Algorithmic Typewriter Art: Can 1000 Words Paint a Picture?

Roy G. Biv*
Starbucks Research

Ed Grimley†
Grimley Widgets, Inc.

Martha Stewart‡
Martha Stewart Enterprises
Microsoft Research

Figure 1: The left side shows 1 layer of typed characters. Moving right, overlapping layers are added up to 12 layers in 4 positions.

ABSTRACT

We develop an optimization-based algorithm for converting input photographs into typewriter art. Taking advantage of the typist's ability to move the paper in the typewriter, the optimization algorithm selects characters for four overlapping, staggered layers of type. By typing the characters as instructed, the typist can reproduce the image on the typewriter.

Compared to text-mode ASCII art, allowing characters to overlap greatly increases tonal range and spatial resolution, at the expense of exponentially increasing the search space. We use a simulated annealing search to find an approximate solution in this high-dimensional search space. Considering only one dimension at a time, we measure the effect of changing a single character in the simulated typed result, repeatedly iterating over all the characters composing the image.

Both simulated and physical typed results have a high degree of detail, while still being clearly recognizable as type art. The accuracy of the physical typed result is primarily limited by human error and the mechanics of the typewriter.

Index Terms: Computing methodologies—Non-photorealistic rendering—; Image Processing—

1 INTRODUCTION

Typewriter art involves producing images with typewritten text. A modern computer graphics practitioner is likely familiar with ASCII art, where an image is formed out of text characters on the screen. Typewriter art offers additional flexibility, insofar as characters are not restricted to a regular grid. Multiple characters can be typed at a single location, a practice called *overstriking*, and rows and columns of characters can partially overlap with previously typed rows and columns. Further, keys on the typewriter can be struck with varying levels of force, transferring greater or lesser quantities of ink from the typewriter ribbon. Varying the strike force produces a much smoother tonal range than monochrome ASCII art, which is especially important in lighter-tone regions of the image. Overstriking and overlapping improve outcomes in the darker regions of the image as well as improving detail.

In this paper, we present an algorithm for converting an input image into typewriter art, exploiting overstriking, overlapping, and strike force to add detail and to improve tone matching. We can directly render an output image, or produce a set of instructions that can be typed to create a physical realization of the image, in keeping with recent trends in computer graphics towards assisting computational fabrication [3]. Of course, we can create digital versions of the images, and by employing a computer font as input, bypass the typewriter entirely should that be desired by the user.

Manually crafted typewriter art can be extremely detailed, and historically was often created using primarily the period key or other small, geometric shapes. Without restriction to a regular grid, overlapping these small characters yields both fine spatial resolution and a perceptually wide tonal range, with a texture reminiscent of pointillism or stippling.

Considerable artistic skill was needed to craft these works. However, the typewriter also allowed users with less skill to produce images. "Typewriter mystery games" [15] provided typists with instructions that, when carried out, produced an image; e.g., see Figure 2 [4]. These instructions exploited the backspace key to enable overstriking to create much darker shades.

In our method, we produce instructions for four separate layers, each offset by half a space horizontally, vertically, or both; formally, the layers have their respective origins at (0,0), (0.5,0), (0, 0.5), and (0.5,0.5). Together, over-typing and half-spacing provide nearly full ink coverage. Figure 1 shows a rendered result, where superimposed layers of text cooperate to form a detailed image.

Our algorithm, outlined in Figure 3 takes as inputs a target image to be reproduced on the typewriter and a scan of the typewriter's character set. The program selects the characters to be typed for each of four layers, which then overlap to produce the image. Within each layer, the placement of characters is limited to a grid. The

*e-mail: roy.g.biv@aol.com
†e-mail: ed.grimley@aol.com
‡e-mail: martha.stewart@marthastewart.com

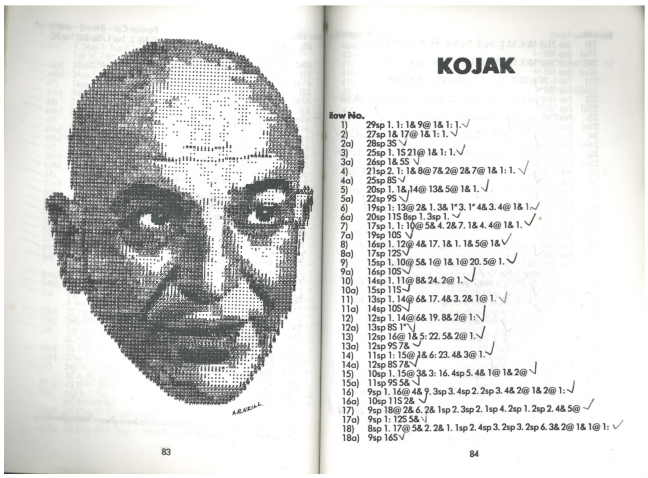

Figure 2: Instructions from Bob Neill's Book of Typewriter Art

algorithm optimizes by measuring the effect of changing a single character in the simulated typed result, repeatedly iterating over all character positions until no change to any single character can improve the outcome. The selected characters for each layer form the typist's instructions. By following these instructions the image can be reproduced on the typewriter without demanding any artistic skill.

With respect to both tone reproduction and shape matching, this technique produces a better approximation of the input image than non-overlapping ASCII art with a similar number of characters. We posed the question "Can 1000 words paint a picture?"; we find that 5790 characters suffices for a fair facsimile of a portrait, though more resolution is required for complex scenes.

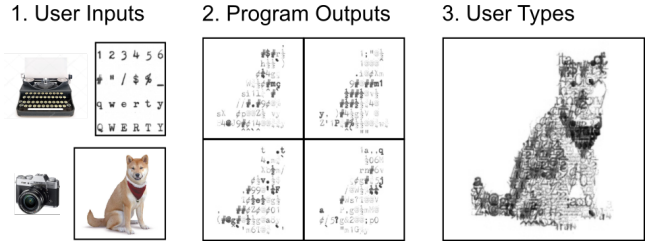

Figure 3: Simplified process

This paper makes the following contributions:

- We exploited overlapping characters to increase the dynamic range and expressivity of text art.
- We present an optimization algorithm for automatic creation of typewriter art, paying special attention to physical reproducibility on a mechanical typewriter.
- We propose *asymmetric mean squared error*, in which positive error (too little ink) is weighted less than negative error (too much ink). Subjectively, optimizing with asymmetric MSE produced the best results.

2 Background

Much recent work in ASCII art has focused on improving shape matching, which can be traced back to the introduction of the Structural Similarity metric (SSIM) [21]. Another metric, created specifically for ASCII art, is the Alignment Insensitive Structural Similarity Metric (AISS) [18]; Xu et al. deform the target image to better match the available character shapes at a given position. Deforming the input image poses issues for the high-fidelity approach we pursue, but the idea of optimizing the alignment of the target image proves useful.

Conventionally, ASCII art used monospaced fonts. Xu et al. [19, 20] achieved superior results using proportional-width fonts. Although this approach provides flexibility in the columns of type, the rows of type are still fixed; typewriter art allows both rows and columns to vary.

Some recent approaches to generating ASCII art involve machine learning with no explicit metric. Akiyama employed a convolutional neural network, trained on manually created structural ASCII art, to produce compelling results for this style [2]. Markus et al. use a decision tree to approximate SSIM comparison for a particular character set, yielding a good approximation at high speed [14]. With our interest in overlapping characters, these trained model approaches pose the issue that only single characters are stored in the model, so overlapping composites of those characters would not be compared with the target image.

In typewriter art, the characters can in principle be freely positioned, which brings to mind stippling [7]. Computer-generated stippling usually seeks non-overlapping stipples, unlike our case which encourages overlaps. Also, while stippling need not be restricted to points [6, 10], computer-generated stippling methods do not exploit choice of object shape to better approximate an input image, while this is a fundamental aspect of ASCII art and typewriter art.

Typewriter art can be seen as a variant of halftoning [8], where a continuous-tone input image is represented with a small number of discrete tones. Our output images are quantized in space (discrete character placement positions) and tone (individual character selection) with the quantization partly hidden by overlapping characters. We employ optimization directly informed by an error metric, similar to the method of Pang et al. [17].

3 Algorithm

The aim of this project was to marry the mechanical reproducibility of the "typewriter mystery game" with the improved spatial resolution of freehand typewriter art by employing four overlapping layers, offset both vertically and horizontally. We developed an algorithm which considers the effect of overlapping characters.

Figure 4: Nine offset characters overlap within one character position

The algorithm takes as input a target image to be reproduced, and an image of a set of characters (which can be produced by scanning typed characters or from a computer font). The "character set" is chopped into a set of images, one for each character plus one blank character.

3.1 Search technique

We take an iterative approach, choosing a character for a single position at a time. Each selection answers the question "What character, placed in this position, will best complement the already-chosen characters to most closely match the target image?"

With greedy search, we find the best character for a particular position independent of the rest. We then do the same for all character

positions in the typed image. This completes a single optimization pass.

Algorithm 1: Simplified optimization algorithm

```
Create list of character positions [charPos]: {layer, row, col,
  charId};
Initialize each charId in [charPos] randomly;
for i = [1.. maxIterations] do
    for pos in [charPos] do
        Create list of character scores [charScores]: {charId,
          score};
        for char in [charScores] do
            char.score = compare(simulated typed output,
              target image at this position);
        end
        pos.charId = highest scoring char in [charScores] for
          this pos;
    end
end
Simulate composite typed output;
Simulate individual typeable layers;
```

Due to the use of overlapping layers, past selections may later become suboptimal: when any character overlapping a position changes, that position must be re-evaluated. For example, if all neighbours of a certain position changed from a dark character to a light one, the selection for that position should possibly be changed to a darker one to compensate. Moreover, because all the characters are connected through overlap, changing a single selection can trigger a cascade of selection changes spanning the image.

3.2 Stopping condition

A character position is considered stable if none of its neighbours have changed since it was last evaluated; thus, re-evaluating the position would lead to the same selection as before. The search terminates when no further improvement can occur by changing any one character selection. A stable state is reached when a complete optimization pass has occurred with no selection changes.

It is possible that the algorithm will never reach a stable state, as the cascades of selection changes create cycles. To ensure termination, we limit the search to 500 selections per character position. In practice, this limit was never reached, with 25 iterations being typical.

3.3 Loss function

We relied on previous work in image fidelity measurement to provide the error function used to score character selection. We also employed a variation on mean squared error (MSE), *asymmetric mean squared error* (AMSE), multiplies positive error by a factor of $1+a$ before squaring.

A combined loss function $(1-ssim) \times amse$ and maximizing SSIM were also explored.

3.4 Optimized cropping

Before the iterative search begins, we first generate quick approximations – using a single optimization pass – for each of 64 slightly different crops of the target image. The crop parameters (translation and scale) that maximize SSIM + PSNR are applied to the input before iterative optimization.

3.5 Simulated annealing

A greedy best-first search evaluates every position along a single dimension, selecting the character which results in the highest similarity to the target image. However, there is no guarantee that this is optimal in other dimensions. The greedy search is quick to converge to a local optimum, where no single character swap will increase the similarity to the target image.

To combat the lock-in to local optima exhibited by greedy search, we use stochastic SA to intelligently expand the search space. SA is modelled after the physical process of heating and cooling metal to reduce defects [9]. At any selection, it is more probably that a high scoring candidate will be chosen than a lower scoring one. The width of this probability distribution is decreased over iterations, as the "temperature" is reduced by a fixed amount after each optimization pass. When the temperature reaches 0, it "reheats" to a percentage of the initial temperature.

Each time the algorithm visits a character location, it evaluates the candidate characters in a random order. If selecting the character increases the similarity score, it is chosen. A lower scoring character may also be selected with probability inversely proportional to the delta between the current score and the score resulting from its selection. In other words, a very bad selection is possible, but a better selection is more probable, and increasingly so at each iteration, as the temperature is reduced.

4 Results

4.1 Loss function

As a loss function, standard MSE works very well in this application. MSE matches tone quality well, and the penalty for larger error at a single pixel gives it a sharper, shape matching quality than mean absolute error.

AMSE penalizes positive error (too much ink) to a greater degree than negative error (too little ink). This has two advantages: first, erroneously placed ink is subjectively more noticeable; second, an overlapping character can fill in missing ink later, while erroneously placed ink cannot be removed. In practice, AMSE improves shape matching – both subjectively and as measured by SSIM on the resultant image – while only slightly compromising MSE.

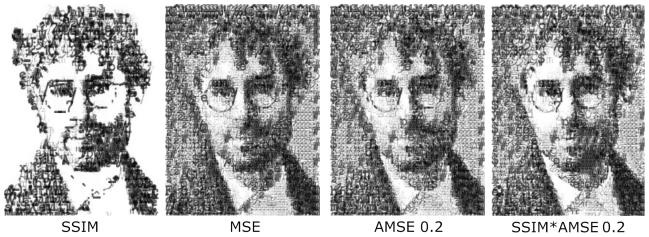

Figure 5: SSIM, MSE, AMSE, (1-SSIM)*AMSE ($a = 0.2$) [20w]

As a loss function SSIM differed markedly from MSE in this application, working much like an edge detector with little respect for tone matching. At small sizes, SSIM can yield a linear, sketch-like style.

At small sizes, some of the best results come from blending SSIM with AMSE. The blended loss function performs well at sizes of around 20 characters (Figure 5); we use the notation [20w] to indicate a result generated at 20 characters in width with no overstrike, and [20w, 2p] to indicate 20 characters in width with 2 overstrike passes. SSIM slightly emphasizes lines while leaving some lighter areas bare of ink, effectively increasing contrast.

4.2 Search technique

4.2.1 Greedy search sensitivity to initial state

By default, we begin the search from an initial blank page where every character selection is the blank character (spacebar). To evaluate the effectiveness of greedy search, we performed multiple runs of the algorithm from different initial states, including random states

as well as single-pass approximations. The exhibited sensitivity to initial state (shown in Figure 8) indicates that the local optimum reached by a greedy search may still be some distance from the global optimum.

4.2.2 Simulated Annealing (SA)

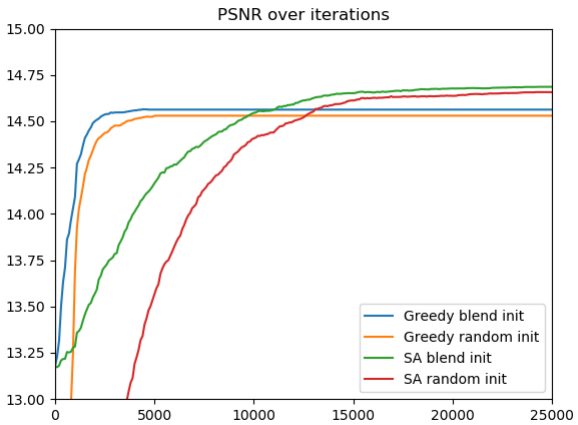

Figure 6: Effect of search technique on score and convergence time

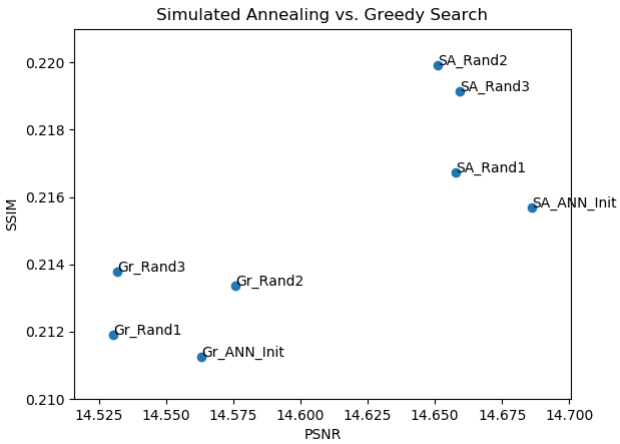

Figure 7: Effect of different initial states on score

Despite taking longer to converge (Figure 6), SA consistently finds a lower-error state than greedy search, and is less affected by initial state (Figure 7). The results using SA from different random states (Figure 8) are hard to distinguish. However, all SA results captured the whites of the eyes while some greedy results blurred them out.

4.2.3 Selection order

When using a greedy search, the order in which selections are made (which character position, or dimension, is evaluated next) has a predictable influence on the output. This is especially evident when the search is initiated from a blank state. The first selections will tend to be overly dark, as the single character under evaluation takes full responsibility for tone matching over an area that could ultimately include 9 overlapping characters; see Figure 4. The

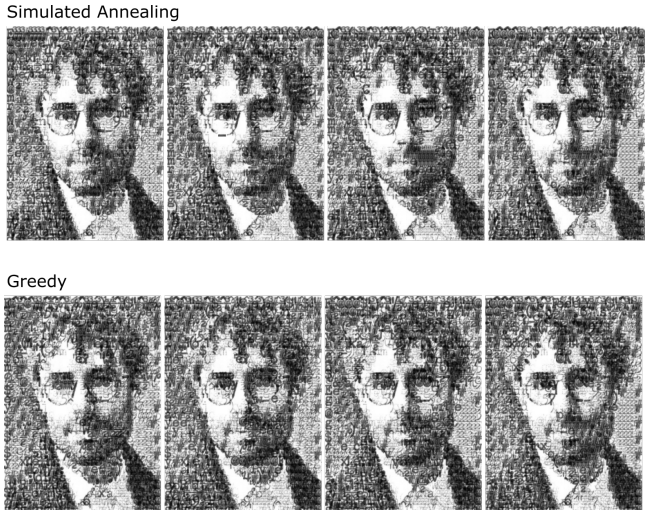

Figure 8: Greedy and simulated annealing results from random initial states [20w]

greedy search tends to become trapped in local optima with overly dark initial selections. The characters that are selected first contribute disproportionately to the result. Our intuition was that this would lead to a suboptimal result.

We experimented with priority-ordered selection – where the priority of a position is determined by the maximal increase in score resulting from selecting the best character at that position – but found that it does not improve the results when simulated annealing is used, and comes at a high computational cost. While a linear selection order resulted in noticeable artifacts, a random selection order was sufficient to avoid these without incurring high computational cost and hence was used for all our results.

4.3 Pre-processing the target image

Since many source images are colour, the black and white conversion method has a substantial effect on the result (Figure 9), as does edge enhancement and adjustments to the contrast and brightness [11, 13]. We considered these processes out of scope, but have worked to optimize one critical aspect of image pre-processing unique to this application: the alignment of the target image to the typewriter's "character grid" .

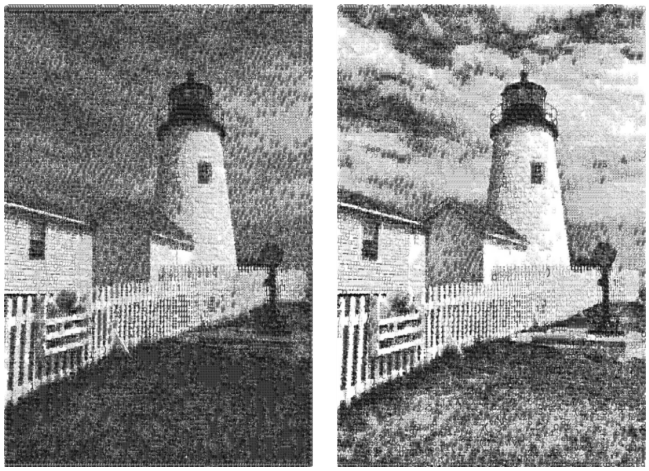

Figure 9: RGB average vs. STRESS algorithm [40w]

4.3.1 Optimized cropping

Figure 10 shows two results for the same checkered test pattern, with and without cropping (scaling and shifting the image within the same fixed-size container). The results after cropping via the auto-alignment routine are much sharper. In realistic scenarios – rendering a portrait, say – the alignment still has a strong effect.

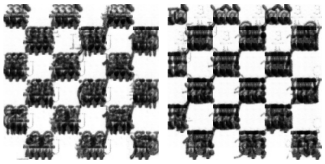

Figure 10: Effect of optimized cropping on test pattern. Left: no crop; Right: optimized crop [12w]

In Figure 11, the target image has been slightly scaled and shifted to better align with the "character grid" of the typed result. Note the improvement in shape resolution of the eyes, glasses, hair and nose. At small image sizes, alignment is one of the dominant factors in the quality of the result.

Figure 11: Effect of optimized cropping on portrait [20w]

4.4 Number of "overstrike" characters at a single position and number of total characters

Despite having partly overlapping typed characters, we have so far discussed only one character to be selected at a particular location. Making use of the backspace key, the typist can "overstrike" multiple characters at the same position within the same layer. This increases the tonal range, especially allowing darker tones, and can to some extent increase shape matching.

The algorithm incorporates the backspace key by using the simulated result produced by one run of the program as a background layer during compositing operations in a second pass. The most obvious effect is an expanded tonal range in darker regions of the image, along with increased gradient smoothness and shape matching (Figure 12, Figure 13).

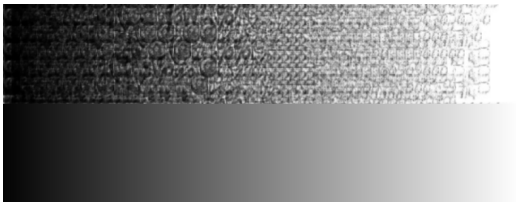

Figure 12: Top: Simulated typed gradient [30w, 3p]. Bottom: Input gradient.

In exploring selection order, we learned that it was preferable to distribute ink between several layers, rather than concentrating ink in a few. By default, adding a second overstrike pass will contribute

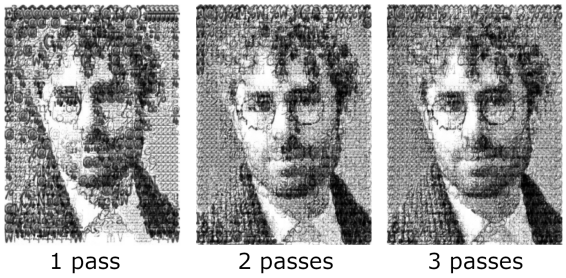

Figure 13: Distributing ink between multiple overstrike passes [19w]

little ink, as the majority of the ink has already been placed in the first pass. With the AMSE metric, we can increase the penalty for wrongly placed ink (asymmetry) on the first overstrike pass and reduce it on the second pass. A higher asymmetry factor encourages the selection of characters which are an especially good match in some areas, but have too little ink in others, over darker characters which do not match the shape as closely. The second pass, with a lower asymmetry, can fill in the gaps.

Unsurprisingly, increasing character resolution produces better results. Figure 14 shows two simulations, each with three overstrike passes, at 20 characters wide and 30 characters wide. The higher resolution yielded a marked improvement in detail reproduction.

Figure 14: Left: short row length [20w,3p], Right: longer row length [30w,3p]

Keeping to a fixed maximum number of characters, generally the best results come from dividing the characters between two to three overstrike passes. Darker images especially benefit from overstrike, as seen in Figure 15. The increased tonal range in the shadows more than compensates for the reduction in size, up to around four overstrike passes. With a lighter image, the trade-off favors a larger size with two layers, as overstrike is less needed in light areas.

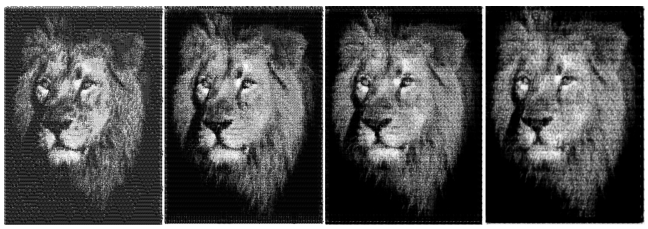

Figure 15: Left to right: [45w,1p], [32w,2p], [23w,4p], [16w,8p]

4.5 Character set

The choice of typewriter greatly influences the outcome, as it forms the character set that it is available to the algorithm. With most

Figure 16: Characters from Smith-Corona typewriter with dry ribbon, typed hard and soft

mechanical typewriters, the typist can vary their strike force to produce lighter and darker variations of the same character. This has dramatic results: including lighter variations greatly increases tonal range in the midtones.

Figure 17 shows the results with different typed character sets and the SF Mono computer font. The leftmost images, created with monotone character sets, exhibit limited tonal range and draw more attention to the textual characters. In comparing these two images alone, the computer font does not reproduce the image as effectively. The uneven distribution of ink on the typewriter characters aids in shape matching, compared to the perfectly flat distribution across the character a computer font. Critically, even the darkest typewriter characters are less than fully black, which yields a greater tonal range when characters are allowed to overlap.

The center and right columns in Figure 17 exhibit progressively better tonal range as the character set includes a better distribution of greys. Even if we limit ourselves to two intensities, results improve by having medium and light intensities, rather than dark and medium. There is a trade-off here: overlapping lighter characters can produce a smoother tonal range, at the expense of requiring more overlaps to produce full black.

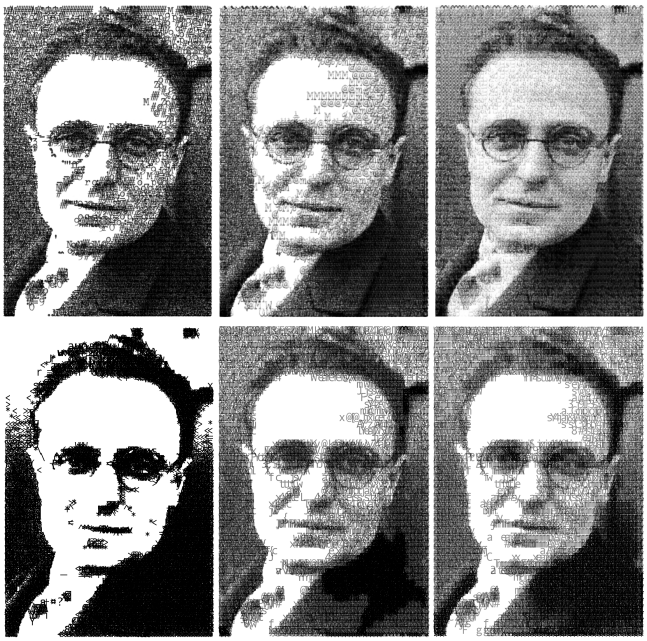

Figure 17: Top: Typed characters, Bottom: SF Mono font; Left to right: 1 tone, 2 tones, 2 lighter tones [30w,4p]

4.6 Timing notes

The algorithm is expensive due to the large number of image compositing and comparison operations required to determine the best character selection for a given position, multiplied by the number of iterations before convergence. Each of these operations is performed at a high resolution (around 1000 pixels total) at 8-bit depth.

We leaned on Google Colaboratory, a cloud computer service which supplies 2 threads of an Intel Xeon processor at 2.3GHz, and 13GB of RAM [5]. The software is written in Python 3, making use of numpy, OpenCV and scikit-image libraries.

The top left image in Figure 18 is composed of two overstrike passes of 2670 characters each. To converge to a stable state the algorithm visited each character position 25 times. The first pass took 29 minutes to complete; subsequent overstrike passes converge more quickly. Auto-alignment took an additional 10 minutes. In total, the image took approximately one hour to generate. These timings are representative and vary little over different input photographs.

The time to converge increases exponentially with number of characters: at 420 characters, the algorithm took 0.64 seconds per character position; at 2670 characters, 0.77 seconds; and at 6392 characters, 0.94 seconds.

5 Evaluation

The metric used for comparison between the target image and the simulated typed result has, subjectively, the strongest effect on the results. Best results were obtained from an asymmetric variation on MSE which disproportionately penalizes wrongly placed ink. Blending this metric with SSIM can sharpen results, especially when the typed image is small.

Greedy search will necessarily converge to a local optimum in which each dimension is at a global optimum in that dimension (no single character swap can improve the score). We can get closer to the overall global optimum state by using simulated annealing.

Alignment of the target image with the typewriter's "character grid" can have a strong effect on shape matching. Alignment can be optimized by measuring the effect of translation and scaling on the target image, selecting the best parameters for each. This is the primary factor when the typed image is small.

Allowing multiple characters to be typed at a single position can substantially increase tonal range and shape matching, especially when the character set is monotone (with no variation in strike force). Allowing for variation in strike force is the largest factor in obtaining a good tonal range in the midtones and highlights.

5.1 Settings for best results

To best answer the question posed by the title of this paper - can 1000 words paint a picture - we limit ourselves to only typing 5790 characters. This number of characters reflects the average English word length (4.79) plus one space per word [16]. For a 4:3 portrait as the target image, this allows for 25 characters width, with 4 overlapping staggered layers and up to 3 typed characters at any single position while remaining within our budget of 5790 characters.

The settings described below result in the highest combined SSIM + PSNR. To the author's eye, this metric best reflects overall quality as explored in 4.1.

- Run auto-alignment for the maximal row length within character budget. Choose the crop parameters that maximize SSIM + PSNR.
- Use a character set which includes all typewriter characters, typed both hard and soft.
- Search with simulated annealing, reheating to 10% each time the temperature reaches 0
- Start from a random state and randomize the selection order
- Run the algorithm 2 times (2 overstrike passes). Use AMSE, with asymmetry settings $a = 1.5$ on the first pass, and $a = 0.1$ on the second.

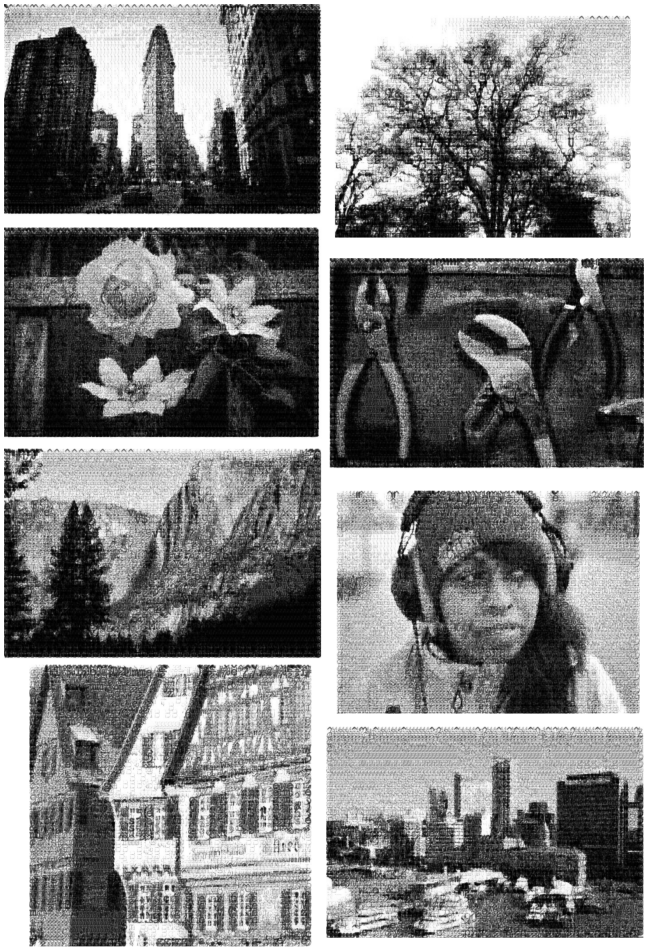

Figure 18: Gallery of results with consistent settings [¡=5790 characters,2p]. Zoom to see details.

Figure 18 shows 8 images produced with these settings: maximizing our character budget of 5790 characters and distributing the characters across two overstrike passes. In general, the simulated typed results communicate the content and mood of the input photographs regardless of content.

To represent noisy textures, the algorithm selects a variety of overlapping characters, demonstrated in the thin tree branches, or the windows of the Flatiron Building (top left). In the absence of texture, characters with more even ink distribution – such as @ or # – are selected. This applies to both light and dark regions, with the darkness of the character and number of overlaps determining overall tone. This is evident in the skin tones of the woman with headphones and in the sky of several images, as well as the gradient shown in Figure 12.

Employing overstrike achieves good tonal range in the shadows, as seen in the handles of the wrenches and the windows of darker buildings; and a deep black level, demonstrated in the landscape and flowers results. Further emphasizing the value of overstrike, light shapes against a dark background are especially clear, like the round nail suspending the rightmost wrench. While the fixed character grid is set at 1/2 character resolution, some details in the output demonstrate a spatial resolution of less than 1/16 of a character.

In general, individual characters are difficult to discern except in the lightest areas, where characters do not overlap – e.g., the sky in the cityscape. The algorithm selects characters which nest, hiding their textual origins, unless the character shapes are a particularly good match – as in the underscores used for the decks of the cruise ships.

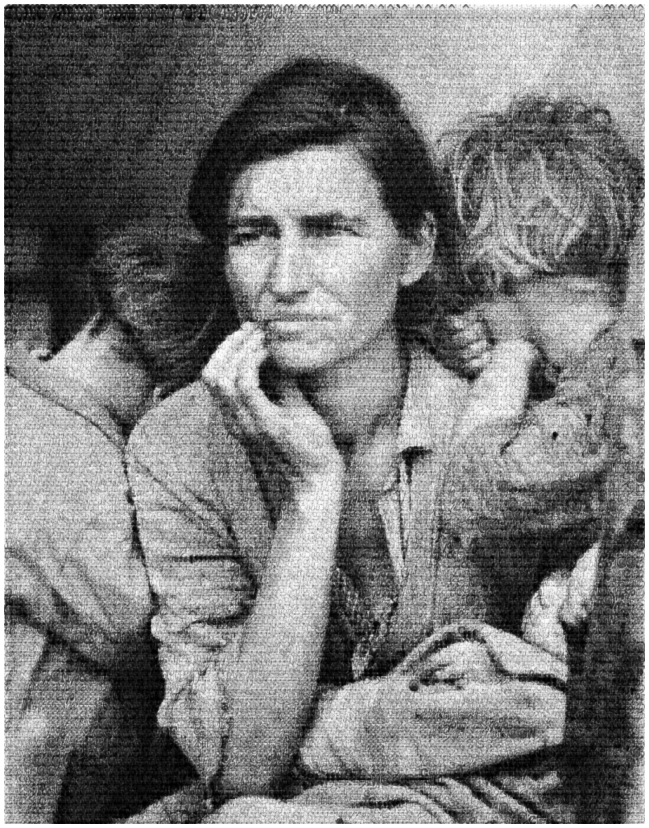

Figure 19: Migrant mother, optimized settings [68w,3p]

Details with low local contrast, such as the face of the woman with headphones, display coarser spatial resolution than areas of high local contrast such as the edges of buildings. This is due to relying on overlapping characters to produce sharp edges. When additional overlap would cause a poorer tonal match, shape resolution suffers. This explains why metrics preferring shape matching result in higher local contrast.

However, relying on MSE suits a high-fidelity approach. Accurate tone matching makes the shadows in the cityscape and waterfront images easy to discern; the three-dimensional nature of the buildings is clearly conveyed. Employing a shape-matching metric such as SSIM obscures this.

Increasing the size of the typed image improves quality. In Figure 19, the output size of 5” x 8” (68 characters wide) is sufficient to reproduce an image with multiple persons with excellent quality, conveying the emotional impact of the input photograph.

5.2 Physical typed results

Physical reproducibility on a mechanical typewriter is a key feature of this work. Figure 20 shows the result of following the generated instructions to type a 70 cm^2 image over 6 hours.

Typing the results revealed difficulties in replicating the rendered results. The physical typed result in Figure 21 shows false textures and loss of blackness in the body of the horses, especially in the bottom right quadrant. Even slight misalignment of the offset layers creates gaps between tightly packed characters – especially noticeable when thin characters like $\frac{1}{2}$ overlap within areas of uniform tone. This is an example of over-optimization: while packing thin characters can improve outcomes in the simulation, in practice the

Figure 20: Physical typed result, in typewriter [40w,2p]

physical result would be improved by favoring wider, overlapping characters in smooth-textured regions.

Tonally, the gradient produced in the rendered result is reflected in the physical typed result, albeit with a compressed dynamic range. While this partly caused by alignment errors, another factor is the incongruity between the subtractive color space of typed ink and the additive color space of the simulation. The simulation uses multiplicative compositing, as a best guess at how the ink would behave. In reality, the combination of ink and paper imposes a lower bound on the black level, while inconsistent strike force creates a crude step between the lightest reproducible tones and the white paper background. The imperfect tone curve of the scanner used to capture the character set also limits the accuracy of reproducing light tones - the characters appear lighter in the rendered result, shown in Figure 22.

Simulated typed result Physical typed result

Figure 21: Horses: Rendered vs. physical typed results [40w,2p]

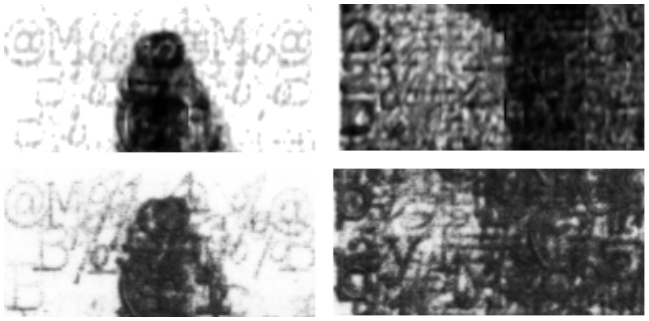

Figure 22: Details of horses. Top: Rendered, Bottom: Physical

Figure 22 shows two details from Figure 21. The bottom left image – a detail of the top-most rows – shows excellent alignment of the characters. Tone matching is accurate overall, but the variation in strike force is apparent on B and % characters. In contrast, the right image – a detail of lower rows – demonstrates substantial misalignment: note how the $\frac{1}{2}$ and y characters are shifted right and down relative to the b. While the layers were properly aligned at the top of the image, moving the page to create an offset layer can introduce unwanted rotation. In this case, the page was rotated counter-clockwise on the layer containing $\frac{1}{2}$ and y, relative to the layer containing the leftmost b characters. This blurs the edge delineating the fence post and the body of the pony, and creates gaps in what should be a uniform texture, visible on the right.

While the typed result of the horses has few misplaced characters (0.15%), achieving such a result is difficult. The left image in Figure 23 shows the most apparent typing error: adding an extra character within a line. This image also reveals the difficulty in typing at more than two dynamic levels; hard, medium and soft were attempted, succeeding only in a smoother gradient in the beard. The right image took our current approach of only hard and soft strikes, limiting the tonal resolution but improving consistency of the background tone, thus avoiding unintended texture.

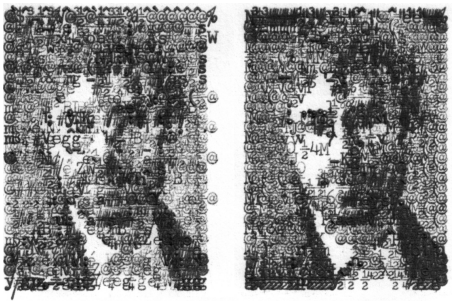

Figure 23: Two attempts at typing the same image [20w,2p]

5.3 Limitations

5.3.1 Physical limitations

The algorithm is sufficiently optimized that human error in typing is a primary limitation. It is especially difficult to use precise strike force. While the addition of softly typed characters into the character set is crucial for producing smooth midtones and highlights without employing dither, physical limitations make reliable reproduction of the lightest tones difficult.

The typewriter itself is also a source of error. The distribution of ink on the ribbon is not perfectly uniform, so the typed characters which were scanned to create the character set may not be exactly

reproduced when typed later. The typewriter may drift out of alignment due to variation in the amount the platen is moved or rotated at each keystroke.

These issues can be ameliorated somewhat by improved preparation of the character set. Multiple typed copies of each character could be averaged both in intensity and alignment. This would reduce tone error due to variation in strike force and ribbon inkiness, and increase placement consistency between the simulated and physical typed output.

5.3.2 Algorithmic limitations

The most notable limitation of this method is that the optimization is quite time-consuming. We were primarily concerned with image quality; future work could focus on acceleration.

Halftoning-type patterns can emerge which falsely delineate tonally consistent areas, creating spurious shapes. Repeated characters create false textures. Both of these can be addressed by penalizing repeated adjacent characters. and/or through dithering. For large images, dither would also subjectively improve tone matching over the highlight areas [8] [12].

Our implementation of "overstrike" was made for pragmatic reasons; in it, each pass is computed in sequence with no backtracking, so selections made in the first pass are not revisited. For example, two slash characters \and / typed in the same position might provide a better shape match than the typewriter's serif X character, but this might only be realized if both "overstrike layers" are considered at once.

6 Conclusions and Future Work

The typewriter, as an analog mechanical device, possesses degrees of freedom not readily available to digital ASCII art: multiple characters can be typed at a single grid location; typing is not restricted to a grid; and characters can be typed with different levels of force. Our method is able to produce high quality output, both physical (by manually typing the selected characters) and digital (displaying the simulated typed result).

We used four overlapping layers of characters as a compromise balancing mechanical reproducibility (can the image be physically typed?) and image quality (does it faithfully represent the input?). We were able to obtain good results over a variety of subject matter under these conditions; dark tones and fine details are handled with overlapping and overstriking, while application of lighter strike force helps increase fidelity in light areas of the image.

We suggest a few different directions for future work, both in modifying details of the existing algorithm and in extending automatic typewriter art to handle additional degrees of freedom. We discuss both types of future work below.

As the choice of error metric has a strong effect on the result, other metrics should be explored. To improve shape matching, we could employ Alignment Insensitive Shape Similarity, created for ASCII art. To produce results with good tone matching, it would have to be used in combination with other metrics. Machine optimization of the metric blending function's parameters could improve results.

Freehand typewriter artists such as Paul Smith take advantage of rotating the page: characters can be typed at any angle [1]. We would like to investigate allowing some layers to be typed with the paper rotated.

Finally, this work has focused on achieving fidelity by optimizing the algorithm more than optimizing physical reproduction. The physical results could be improved by exploring human factors or robotics in future work.

References

[1] The Paul Smith Foundation home page.

[2] O. Akiyama. ASCII art synthesis with convolutional networks. In *NIPS Workshop on Machine Learning for Creativity and Design*, 2017.

[3] A. H. Bermano, T. Funkhouser, and S. Rusinkiewicz. State of the Art in Methods and Representations for Fabrication-Aware Design. *Computer Graphics Forum*, 2017. doi: 10.1111/cgf.13146

[4] Bob Neill. *Bob Neill's second book of typewriter art*. 1984.

[5] T. Carneiro, R. V. Medeiros Da NóBrega, T. Nepomuceno, G. Bian, V. H. C. De Albuquerque, and P. P. R. Filho. Performance analysis of Google Colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, 6:61677–61685, 2018.

[6] K. Dalal, A. W. Klein, Y. Liu, and K. Smith. A spectral approach to NPR packing. In *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '06, p. 71–78. Association for Computing Machinery, New York, NY, USA, 2006.

[7] O. Deussen, S. Hiller, C. Van Overveld, and T. Strothotte. Floating points: A method for computing stipple drawings. *Computer Graphics Forum*, 19(3):41–50, 2000.

[8] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial greyscale. *Proceedings of the Society for Information Display*, 17(2):75–77, 1976.

[9] D. Henderson, S. H. Jacobson, and A. W. Johnson. The theory and practice of simulated annealing. In F. Glover and G. A. Kochenberger, eds., *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, pp. 287–319. Springer US, Boston, MA, 2003. doi: 10.1007/0-306-48056-5_10

[10] S. Hiller, H. Hellwig, and O. Deussen. Beyond stippling – methods for distributing objects on the plane. *Computer Graphics Forum*, 22(3):515–522, 2003. doi: 10.1111/1467-8659.00699

[11] Kolås, I. Farup, and A. Rizzi. STRESS: A framework for spatial color algorithms. *Journal of Imaging Science and Technology*, 55:040503, 2011.

[12] H. Li and D. Mould. Contrast-aware halftoning. In *Computer Graphics Forum*, vol. 29, pp. 273–280. Wiley Online Library, 2010.

[13] Q. Liu, S. Li, J. Xiong, and B. Qin. WpmDecolor: weighted projection maximum solver for contrast-preserving decolorization. *The Visual Computer*, 35(2):205–221, Feb. 2019. doi: 10.1007/s00371-017-1464-8

[14] N. Markuš, M. Fratarcangeli, I. S. Pandžić, and J. Ahlberg. Fast rendering of image mosaics and ASCII art. *Computer Graphics Forum*, 34(6):251–261, 2015.

[15] J. Nelson. *Typewriter mystery games*. Artistic Typing Headquarters, Baltimore, USA, 1979.

[16] P. Norvig. English letter frequency counts: Mayzner revisited or ETAOIN SRHLDCU.

[17] W.-M. Pang, Y. Qu, T.-T. Wong, D. Cohen-Or, and P.-A. Heng. Structure-aware halftoning. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08. Association for Computing Machinery, New York, NY, USA, 2008.

[18] X. Xu, L. Zhang, and T.-T. Wong. Structure-based ASCII art. *ACM Transactions on Graphics*, 29(4):52:1–52:9, July 2010.

[19] X. Xu, L. Zhong, M. Xie, X. Liu, J. Qin, and T. Wong. ASCII art synthesis from natural photographs. *IEEE Transactions on Visualization and Computer Graphics*, 23(8):1910–1923, Aug. 2017. doi: 10.1109/TVCG.2016.2569084

[20] X. Xu, L. Zhong, M. Xie, J. Qin, Y. Chen, Q. Jin, T.-T. Wong, and G. Han. Texture-aware ASCII art synthesis with proportional fonts. In *Proceedings of the workshop on Non-Photorealistic Animation and Rendering*, NPAR '15, pp. 183–193. Eurographics Association, Istanbul, Turkey, June 2015.

[21] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, Apr. 2004. doi: 10.1109/TIP.2003.819861