Automatically Grading Rey-Osterrieth Complex Figure Tests using Sketch Recognition

ANONYMOUS AUTHOR(S)

The Rey-Osterrieth Complex Figure Test (ROCF) is among the most widely used neuropsychological examinations to analyze visual spatial constructional ability and memory skills, but grading the patient's sketched complex figure is subjective in nature and can be time consuming. With increasing demand for tools to help detect cognitive decline, there is a need to leverage sketch recognition research to assist in detecting fine details within an ROCF's inherently abstract figure. We present a series of recognition algorithms to detect all 18 official ROCF details using a top-down sub-shape recognition approach. This automated grader transforms a sketch into an undirected graph, identifies and isolates detail sub-shapes, and validates sub-shape neatness via a point-density matrix template matcher. Experimental results from hand-drawn ROCFs confirm that our approach can automatically grade ROCF Tests on the same 18-item sketch detail checklist used by neuropsychologists with marginal error margin.

CCS Concepts: • **Applied computing → Health care information systems**; • **Human-centered computing → Gestural input**; Mobile devices.

Additional Key Words and Phrases: digital neuropsychological tests, sketch recognition, digital smart-pens

ACM Reference Format:
Anonymous Author(s). 2021. Automatically Grading Rey-Osterrieth Complex Figure Tests using Sketch Recognition. In *Proceedings of CHI '21: ACM CHI Conference on Human Factors in Computing Systems (CHI '21).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

1.1 Rey-Osterrieth Complex Figures

The Rey-Osterrieth Complex Figure Test (ROCF), developed by Rey [18] in 1941 and refined by Osterrieth [15] in 1944, is a neuropsychological test that evaluates several cognitive functions including visuospatial abilities, memory, attention, planning, working memory and executive functions [10, 23]. The ROCF is characterized as a complex cognitive task [22], and is known in the field of neuropsychology as a useful metric for the frontal lobe function [21]. A participant is asked to copy the figure into a piece of paper, then copy it again two more times from memory. The shape is specifically designed to be abstract so that participants cannot associate it with any common object or concept. A clinician then grades all three sketches on whether 18 separate sub-shapes (henceforth called "details") exist and, if they do, how neatly they were drawn. A clinician grants up to 2 points for each detail that totals to 36 points, with partial credit given to distorted or misplaced shapes. Points for overall neatness of individual details is subjective and is generally dependant on an expert's intuition, especially for shapes that exist but might be drawn poorly. This results in different ROCF graders potentially producing two different scores. The proliferation of digital

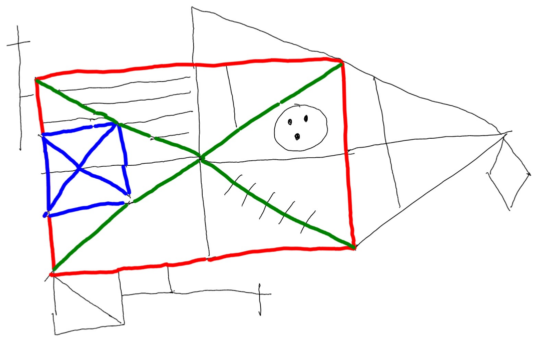


Fig. 1. Our automated grader highlighting Details 2, 3, and 6 in red, green, and blue respectively.

sketch recognition techniques and a push to digitize clinical neuropsychological examinations motivated our creation of an automated ROCF that can grade itself on the existing grading scheme.

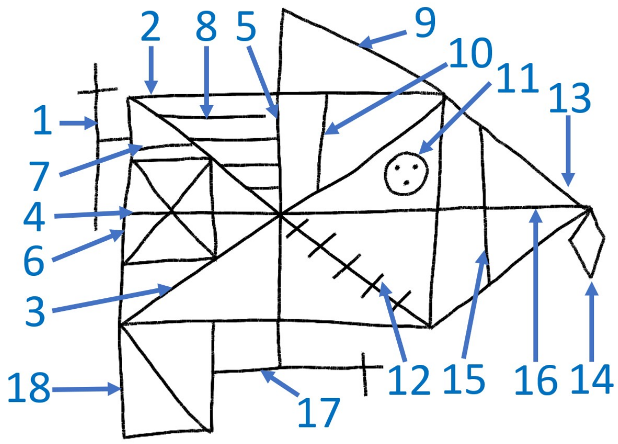
From a digital sketch recognition standpoint, automatically grading an ROCF is non-trivial due to the complexity of the figure and test conditions resulting in inherently fuzzy sketch data. No two completed sketches are drawn in the same order, and very frequently shapes are drawn using portions from other shapes [4]. Bottom-up approaches tend to classify shapes as soon as their constraints are met, but shapes in an ROCF may in fact be only part of a detail or may end up as a portion of an entirely different one. A top-down approach not only more closely resembles a human grading an ROCF, but it also simplifies the recognition process by not needing to re-classify a shape at every step of the hierarchical recognition process.

1.2 Contribution

Our top-down sub-shape recognizer is an expansion on existing techniques used to identify hand-drawn trusses [6]. This technique transforms a drawn shape into a graph where every line intersection, corner, and line endpoint is expressed as a vertex and the lines connecting them expressed as edges. The truss is recognized by identifying the resulting graph's triangles.

Whereas previous efforts in automatically grading the ROCF can identify only a subset of the complex figure's details, we present the first fully automated ROCF grader that does not require user input to point to baseline shapes from which to begin recognition. Our contribution widely expands on Field's truss recognition technique [6] by introducing several lightweight graph traversal algorithms in order to isolate specific sub-shapes or regions from a given sketch. In addition to triangles, we also recognize squares, parallel lines, crosses, straight horizontal and vertical lines, and diamonds as well as shapes specific to the ROCF such as detail 6 (Cross with Square), detail 14 (Circle and 3 Dots), and 18 (Square with Line). Many of our recognition algorithms utilize well-known graph traversal and optimization algorithms (such as Dijkstra's Shortest Path [5] and Depth-First Search [24]).

To test our recognizer's performance, the system graded 141 digitized Rey-Osterrieth tests from participants, and we compare how closely our system's grades correlate with those of an expert grader. The experimental results demonstrates the proposed approach is successful in identifying the existence of sub-shapes within a large abstract shape.



1: Cross
2: Large Rectangle
3: Large X Across 2
4: Horizontal Line
5: Vertical Line
6: Square+Cross
7: Small Line
8: Parallel Lines
9: Triangle
10: Line
11: Circle+3 Dots
12: Parallel Lines
13: Triangle on Right
14: Diamond
15: Line
16: Line
17: Cross
18: Square+Line

Fig. 2. A Rey-Osterrieth Complex Figure Test, with all 18 Details listed.

2 RELATED WORK

Digital sketch recognition techniques favor bottom-up approaches that employ computational geometry to classify shapes [2, 9, 19]. Hierarchical sketch recognition systems such as LADDER [8], Sketchread [1], Chemink [16] and Mechanix [6, 6] generate composite figures by re-classifying shapes into more complex shapes in every step of the sketching process. Corner detection also helps characterize digital shapes, with lightweight systems such as ShortStraw [28] and iStraw [29] being among the most efficient.

The “Dollar” family of recognition systems [27] remains among the most well known single and multi-stroke gesture classification algorithms. While most techniques rely on stroke order, geometric properties, and physical characteristics such as speed, acceleration, etc., the “$P+” recognizer calculates similarity via point cloud approximation [25]. This is especially flexible when the application in question necessitates recognition that is agnostic to stroke order.

Efforts to automate other neuropsychological tests has renewed interest in sketch sub-object detection [13, 14]. However, whereas recognized objects for these tests tend to have heavily distinct characteristics, ROCF details are mostly composed of simple primitives that appear frequently. For example, detail 5 shown in Figure 2 is defined not only as any vertical line, but rather a specific vertical line within the sketch. Work presented by Prange *et al.* [17] cites Rey-Osterrieth figures as a motivating factor in the need to identify geometric shapes inside complex abstract figures. Existing attempts to automatically grade ROCFs are semi-automated or do not implement detection of all 18 details[3, 4]. The most recent attempt automates grading using a deep-learning neural network but leaves ample room for improvement of individual segment detection, most notably single-line details [26].

Hierarchical sketch recognition approaches generally check drawn lines to see if they meet requirements for a composite shape [11, 12]. However, this creates a need to check for every possible shape component at every step of the sketching process. This becomes prohibitive for large, complex line drawings such as the Rey-Osterrieth complex figure, especially if there is no guarantee that every line is present. Although important foundational work has been established, no one system has presented fully automated grading of ROCFs that also identifies absent and distorted details.

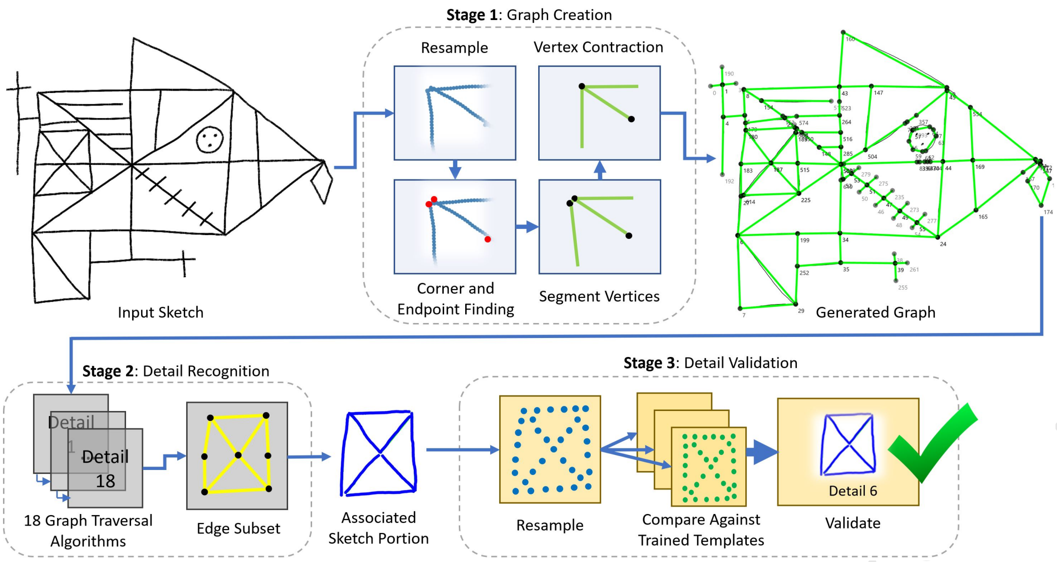

Stage 1: Graph Creation
Resample
Vertex Contraction
Corner and Endpoint Finding
Segment Vertices
Input Sketch
Generated Graph
Stage 2: Detail Recognition
Detail 1
Detail 18
18 Graph Traversal Algorithms
Edge Subset
Associated Sketch Portion
Stage 3: Detail Validation
Resample
Compare Against Trained Templates
Detail 6
Validate

Fig. 3. Description of ROCF sub-shape recognition system. Stages 2 and 3 shown in the figure are repeated for each of the 18 details of a Rey-Osterrieth complex figure.

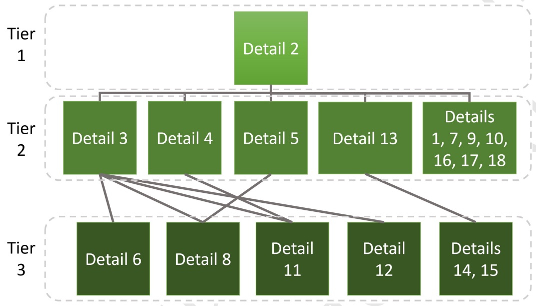


Fig. 4. Auto ReyO's recognition hierarchy, designed to have as few dependencies as possible.

3 AUTOMATED REY-OSTERRIETH COMPLEX FIGURE TEST GRADER (AUTO REY-O)

Our top-down sub-shape recognizer divides the ROCF grading process into three distinct stages as shown in Figure 3.

3.1 Stage 1: Graph Creation

The graph creation stage is divided into four distinct steps. First, we prepare the sketch for corner detection by resampling to a uniform interspace length S as follows:

$$S = \frac{\sqrt{(x_m - x_n)^2 + (y_m - y_n)^2}}{c} \tag{1}$$

where (x_m, y_m) is the lower-right corner of the sketch, (x_n, y_n) is the upper-left corner of the sketch, and c is a constant $c = 40$.

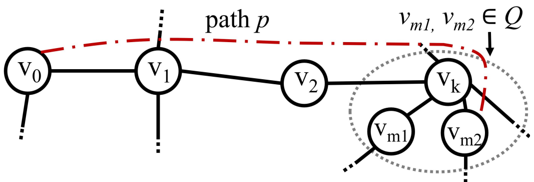


Fig. 5. Finding path p for the top horizontal side of detail 2's rectangle. Dotted area on right indicates *dist* radius. In this example $v_{m2} = c_1$, $dir = Right$ and $nextdir = Down$ (See Algorithms 1 and 2)

The second step utilizes the corner-finding algorithm from Wolin [28] to identify any "corner" from drawn strokes. To detect line intersections, two straight-line segments are compared with the target segment $y_a = a_2x + a_1$ checked for intersection against comparison segment $y_b = b_2x + b_1$ with equation 2.

$$\frac{a_1 + b_1}{a_2 + b_2} \in \left(x_1 - \frac{(0.15l)^2}{1 + a_2^2}, x_n + \frac{(0.15l)^2}{1 + a_2^2} \right) \tag{2}$$

where x_1 and x_n represent the x values of the less and greater vertices of the target segment respectively with l being its segment length.

The third step creates undirected graph G, where every vertex v is a line endpoint, corner, or intersection, and every edge e is a drawn line connecting each v. Each v contains a point from the sketch, and each e contains the sampled points that connect the two vertices.

The fourth and final step performs vertex contraction on the created graph. Each vertex is iterated over and checked for near vertices that fall below a predetermined distance threshold. If two vertices are joined, their respective sampled points s_i including points from an edge that might fall between them are combined into a single vertex v containing all the sampled points. The distance measure is determined through complete distance used in hierarchical clustering, taking the maximum of the set in equation 3.

$$\{\ ||s_i - s_j||_2 \mid s_i \in v_1, s_j \in v_2\ \} \tag{3}$$

This serves both to connect segmented or near vertices and to reduce the overall complexity of the graph by eliminating edges. Finally, the vertices are iterated over a second time, checking if near vertices fall below a distance threshold, where the distance is determined through taking the minimum of the set in equation 3 referred to as single distance used in hierarchical clustering. If the distance falls below a predetermined threshold, the points are then are linked through an edge.

Algorithm 1 Detail 2: Largest Rectangle

Input: Sketch Graph's vertex adjacency list
Output: Largest rectangle vertices

```
1:  for all node $n$ in Graph do
2:      corner $c_1$=SideAndCorner($n$,$Right$,$Down$)
3:      corner $c_2$=SideAndCorner($c_1$,$Down$,$Left$)
4:      corner $c_3$=SideAndCorner($c_2$,$Left$,$Up$)
5:      corner $c_4$=SideAndCorner($c_3$,$Up$,$Right$)
6:      if $n = c4$ then
7:          add all SideAndCorner $b$ sides to rectangle $q$
8:      end if
9:  end for
10: return largest $q$
```

3.2 Stage 2: Detail Recognition

All 18 ROCF details are recognized by applying a graph traversal algorithm to identify a "shape" within the graph. Each detail has an associated algorithm that is called in the hierarchical order defined by Figure 4. Every algorithm is designed to accommodate inherent graph imperfections from both the graph creation stage and the participant's hand sketch. For example, if the algorithm checks for a horizontal edge, there we allow a slope between 0.3 and -0.3 since participants are not expected to produce a perfectly horizontal line. The algorithms were designed to strike a balance between leniency to accommodate the imperfect nature of a hand-drawn shape and precision to find the expected shape if it exists. We are unable to thoroughly explain every detail's graph traversal algorithm, but we have chosen to explain detail 2 as Algorithms 2 and 1 since it is the most sophisticated of our recognizers and best illustrates our graph traversal approach.

3.2.1 Recognizing Detail 2. detail 2's recognition algorithm defined in Algorithm 1 and 2 is a greedy graph traversal algorithm tasked with finding the large rectangle that serves as the anchor for all other shapes in the graph. Our pathfinder finds one rectangle side b and the corner at its end c at a time. For each side dir is the intended path direction, and $dirnext$ is the next path direction once we find our corner.

We define N as single agents where every agent $n \in N$ has a start location $s_n \in G$ and a goal location $g_n \in G$. The path p of n consists of one side b of our rectangle where $g_n = c$. Path p is of length k that is a sequence of vertices $p = \{v_0, v_1, v_2, ..., v_k\}$ such that each consecutive vertex is either in a defined direction dir (up, down, left, right, or diagonals) or Eucledian distance $r < 25$. At the end of our sequence v_k one of two **conditions** is true:

(1) v_k is connected to a vertex v_x such that direction of $(v_k, v_x) = dirnext$
(2) v_k is connected to *other* vertices v_m such that r of $(vk, vx) < 25$.

The conditions describe that we have either (1) found a corner characterized by the start of the next side of the rectangle, or (2) the end of our sequence consists of various vertices very close together. This creates a set of "dead-end" nodes Q. For condition 1, $v_k \in Q$ and $c = v_k$. For any v_m that satisifes condition 2, $\{v_{m1}, v_{m2}, ..., v_{mn}\} \in Q$. We check all vertices in Q and return the vertex v_e that satisfies condition 1. The final sequence is $p = \{v_0, v_1, v_2, ..., v_e\}$, and $c = v_e$. This is repeated four times to find the four sides of our rectangle, and return the largest such rectangle as detail 2.

3.2.2 Examples of Other Details. The rest of our graph traversal algorithms can be divided into two distinct categories: graph-crawling algorithms that identify shapes from the graph itself, and algorithms that use vertices as boundaries and then isolates all pen strokes within a specified region.

Algorithm 2 SideAndCorner

```
Input: Start node $n$, directions $dir$, $dirnext$
Output: Side $d$ of rectangle, corner $c$
 1: push $n$ to stack $s$
 2: while $s$ not empty do
 3:     pop $s$ to $p$, mark as visited
 4:     for all adjacent pairs ($p_1$,$p_2$) of $p$ do
 5:         if direction of $(p_1, p_2) = dir$ or distance
            $e\ (p_1,p_2)<25$ then
 6:             push $p_2$ to $s$
 7:             $p=p_2$, repeat from line 5
 8:         end if
 9:         if dead end $p_n$ reached then
10:             add shortest path as a stack from $p$ to
                $p_n$ to set $c$
11:         end if
12:     end for
13: end while
14: for all current longest path $a$ in $c$ do
15:     for all adjacency pairs ($p_{a1}$,$p_{a2}$) of leaf $p_{an}$
        in $a$ do
16:         if $p_{a2}$ is $dirnext$ of $p_{a1}$ then
17:             return $c = p_{a1}$, $d = a$
18:         else
19:             pop $p_{an}$ from $a$
20:             repeat from line 15
21:         end if
22:     end for
23: end for
```

The former category is best for detail 2 as described previously, as well as simple shapes and lines like details 3, 4, 5, 7, 9, 10, 15, 16. The latter category is appropriate in cases when our graph generator may create highly variable graphs from imperfectly-drawn shapes, making it difficult for us to determine what the graph may look like. This is the case for details 1, 6, 11, 12, 14, and 17. In these instances we identify specific regions where we expect the detail to exist, and save all edges that are found. We isolate specific regions and run a bounded Depth-First-Search algorithm that returns all edges and vertices within the given region.

For detail 6, for example, our "region" is defined by the area inside the detail 2 rectangle and detail 3 cross, and we do not include the detail 4 horizontal line or detail 7 Small Segment in our DFS search. This returns the remaining subset of vertices and edges as seen in bottom portion of Figure 3. Table 1 describes the general method of recognition we applied to detect all 18 details in an ROCF, and the order of recognition is tiered as shown in Fig. 4.

Det.	Method of Recognition	Det.	Method of Recognition
1	Isolate region, then DFS to fill	10	Greedy single-direction pathfinding
2	Greedy pathfinding, repeated per side	11	Isolate triangle, then DFS to fill
3	Dijkstra's between diagonal corners of #2	12	Isolate lower-right region, DFS
4	Greedy single-direction pathfinding	13	Find upward, downward diagonals
5	Greedy single-direction pathfinding	14	DFS to find all edges on tip of 13
6	Direction path for top/bottom, Dijkstra's	15	Greedy single-direction pathfinding
7	Greedy single-direction pathfinding	16	Greedy single-direction pathfinding
8	Connect horizontal lines bet. #3 and #5	17	Isolate region, then DFS to fill
9	Find vertical, diagonal line above #2	18	#2's technique, then single diagonal

Table 1. General recognition method types for all 18 details. DFS is the Depth-First Search pathfinding algorithm. Dijkstra's is the Dijkstra Shortest-Path Algorithm

The "region finding" category allows us to isolate some details, but if the shape is poorly drawn or missing entirely then isolating regions alone could not confirm shape neatness. This motivated the implementation of our third processing stage, which grades the isolated shape for correctness.

3.3 Stage 3: Detail Validation

Stage 3 compares only the **isolated sample** of the recognized detail to a set of template details to score the sample's quality. The system begins by centering, scaling, and resampling the isolated sample points so that they lie in a $[-1, 1]$ range on the x and y plane. We then create a map of some provided resolution n such that a detail is represented as a $n \times n$ matrix where space in the figure is mapped to a cell of the matrix, each cell being some range $[x_i, x_j]$, $[y_i, y_j]$. Each cell is then given a p-value based on the number of points that lie within the range for each cell. A visualization of this is shown in figure 6. The same system is applied to each of the templates, then each template matrix is then averaged cell-wise to form a template mapping. The template matrix Q is compared sampled detail matrix P with equation 4 to determine how closely the two matched. The best match is then found by shifting the the sample matrix by row and column to find the best possible position when compared to the template, given that some details will match in terms of their stroke and dimension but have somewhat different centers relative to the the template.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \max\left(0, Q_{i,j} - P_{i,j}\right) \tag{4}$$

The value of "distance" between template and sample is between 0 and 1, with 0 being the best. A shape that receives full credit for neatness is characterized as how close the sample is to the templates. Any value below 0.5 assigned to a sample is given full credit of 2 points. A value between 0.5 and 0.9 is given partial credit of 1 point. A value above 0.9 is given 0 points.

4 DATA COLLECTION AND RESULTS

We conducted a study with 68 cognitively healthy participants to complete a Rey-Osterrieth Complex Figure Test between the ages of 19-32. Although this test is meant to assess constructional ability and memory loss, healthy participants do not always score full marks on an ROCF [7], and indeed our testing corpus reflects a wide range of scores that conform to established normative data for our participants. All participants took the test in a simulated neuropsychologist's test

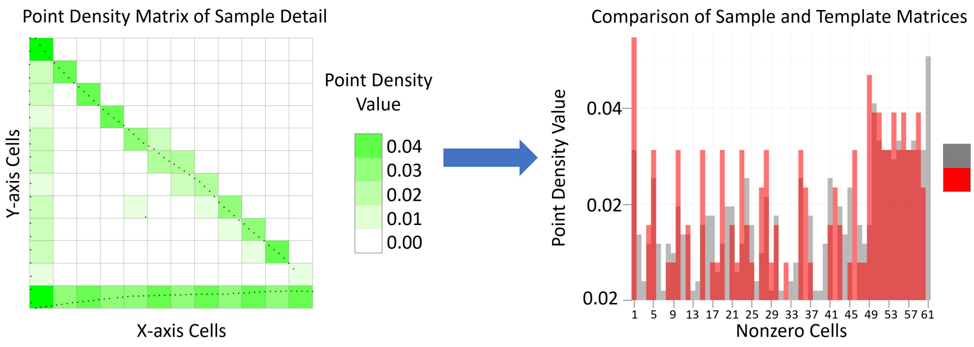


Fig. 6. Comparing two point-density matrices to asses detail distortion: our sample, and a saved template. The process is repeated for all templates.

environment and completed all three conditions (Copy, Recall, Delayed Recall). Participants were given a Neo SmartPen N2 and completed tests on pre-printed "blank" canvas pages that tracked the pen's location and instantaneously digitized all stroke data, allowing a more authentic testing experience since the ROCF is typically administered via pen and paper.

A total of 204 sketches from the 68 participants were collected. Of these, 5 perfect-score tests were set aside to be used as templates for Stage 3 validation. 14 were not gradable or their sketch data was corrupted, bringing the total graded to 185. All tests were also graded by two field experts whose grades we consider "ground truth" in this context. The first grader is a practicing clincial neuropsychologist and the second is a professor specializing in cognitive and visual perceptual rehabilitation in older adults. We measure our system's success in two ways: the F1-Score of our recognition algorithm for each detail, and the comparison between our system's total grade and the expert graders' total grade. For the latter, both our system's and the grades are on the 36-point scale as defined in Section 1.1.

A key factor considered when calculating F1-score was the subjectivity of distortion thresholds. While we implemented our own thresholds for distortion in Stage 3, instructions for the ROCF in the literature leave the definition of "distortion" at the discretion of the grader [23]. For recognition purposes we are interested in gauging whether our system can successfully either find a detail or confirm its absence. The F1-score reports our system's ability to recognize the existence of a detail. Since we are still interested in comparing 36-point grades that also integrate distortion as partial credit, we also calculate Spearman's rank coefficient ($\rho = 0.767$) between our automatically-graded tests and those of our expert grader.

5 DISCUSSION AND LIMITATIONS

Calculating F1-Score for recognition of the 18 details was conditional on whether detail 2 could be successfully recognized within a sketch. The organizational strategy score of the Rey-Osterrieth Complex Figure test places the highest priority on the existence of Detail 2 in a sketch due to its importance to the overall figure structure [20]. Exceptionally poor figures that lack a discernible detail 2 almost always result in very low or ungraded scores when hand-graded. For this reason, we have designed our recognition hierarchy such that the test is not graded if it cannot automatically recognize detail 2.

A total score of 0, however, is not necessarily due to a true negative. For 44 sketches, our algorithm was unable to find detail 2 due to a sloppy or unconnected drawing, but other details would exist.

Det. #	$\Delta_{a,g1}$	$\Delta_{a,g2}$	$\Delta_{g1,g2}$	**F1-Score**	**Det. #**	$\Delta_{a,g1}$	$\Delta_{a,g2}$	$\Delta_{g1,g2}$	**F1-Score**
1	0.69	0.68	0.37	0.787	**10**	0.21	0.20	0.10	0.962
2	0.18	0.44	0.33	0.857	**11**	0.47	0.45	0.21	0.878
3	0.07	0.11	0.16	0.978	**12**	0.46	0.44	0.11	0.872
4	0.27	0.30	0.12	0.927	**13**	0.13	0.11	0.11	0.966
5	0.08	0.49	0.49	0.978	**14**	0.42	0.41	0.15	0.881
6	0.31	0.56	0.26	0.958	**15**	0.50	0.49	0.09	0.770
7	0.47	0.26	0.13	0.855	**16**	0.21	0.19	0.08	0.919
8	0.28	0.33	0.13	0.966	**17**	0.57	0.66	0.34	0.788
9	0.35	0.20	0.07	0.904	**18**	0.45	0.49	0.21	0.925

Table 2. Classification results and average scoring differences for each detail across all graded tests. **n=141** for all details except for Detail 2, where **n=185**. $\Delta_{a,g1}$ denotes the **average point score difference** between Auto Rey-O and Grader 1, $\Delta_{a,g2}$ is the difference between Auto Rey-O and Grader 2, and $\Delta_{g1,g2}$ between Grader 1 and Grader 2.

If we flatly calculated F1-Score of all Details for every sketch included the ungraded ones, this would assign incorrect false negatives to the rest of the details. For that reason we tier F1-score calculations; for detail 2 we calculate it for all sketches (n=185), and for all other Details we calculate it where detail 2 was correctly detected (n=141).

The F1-Score results in Table 2 and the graph in Figure 7 demonstrate the effectiveness of Auto Rey-O. Our top-down system correctly identifies and validates the details with a high enough F1-score that shows the system working for typical test-takers. Table 2 also shows the average differences in scores (in points, maximum of 2) assigned between our Auto Rey-O system and our expert graders. All of the average differences in scores for each detail are well below 1 point and the vast majority below half a point, indicating a marginal difference in scoring between expert graders and our Auto Rey-O system. The system also works successfully for ROCFs with higher amounts of distortion. Such instances display the flexibility of the system in still identifying present details even if the participant has heavy lapses in memory.

The lowest-performing details are 1, 17 and 15. These had the highest amount of false negatives, although our manual review of these false negatives showed our system did recognize the details but chose to grant a score of 0 due to our threshold for distortion. Further refinements of distortion threshold values for these two details would improve their recognition quality.

Figure 7 compares the scores assigned by our automated grader and the two expert graders. Between the two expert graders, the correlation was $p = 0.948$, a Spearman's rank coefficient of $\rho = 0.942$, and an average difference in scores of $\Delta_{g_1,g_2} = 1.68$. Between our system and grader 1, $p = 0.799$, $\rho = 0.765$, and average $\Delta_{auto,g_1} = 3.21$. Between our system and grader 2, $p = 0.829$, $\rho = 0.802$, and average $\Delta_{auto,g_2} = 2.78$. Our automated system produced grades with a generally high correlation with those of the graders, although the grades from the experts were more similar to each other. In all three cases, low-scoring tests somewhat deviate across all graders, even between the expert graders. This is likely due to the aforementioned ambiguity in interpreting detail distortion. Our automated grader can also be observed to be consistently too strict on grading that produces consistently lower scores, which is partially attributed to the fact that it does not recognize details that were placed in the wrong location. In addition, at the suggestion of the expert graders who also served as domain experts, we chose to prioritize consistency in grading over leniency when deciding on partial credit thresholds since consistency is one of the key advantages of an automated recognition system.

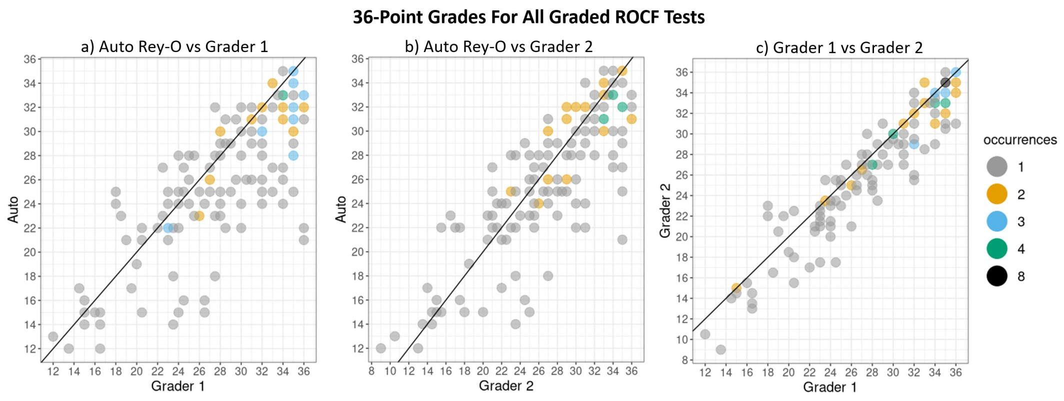


Fig. 7. Grade plots for all 36-point scores, compared between Auto Rey-O and expert graders (**n=141**). (a) p=0.799, (b) p=0.829, (c) p=0.948

The main limitation of this graph-based approach to top-down sketch recognition is the reliance on line connections. Our vertex-contraction algorithm in Step 1 of the system’s process does connect lines with corners within a certain radius. We found this technique worked very well if sketches were drawn with reasonable neatness. If the lines are disconnected by more than half an inch, however, these lines will remain disconnected. This was a conscious design choice since vertex contraction cannot be too aggressive; otherwise, regions where any correct sketch would have high numbers of vertices would all get incorrectly contracted into one. This is the case such as the area where detail 6, 7, 3, and 8 all converge—even neatly-drawn sketches have a high concentration of vertices here. We intend to improve refine the recognition system to “jump” gaps and close disconnected lines only where appropriate.

6 FUTURE WORK AND CONCLUSION

Refinements can be made to help recognize specific kinds of poorly-drawn details. As previously mentioned, most sources of grading inaccuracies for our system came from poorly connected graphs due to sketch sloppiness. For healthy participants taking this test, our expert graders attributed sloppiness as a lack of effort rather than genuine memory loss if the patient has no hand motor issues. Still, there would be an interest in supplementing our graph traversal with connecting otherwise unconnected vertices to improve recognition performance.

Additionally, improvements to our Stage validation approach could be made to recognize finer details. Our validation method sometimes may not properly distinguish between small changes, such as an extra stray mark or one line missing. Identifying missing lines is important for details 8 and 12, where the number of parallel lines drawn is relevant to its grading. Our validation method is able to find these discrepancies somewhat frequently, but potential for improvement exists.

Lastly, we aim to work with clinical neuropsychologists to administer their test to willing clients to evaluate system usability in a clinical setting. This would produce additional sketch data taken from actual patients, and would allow us to perform UI/UX usability studies for clinicians. The ultimate aim of the system is to aid diagnosis process by automating the grading of an ROCF, so evaluating the user experience of clinicians as they collect the digital data and use the Auto Rey-O application for themselves is the next step to further this project.

Our Auto ReyO automatic Rey-Osterrieth Complex Figure test grader demonstrates the validity of a top-down sketch recognition approach using graph traversal algorithms. This significantly simplifies the recognition process where a bottom-up approach would need to take into consideration a prohibitively wide array of possible shape interpretations and re-interpretations. By employing graph crawling, classical vertex search, and optimization algorithms we are able to identify key sub-shapes of geometric shapes.

REFERENCES

[1] Christine Alvarado and Randall Davis. 2007. SketchREAD: a multi-domain sketch recognition engine. In *ACM SIGGRAPH 2007 courses*. ACM, 34.

[2] Chris Calhoun, Thomas F Stahovich, Tolga Kurtoglu, and Levent Burak Kara. 2002. Recognizing multi-stroke symbols. In *AAAI Spring Symposium on Sketch Understanding*. Stanford University, AAAI Technical Report SS-02-08, AAAI Press, 15–23.

[3] RO Canham, SL Smith, and AM Tyrrell. 2005. Location of structural sections from within a highly distorted complex line drawing. *IEE Proceedings-Vision, Image and Signal Processing* 152, 6 (2005), 741–749.

[4] RO Canham, Stephen L Smith, and Andrew M Tyrrell. 2000. Automated scoring of a neuropsychological test: the rey osterrieth complex figure. In *Proceedings of the 26th Euromicro Conference. EUROMICRO 2000. Informatics: Inventing the Future*, Vol. 2. IEEE, 406–413.

[5] Edsger W Dijkstra et al. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

[6] Martin Field, Stephanie Valentine, Julie Linsey, and Tracy Hammond. 2011. Sketch Recognition Algorithms for Comparing Complex and Unpredictable Shapes. In *Proceedings of the Twenty-Second international Joint Conference on Artificial Intelligence (IJCAI)*, Vol. 3. AAAI Press, Barcelona, Spain, Spain, 2436–2441.

[7] Colin Gallagher and Teresa Burke. 2007. Age, gender and IQ effects on the Rey-Osterrieth complex figure test. *British Journal of Clinical Psychology* 46, 1 (2007), 35–45.

[8] Tracy Hammond and Randall Davis. 2003. LADDER: A Language to Describe Drawing, Display, and Editing in Sketch Recognition. In *Proceedings of the International Joint Conference on Aritificial Intelligence (IJCAI)*. AAAI, Alcapulco, Mexico, Mexico, 461–467.

[9] Levent Burak Kara and Thomas F Stahovich. 2004. Hierarchical parsing and recognition of hand-sketched diagrams. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*. 13–22.

[10] Muriel Deutsch Lezak, Diane B Howieson, David W Loring, Jill S Fischer, et al. 2004. *Neuropsychological assessment*. Oxford University Press, USA.

[11] Liang Lin, Xiaobai Liu, and Song-Chun Zhu. 2009. Layered graph matching with composite cluster sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 8 (2009), 1426–1442.

[12] Liang Lin, Tianfu Wu, Jake Porway, and Zijian Xu. 2009. A stochastic graph grammar for compositional object representation and recognition. *Pattern Recognition* 42, 7 (2009), 1297–1307.

[13] Momina Moetesum, Imran Siddiqi, Uzma Masroor, and Chawki Djeddi. 2015. Automated scoring of bender gestalt test using image analysis techniques. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 666–670.

[14] Momina Moetesum, Osama Zeeshan, and Imran Siddiqi. 2019. Multi-object sketch segmentation using convolutional object detectors. In *Tenth International Conference on Graphics and Image Processing (ICGIP 2018)*, Vol. 11069. International Society for Optics and Photonics.

[15] PA Osterrieth. 1944. Le test de copie d'une figure complexe. *Archives de Psychologie* 30 (1944), 205–550.

[16] Tom Y Ouyang and Randall Davis. 2011. ChemInk: a natural real-time recognition system for chemical drawings. In *Proceedings of the 16th international conference on Intelligent user interfaces*. ACM, 267–276.

[17] Alexander Prange, Michael Barz, and Daniel Sonntag. 2018. A categorisation and implementation of digital pen features for behaviour characterisation. *arXiv preprint arXiv:1810.03970* (2018).

[18] André Rey. 1941. L'examen psychologique dans les cas d'encéphalopathie traumatique.(Les problems.). *Archives de psychologie* (1941).

[19] Dean Rubine. 1991. *Specifying gestures by example*. Vol. 25. ACM.

[20] Cary R Savage, Lee Baer, Nancy J Keuthen, Halle D Brown, Scott L Rauch, and Michael A Jenike. 1999. Organizational strategies mediate nonverbal memory impairment in obsessive–compulsive disorder. *Biological psychiatry* 45, 7 (1999), 905–916.

[21] Min-Sup Shin, Yong-Hee Kim, Soo-Churl Cho, and Boong-Nyun Kim. 2003. Neuropsychologic characteristics of children with attention-deficit hyperactivity disorder (ADHD), learning disorder, and tic disorder on the Rey-Osterreith Complex Figure. *Journal of Child Neurology* 18, 12 (2003), 835–844.

[22] Min-Sup Shin, Sun-Young Park, Se-Ran Park, Soon-Ho Seol, and Jun Soo Kwon. 2006. Clinical and empirical applications of the Rey–Osterrieth complex figure test. *Nature protocols* 1, 2 (2006), 892.

[23] Esther Strauss, Elisabeth MS Sherman, Otfried Spreen, et al. 2006. *A compendium of neuropsychological tests: Administration, norms, and commentary.* American Chemical Society.

[24] Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing* 1, 2 (1972), 146–160.

[25] Radu-Daniel Vatavu. 2017. Improving gesture recognition accuracy on touch screens for users with low vision. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 4667–4679.

[26] J Vogt, H Kloosterman, S Vermeent, G Van Elswijk, R Dotsch, and B Schmand. 2019. Automated scoring of the Rey-Osterrieth Complex Figure Test using a deep-learning algorithm. *Archives of Clinical Neuropsychology* 34, 6 (2019), 836–836.

[27] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, 159–168.

[28] Aaron Wolin, Brian Eoff, and Tracy Hammond. 2008. ShortStraw: A Simple and Effective Corner Finder for Polylines.. In *SBM*. 33–40.

[29] Yiyan Xiong and Joseph J LaViola Jr. 2009. Revisiting shortstraw: improving corner finding in sketch-based interfaces. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*. ACM, 101–108.