGENT

- Get input parts and desired product
- Segment product image by color
- Pair inputs and segments
- Compare input and segment
- Attempt a scaling or rotation operation
- Compare results with segment and product to detect matches
  - Accounts for and detects obscuration, which it stores to remember which part goes on top in final product

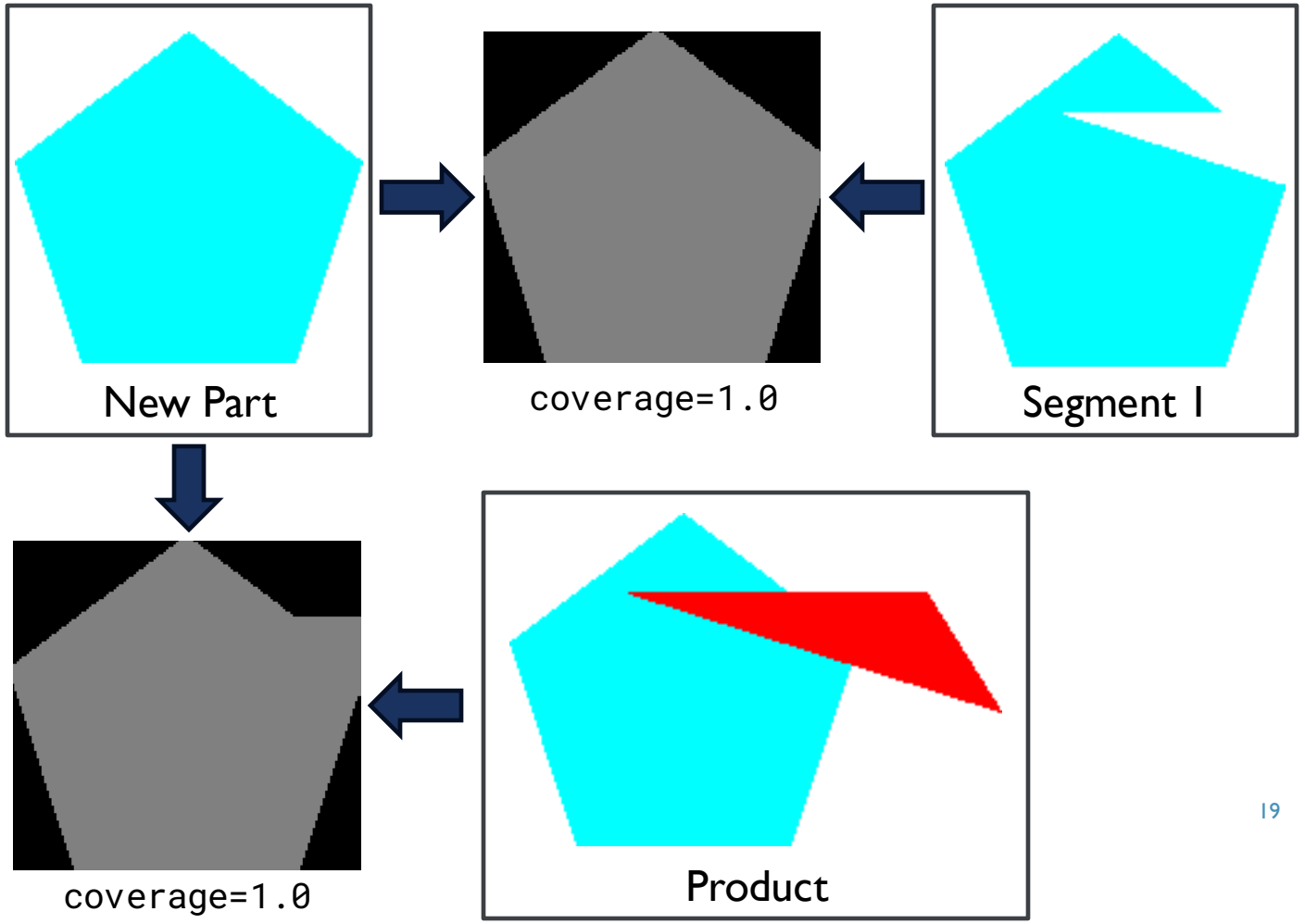
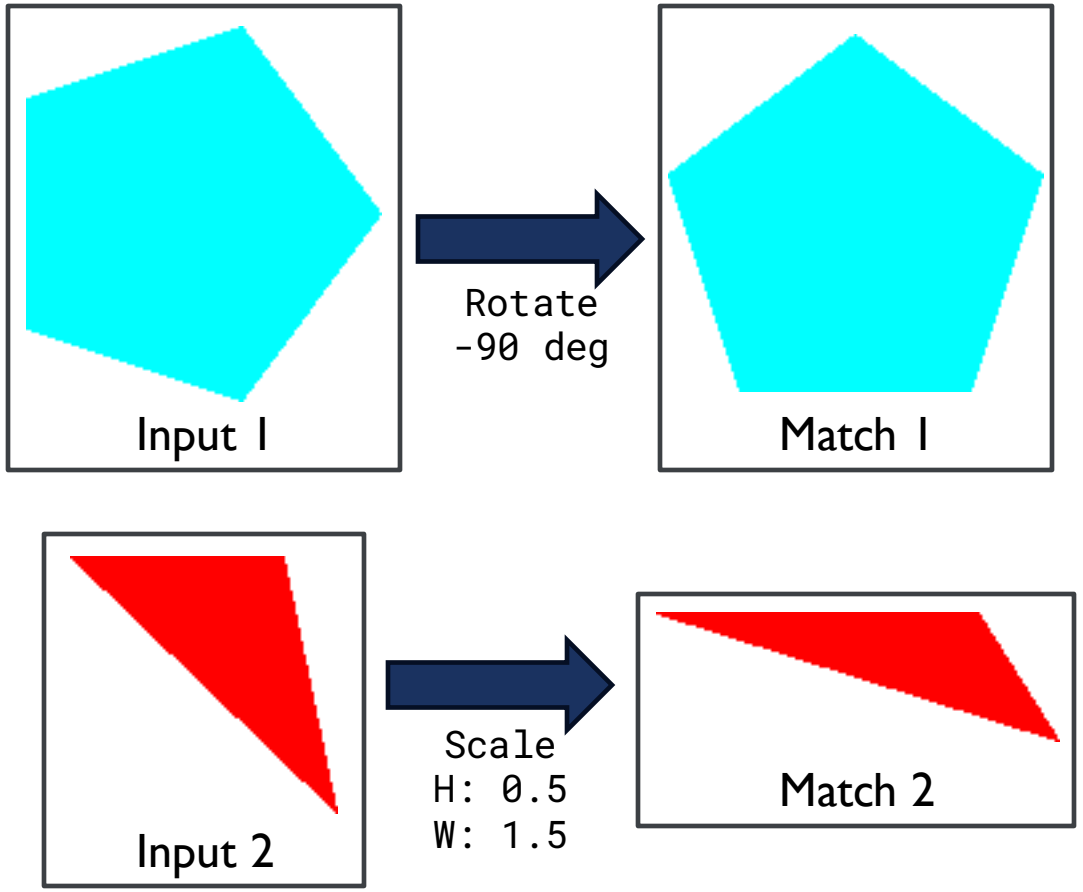


IMAGE FACTORY AGENT

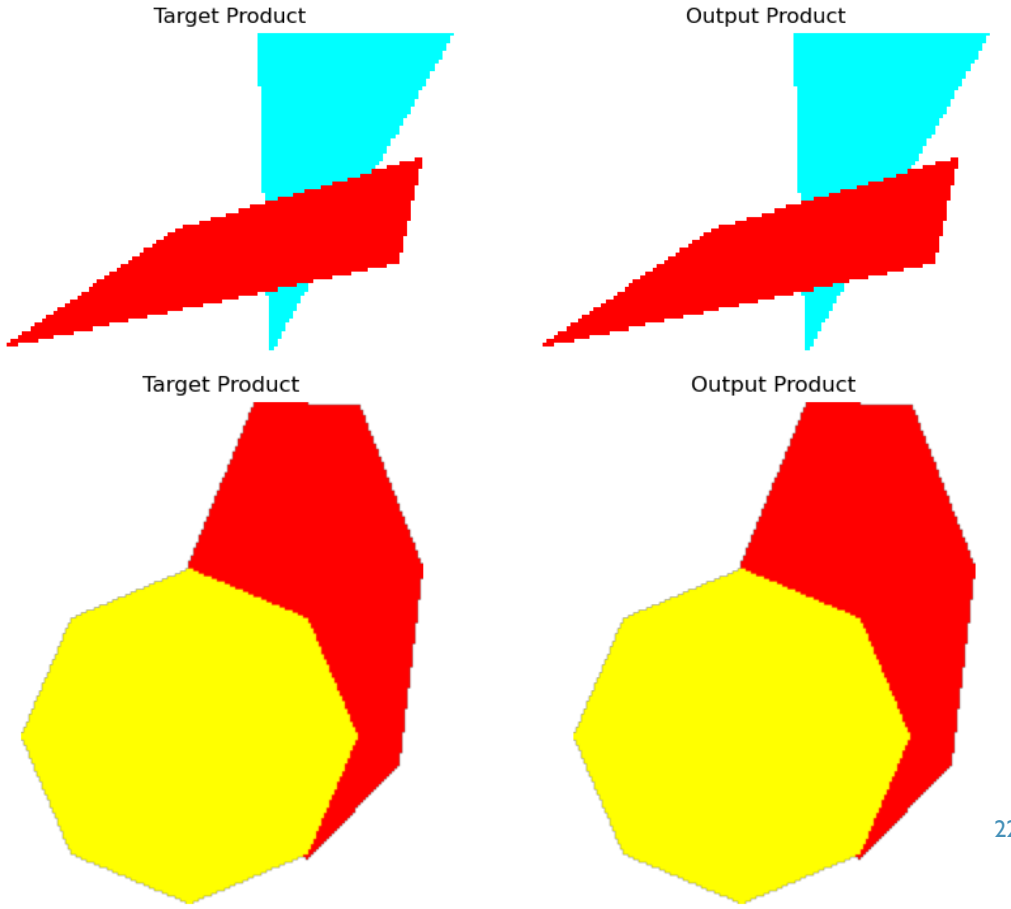
- Get input parts and desired product
- Segment product image by color
- Pair inputs and segments
- Compare input and segment
- Attempt a scaling or rotation operation
- Detect matches
- Build factory design by following chain of operations which led from an input part to a matching part



RESULTS & DISCUSSION

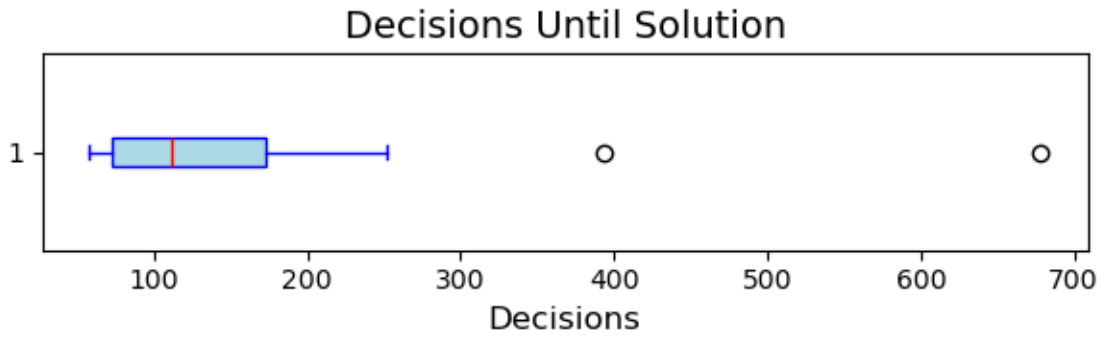
INITIAL RESULTS

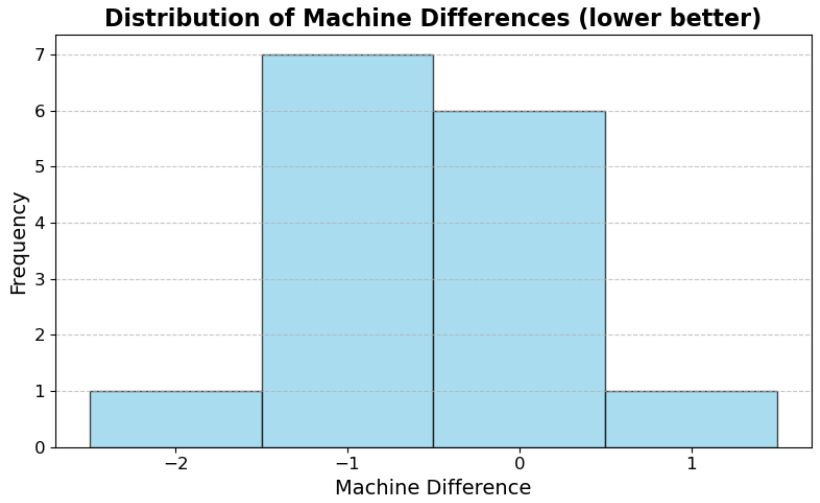
- So far, tested on 16 randomly-generated factories
- 10 agent-generated images were visually identical to goal images
- 5 were within a small margin of error (unnoticeable to humans)
- 1 factory couldn't be completed in reasonable timeframe



INITIAL RESULTS

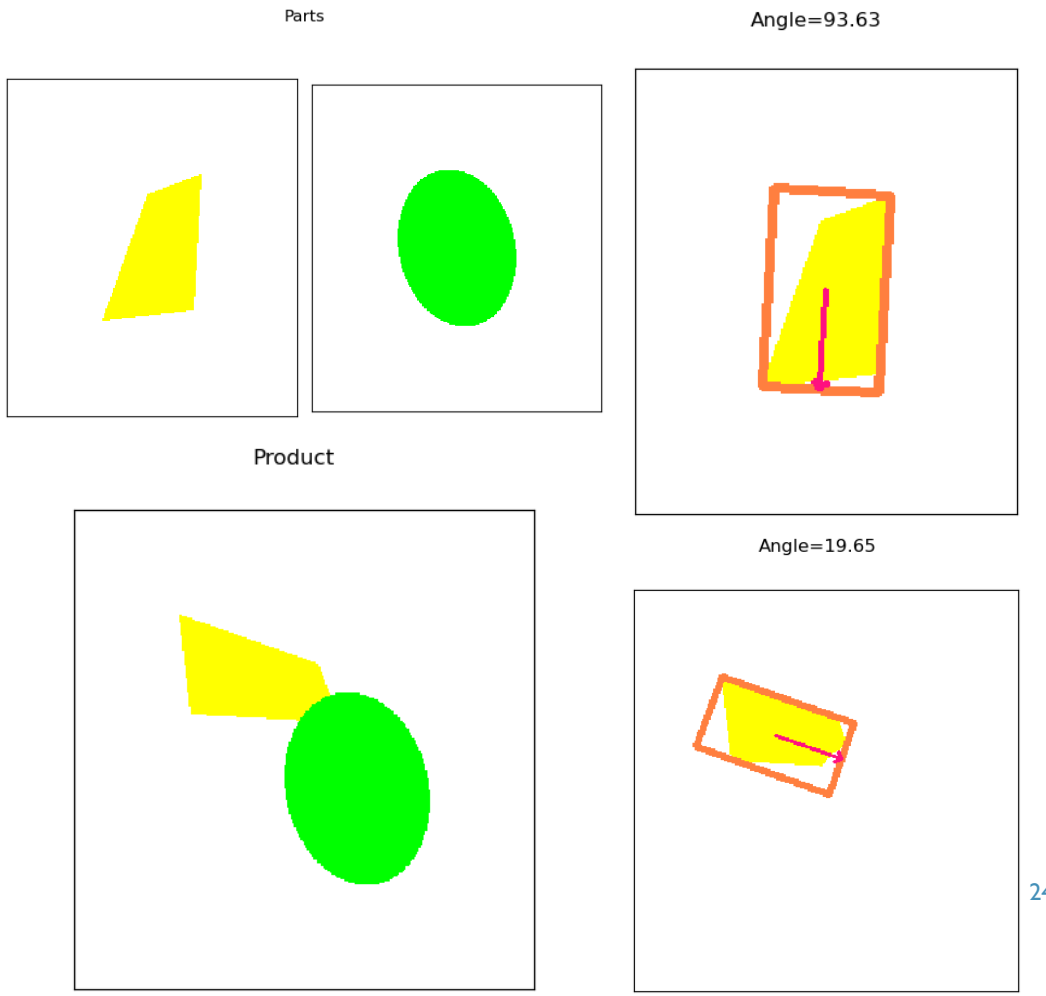
- 15/16 factories were completed in a reasonable amount of time
  - Min decisions: 58
  - Max decisions: 678
  - Longest wall time was 35s
  - Incomplete factory took >2000 decisions
- Agent solutions were almost entirely (14/15) as good or better than original factory
  - Only 1 added an unnecessary machine





ANALYSIS: FAILURE CASE

- Incomplete factory took >2000 decisions
  - Ended up crashing before finding solution (too much memory usage?)
- Possible reason: factory has an obscured, irregular polygon
- Problem: search is guided by:
  - bounding box height & width ratios for scaling
  - minimal rectangle angle for rotation
- Bounding box ratio thrown off by obscuration
- Angle thrown off by obscuration + irregularity
  - OpenCV minimal rectangles are finicky!
- Result: resorts to exhaustive search



FUTURE WORK & CONCLUSION

FUTURE WORK: TWO DIRECTIONS

Better Soar Code

- Significant room for optimization to prevent slow-down and crashing
- Improve search guidance rules by:
  - adjusting heuristic weights
  - baking in more geometric knowledge
  - probably lots of other little tricks

Improve Object Representations

- ~~Add more information to hand-crafted representation~~
  - Too painstaking and too fragile!
- Learn object representations via deep learning
  - regress affine transform between input and product part representations
  - predict complete product part from obscured segment

NUGGETS AND COAL

Nuggets

- Mostly successful proof-of-concept for visual reasoning with SVS2
- Extends Soar to include images and image manipulation, might be helpful to some
- Hopefully enough to finish my dissertation!?

Coal

- Hand-crafted object representations are too fragile
- Fragility bleeds over into Soar code: no way to handle noisiness/fuzziness
  - e.g., slight pixel misalignments can throw everything off
- Bonus coal: using PyTorch in Soar is tricky due to namespace collision