Paper Flattening Using a Deep Learning Approach

Arnav Sharma
University of Michigan
Ann Arbor, Michigan, USA
arnavsha@umich.edu

Austin Jeffries
University of Michigan
Ann Arbor, Michigan, USA
ajeffr@umich.edu

Ali Badreddine
University of Michigan
Ann Arbor, Michigan, USA
abadredd@umich.edu

1. Executive Overview

Folded and bent paper when scanned can be very difficult to read. The creases, shadows, and general sense of turmoil among the lines, letters, and numbers hinders a person's ability to digest the information shown. Our idea is to flatten folded sheets of paper using deep learning and transfer learning, generate answer keys for math worksheets using template matching, and incorporate a web-app as a visual aid of the dataset and worksheet grading. The project website and code can be found at https://arnavsharma.github.io/the-flatteners/. An in-house interactive demo of dataset generation and worksheet grading can be found at https://shrouded-temple-50673.herokuapp.com/.

2. Background and Impact

With the COVID-19 pandemic affecting people's lives worldwide remote learning has become the standard. This has resulted in teachers having to digitally receive assignments as images and grade them. However, taking a perfectly aligned image that is not crumbled is a difficult task to achieve and enforce amongst large groups of students. We seek to work with the reality of remote grading being hindered by these mentioned difficulties. Our approach is to assume work sheets that are submitted as images are distorted and perturbed and we take a deep learning approach to flatten these papers so they are easily visible. We also assume that teachers these days would want to grade these sheets in an automated manner and we extend our work to take a non deep learning approach motivated by concepts of template matching and grouping/clustering treating the worksheet images as graphs and functions.

2.1. Related Work

Document flattening has been approached through a variety of different methods in the past. For instance, some systems use calibrated stereo cameras to measure the 3D distortion of the document [2] while others look for handcrafted, low-level features – such as shading and text-lines – in order to recover and flatten the document. These approaches tend to produce excellent results, but they can be limited in their application and general hardware availability. For example, a calibrated system may yield better results but the hardware is not as widespread as using a single image from a smartphone camera [6].

To address this problem, more recent work has explored using a learning-based approach in order to flatten documents. Both [4] and [3] utilize a U-Net architecture which is often used in image segmentation models. They use the U-Net to learn the correct mapping between a flattened and non-flattened image. These two works have shown very promising results and thus this paper looks at replicating this success and building off of it.

3. Methods

There are three key computer vision methods employed in this project. They include deep learning and transfer learning, generating a synthetic distorted 3D paper mesh with realistic lighting, and template matching with K-means clustering.

3.1. Deep Learning and Transfer Learning

Motivated by the learning-based method [3] to reconstruct document images, we begin by partitioning the input image into a grid of patches with overlap in both the horizontal and vertical directions, and evaluate each of them with a well-trained model. This model gives us the local distortion flow for each patch where the flow is a 2D displacement vector at each pixel showing the mapping between the distorted image and the flattened image. The generalized model architecture is in Fig. 1, and the details regarding the U-Net architecture are in Fig. 2. Equipped with flows and patches, we can then re-construct the document in a way that minimizes distortion. Furthermore, since shadows and sampling issues are still prevalent, we take the output flattened image through a second model which evenly illuminates and removes shadows on the image.

3.2. Generating Synthetic Distorted 3D Paper with Realistic Lighting

The training model consists of many CNN layers and therefore needs to be trained with many samples [3]. In this project, a completely synthetic dataset was generated

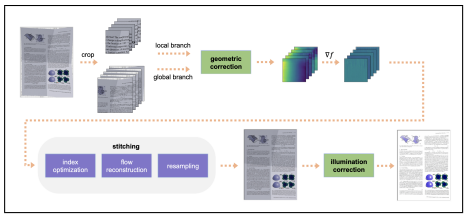

Figure 1. Generalized neural network architecture [3]

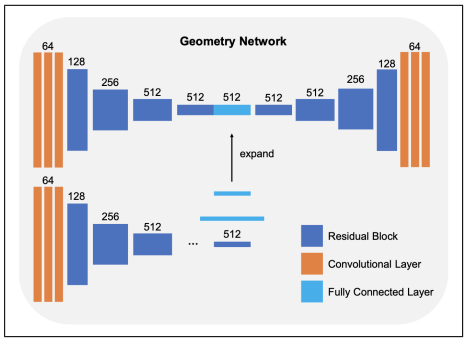

Figure 2. Detailed look at U-Net architecture [3].

to reduce time and complexity of gathering real-world examples. The relevant components of generating such data are a distorted 3D mesh, warping of desired image on to the 3D mesh, gathering the 2D flow map and mask image, applying a background to the warped paper, illuminating the paper with realistic lighting to highlight the creases and faces of the paper. Four examples of this technique are provided in Fig. 3.

3.2.1 General Distorted 3D Mesh

We distort a flat plane 3D mesh at random vertices in all three directions. This will help train the model in learning the 3D distortions to then be able to flatten the creased papers. There are two empirical guidelines that are followed when creating the distortions:

- Paper is a rigid body. It cannot shrink or stretch. The random 3D vertex at which the paper is perturbed is then propagated as a line across the mesh to mimic folds and creases.
- Although in reality paper can be folded or curled up, we focus on folded paper to save time when generating the dataset.

The functions and general scaling of perturbations was adapted from existing code [4].

Once the perturbations are applied, the desired image is warped on to the surface and the folds and creases become apparent. A background is applied that can mimic a wooden table and many other surfaces [5].

3.2.2 Lighting

Lighting in computer vision is crucial to understand. The model's performance greatly improves when realistic lighting and shadows are applied to the paper surface. We assumed that paper is a Lambertian surface in a well-lit ambient environment. The diffusive property of paper gives it the matte finish. By incorporating Plotly's lighting properties such as ambient, diffuse, specularity, roughness, and fresnel. Since the diffuse property is providing the the matte finish of the paper, we will assume a roughness value of 0.

3.3. Template Matching with K-means Clustering

Template matching is an exciting computer vision method of finding a particular sub-image within a larger image. The sub-image is called the template image, hence this method is template matching. In this method, the template image sweeps across the larger image (as in 2D convolution). The matches (or detections) lead us to finding the digits and math symbols on the worksheet. The position within the larger image is also outputted. Template matching can be scale and rotation invariant; however, we have assumed neither are true in this project. K-means clustering clusters detections based on distance to each other. With proper tuning, the clustering pairs a numerical problem's digits with its respective math operation symbol.

4. Prototype

4.1. Dataset Generation

Math worksheets produced in batch are adopted from https://webmathminute.com/sheets. The problems include Addition, Subtraction, Multiplication, and Division with 1-3 digit numbers. 100 different sheets were generated from the site and fed through the automated dataset generation script.

The script outputs the 2D flow map, an image of the perturbed math worksheet, and a mask image. We also varied the lighting orientation, selection of background images and developed 2700 images for our dataset. The size is roughly 18 gigabytes (GB) worth of data. For reference, [4] utilized a similar number of samples as well.

Four examples of this technique are provided in Fig. 3.

4.2. Model Training

Motivated by deep learning work from different sources {[3], [4]}, we utilized their network to map distorted images to their estimated respective flow maps in an effort to reconstruct a flat paper representation of the image. The network used in [3] takes advantage of a U-Net which lends

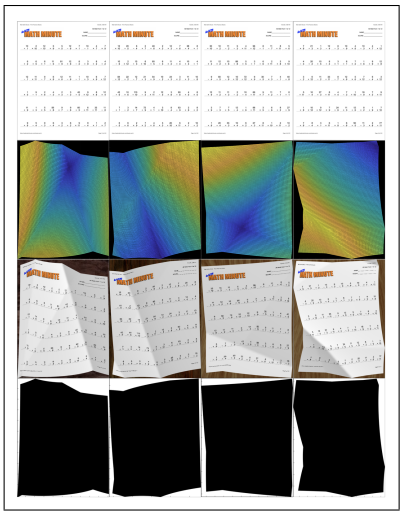

Figure 3. Sample dataset with the original image, 3D mesh, warped paper, and mask image.

well to semantic registration problems. We input in local and global image patches to capture a wide variety of spatial features in the image hoping for optimal training/test results.

4.3. Answer Key Generation

Generating answer keys for the math worksheets requires two different computer vision methods, template matching and K-means clustering. These methods are used to detect the digits, math symbols, and problems as clusters. Combining these two methods allows for quick, efficient answer key generation.

4.3.1 Digits and Math Symbols

Given the math worksheets all contain the same size digits and math symbols, template matching proved to be a robust way in reading the worksheet. Each individual digit from $0-9$ and each math symbol including $+, -, \times,$ and $\div$ are saved as individual template images. As seen in Fig. 4, the template images are discrete images. The edges of the digits and symbols are rough and not smooth.

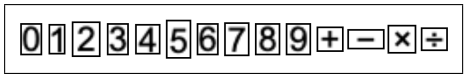

Figure 4. Digits and math symbols for template matching method.

The lack of smoothness around the edges of the digits and symbols requires the OpenCV template matching algorithm to use a threshold. For example, the gray pixels between the body and tail of the digit 9 could match the 0 template image. Varying thresholds for each digit and symbol allowed for more accurate detections. Over 10 worksheets, similar to those in the top row of Fig. 3, the well-tuned algorithm performed at a reasonable level.

One nuance of this approach that actually leans in our favor is that template matching is not scale or rotation invariant. The other digits around the page for page number, date and time of printing, and worksheet version were not detected and therefore did not affect the template matching algorithm. Another nuance is that template matching can provide duplicate detections one part of the input image. To eliminate this failure mode, a mask of the input image was filled with white over every distinct detection.

4.3.2 Problem Clustering

The math problems on each worksheet are organized in a consistent manner. They are spaced and set up evenly and consistently across each worksheet. This allows for the K-means algorithm to cluster the detections as defined above into a defined number of 60 distinct clusters.

Each cluster contains a math symbol and a minimum of two digits. If more than two digits are detected, utilizing a DataFrame sorting algorithm helps to determine the order of each digit in the two numbers by investigating the x and y positions of the detections.

Finally, the appropriate digits are combined for each number and then the desired math operation is performed. The result is printed in red text below the equation line. The answer key worksheet is saved in the same directory for future viewing.

5. Results

5.1. Deep Learning

For our first attempt the training was validated with an 80/20 percentage split of training to test data for local and global patches with a resolution factor of 4 and a patch size of 256 pixels. However, this yielded poor results. In an effort to improve the results and to build on the novelty of the network by [4] we decided to add illumination to the generated training data since [4] suffered from shading issues in varied lighting conditions. With this new set of data, we ran multiple training sessions utilizing GPU-acceleration (CUDA) for improved speeds. The settings (number of epochs, local patch resolution, learning rate, resolution factor of global patch to local patch, training/test data split, and minimum achieved Expected Predicted Error (EPE) losses) for the training are in Table 1.

As seen in Fig. 5, we achieved the best results in our second run with an 80/20 split, res factor of 6, and learning rate of 0.01. This was achieved at the fourth epoch.

Considering the above method worked decently well after being trained on a modest dataset size of 2700 im-

Table 1. Training setups across three runs.

Run	Epochs	Patch (px)	L-Rate	Res Factor	Training/Test Split (%)	Min. EPE Loss
1	5	128	0.001	4	70/30	2.17
2	4	64	0.01	6	80/20	0.51
3	5	64	0.001	4	80/20	1.12

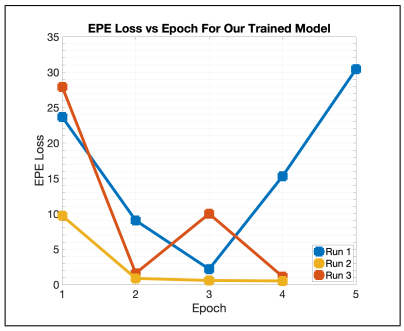

Figure 5. EPE Loss vs Epoch over 3 different training runs.

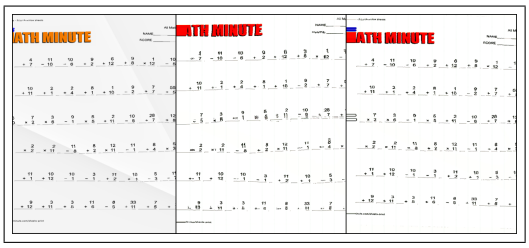

Figure 6. 1) Input image with the distortions and shadows.; 2) Output of our trained model; 3) Output of pre-trained model from [3].

ages, we decided to explore improving these results through transfer learning [1] on a previously trained model. Since the model created by [3] produced excellent results for a wide variety of different types of documents, we decided to initialize our model training with the same weights of their model to hopefully extend their work to a larger dataset with more variety in types of creases and folds. However, likely due to subtle differences between our dataset and the pre-trained dataset, the model did not learn as expected.

5.2. Answer Key Generation

The template matching algorithm on the worksheets used in our dataset works most of the time. It is generally robust and tuned appropriately to avoid misclassifying digits and symbols. On the other hand, the K-means clustering algorithm requires some more tuning for even better performance as it works almost all the time. However, it is usable and the Web-App interface shows as such. An example of the template matching mask, detections, clustering, and generated answer key worksheet are in Fig. 7.

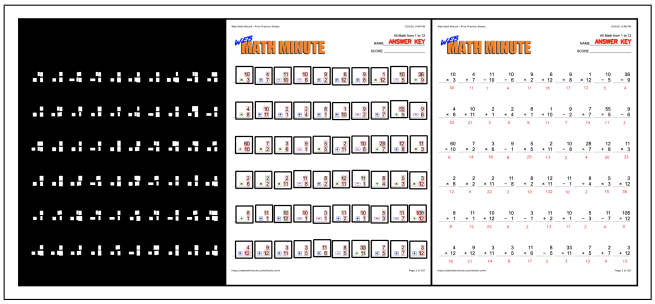

Figure 7. 1) Generated mask to eliminate duplicate detections.; 2) Visualizing all detections and clustering.; 3) Generated answer key worksheet.

6. Acknowledgements

We would like to thank Professor Jason Corso for inspiring and motivating us to work on this project idea. His excitement and fervor regarding this project led us to continue through the difficulties we faced. We would also like to appreciate the GSIs, Oscar de Lima and Parker Koch, for their tremendous support. Their suggestions and recommendations throughout the project provided us with a new angle and approach to completing this project.

In addition, we would like to show our gratitude to Zhixin Shu (Adobe Research) as well as Sagnik Das (Stony Brook University) for connecting us with Ke Ma, a PhD candidate at Stony Brook University. Ke provided us with a base code to generate the document dataset. Without this starter code, we would not have been able to generate our own training dataset within the project's allotted time.

References

[1] Stanford University CS231n. Transfer learning. https://cs231n.github.io/transfer-learning/. 4

[2] Sagnik Das, Ke Ma, Zhixin Shu, Dimitris Samaras, and Roy Shilkrot. Dewarpnet: Single-image document unwarping with stacked 3d and 2d regression networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[3] Xiaoyu Li, Bo Zhang, Jing Liao, and Pedro V. Sander. Document rectification and illumination correction using a patch-based cnn, 2019. 1, 2, 4

[4] Ke Ma, Zhixin Shu, Xue Bai, Jue Wang, and Dimitris Samaras. Docunet: Document image unwarping via a stacked u-net. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2, 3

[5] Oxford University. Describable textures dataset (dtd). https://www.robots.ox.ac.uk/~vgg/data/dtd/. 2

[6] Toshikazu Wada, Hiroyuki Ukida, and Takashi Matsuyama. Shape from shading with interreflections under a proximal light source. In *International Journal of Computer Vision volume*, 1997. 1