Pattern Recognition Lab

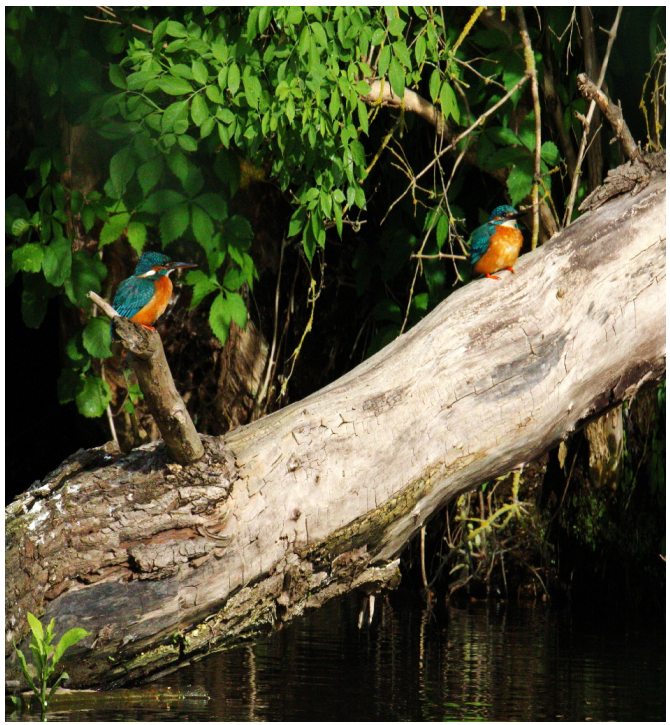
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Seam Carving

Image Resizing & Text Line Segmentation

Mathias Seuret, LME, April 26, 2021

Teaser



Outline

- Problem introduction
- Energy function
- Computing seams
- Application
  - Downscaling
  - Upscaling
  - Text lines segmentation

Pattern Recognition Lab

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Problem Introduction

This is a high-resolution image. Let's make it smaller to share it !

In this case, cropping is simple and efficient.

This is a high-resolution image. Let's make it smaller to share it !

In this case, cropping is simple and efficient.

This is a high-resolution image. Let's make it smaller to share it !

In this case, cropping is simple and efficient.

What about this picture?

Going back to the common kestrel...

- Why can we crop the image?
- Some parts contain little information
- Here, unimportant areas are uniform
- For this lecture: uniform areas are less important
- By extension: pixels similar to their neighbors are of lower importance

Going back to the common kestrel...

- Why can we crop the image?
- Some parts contain little information
- Here, unimportant areas are uniform
- For this lecture: uniform areas are less important
- By extension: pixels similar to their neighbors are of lower importance

Going back to the common kestrel...

- Why can we crop the image?
- Some parts contain little information
- Here, unimportant areas are uniform
- For this lecture: uniform areas are less important
- By extension: pixels similar to their neighbors are of lower importance

Cost Function

We can use the Laplacian of the pixels to compute a pixel's importance.

The Laplacian of a function $F\left(x_1, \ldots, x_n\right)$ is defined as

$$\vec{\nabla}\left(\vec{\nabla} F\right) = \sum_i \frac{\partial^2}{\partial x_i^2} F \tag{1}$$

Let $\Phi\left(x, y\right)$ be a function which returns the value of a pixel at coordinates $\left(x, y\right)$.

Then we have

$$\begin{aligned} \frac{\partial}{\partial x}\Phi\left(x, y\right) &= \frac{\Phi\left(x+\frac{1}{2}, y\right) - \Phi\left(x-\frac{1}{2}, y\right)}{1} \\ \frac{\partial^2}{\partial x^2}\Phi\left(x, y\right) &= \Phi\left(x+1, y\right) + \Phi\left(x-1, y\right) - 2 \cdot \Phi\left(x, y\right) \end{aligned} \tag{2}$$

We can compute the same for $\frac{\partial}{\partial y}$ and $\frac{\partial^2}{\partial y^2}$.

Trick: using $\pm\frac{1}{2}$ in Equation 2 allows simplifications

So, we have

$$\vec{\nabla}\left(\vec{\nabla}\Phi(x,y)\right) = \Phi(x+1,y) + \Phi(x-1,y) + \Phi(x,y+1) + \Phi(x,y-1) - 4\cdot\Phi(x,y) \tag{3}$$

We can make a 3×3 convolutional filter out of this:

0	1	0
1	-4	1
0	1	0

After convolving this filter on each channel of an image, we can take the absolute values of the results, and sum them to get the cost, or importance, of the pixels.

Pattern Recognition Lab

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Removing Rows & Columns

Common kestrel photo:

1. Compute pixels cost
2. Look for column with lowest cost
3. Remove it
4. Compute pixels cost
5. Look for row with lowest cost
6. Remove it
7. If small enough: break
8. Goto 1

In this image, recomputing costs is not really needed.

In the general case, it is.

What about this picture with a couple of kingfishers?

What about this picture with a couple of kingfishers?

What about this picture with a couple of kingfishers?

What about this picture with a couple of kingfishers?

Two issues remain to be solved:

1. Some elements (eg., kingfishers) must not be modified

2. Removing rows and columns cause large distortions

Solution 1: using a mask to increase values of some pixels

Solution 2: seam carving

Two issues remain to be solved:

1. Some elements (eg., kingfishers) must not be modified

2. Removing rows and columns cause large distortions

Solution 1: using a mask to increase values of some pixels

Solution 2: seam carving

To add a mask to the pixels' cost, create an image of the same size and paint in white pixels which you want to protect.

Important:

- You must resize the mask in the same way as the image
- Do not forget to re-add it to the costs whenever you update them


+


Pattern Recognition Lab

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Seam Carving

Seam carving in short:

- Each pixel has a cost/energy
- Find a path (or seam) which crosses the image such that:
  - The path never goes backward,
  - It can move laterally by a limited distance,
  - The sum of its pixels is minimal
- For image size modification:
  - Remove the path (instead of removing cols/rows) to downscale
  - Double the path width (with interpolation) to upscale
- For text line segmentation:
  - Find a path between two lines which avoids the ink
  - Ink has high cost, “paper” low cost

Many path finding algorithms exist:

- **Dijkstra**,
- **A***,
- Bellman-Ford,
- Floyd-Warshall,
- Johnson,
- ...

Here, we use two properties of the problem:

- Pixels are arranged as a grid,
- It is not possible to move backward

to produce a simple, yet efficient path finding algorithm.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

Let us consider the following cost array.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

We need first to compute the special cumulative sum on the right.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

The first row is identical. Let us see how to compute the highlighted cell.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

We must consider the cells above it. In this case, we use from 1 to the left, to 1 to the right.

This “1” corresponds to the maximum horizontal offset of the seam we want to compute.

Larger values are possible, but in all our examples we will use 1. Of course, feel free to try using more if you are curious.

像素的量矩阵

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

累积向量矩阵

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

$8 = 4+4$

6=5+min([1,3,5])

递推关系

You probaly already guessed it: the target cell is the sum of the minimum value in the considered cells above it, plus its corresponding cost.

Thus, you can compute all rows sequentially.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

Now that we have the array on the right, we will see how to compute a minimum cost seam.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

We will proceed backwards!

First, let us look for the smallest value of the last row.

In our case we have 3 of them, and we can pick any of them, they are equivalent.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

`numpy.argmin`, which you most likely will use, returns the index of the first one, so let us continue with this approach.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

Again, we consider the 3 cells above the current one.

在上一行找三个中最小的

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

Again, we consider the 3 cells above the current one. The optimal path to get to the current cell necessarily went through the one with the lowest value, 11 in this case.

Indeed, if we consider all of the possible paths leading to the three cells above, then none can have a smallest value than the one with the smallest value (11 in this case). Thus, a path with the smallest cost necessarily goes through this cell.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

We can go on in the same fashion up to the top.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

We can go on in the same fashion up to the top.

Always selecting the first lowest cell, or a random cell containig the lowest value.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

We can go on in the same fashion up to the top.

Always selecting the first lowest cell, or a random cell containig the lowest value.

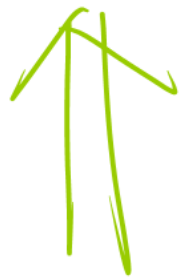
4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15

Continuing like this to the top of the array gives a minimum cost seam.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	3	2	1	3	5	4
5	7	5	4	6	4	7
9	6	7	6	8	8	6
7	11	9	8	11	7	7
11	9	9	11	9	9	11
14	11	14	14	11	13	10
14	16	12	15	12	12	15



Continuing like this to the top of the array gives a minimum cost seam.

Note that there can be many such seams, not only a single one.

找到几个裂缝 seam .

Pattern Recognition Lab

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Resizing Images

Downscaling images is pretty straightforward. Compute a seam. Let us assume that it crosses pixels (x_i, i), where i is row numbers.

For all i, offset to the left the values of pixels $(x > x_i, i)$, and thus overwrite the value of (x_i, i).

Example for $i = 0$:

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

4	2	1	3	5	4	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

Once you are done, remove the last column from the image.

Programming hints

You have to work with an image (three channels), a cost array, a cumulative cost array, and a mask.

I recommend to store all of these in a Numpy array with 6 channels:

- The mask can be modified together with the image, no need for duplicated code
- It opens the possibility to easily update only a part of the cost and cumulative cost arrays
  - Parts of them far from seams are not impacted by image modifications
  - Optimizing the area to update can save much time

Do not implement the seam carving twice, for vertical and horizontal seams. Instead, rotate your images by 90 degrees.

Upscaling an image, i.e., making it wider or taller, is done by *expanding* the seams.

Upscaling an image, i.e., making it wider or taller, is done by *expanding* the seams.

4	3	2	1	3	5	4
2	5	4	3	5	1	3
4	1	3	2	4	4	2
1	5	3	2	5	1	1
4	2	1	3	2	2	4
5	2	5	5	2	4	1
3	5	1	4	1	2	5

To expand a seam, we consider the pixels which are on its right as well.

Note that this implies seams should not reach the side of the image! You can prevent this simply by setting a very high cost on the right-most column.

Upscaling an image, i.e., making it wider or taller, is done by *expanding* the seams.

4	3		2	1	3	5	4
2		5	4	3	5	1	3
4	1		3	2	4	4	2
1		5	3	2	5	1	1
4	2		1	3	2	2	4
5	2		5	5	2	4	1
3	5	1		4	1	2	5

First, we need to copy the data in an array which is 1 pixel wider. To shrink the image, the values of the pixels on the right of the seam were offseted to the left. To expand the image, we offset them to the right.

This leaves a gap to fill.

Upscaling an image, i.e., making it wider or taller, is done by *expanding* the seams.

4	3	2	2	1	3	5	4
2	3	5	4	3	5	1	3
4	1	2	3	2	4	4	2
1	3	5	3	2	5	1	1
4	2	1	1	3	2	2	4
5	2	3	5	5	2	4	1
3	5	1	2	4	1	2	5

We can fill the gap by computing the average value between the pixels on their left and right.

Here, these values have been rounded to the lower integer.

One final issue has to be solved.

What is missing in this expansion approach?

One final issue has to be solved.

What is missing in this expansion approach?

Expanding the images several times *always* lead to this error.

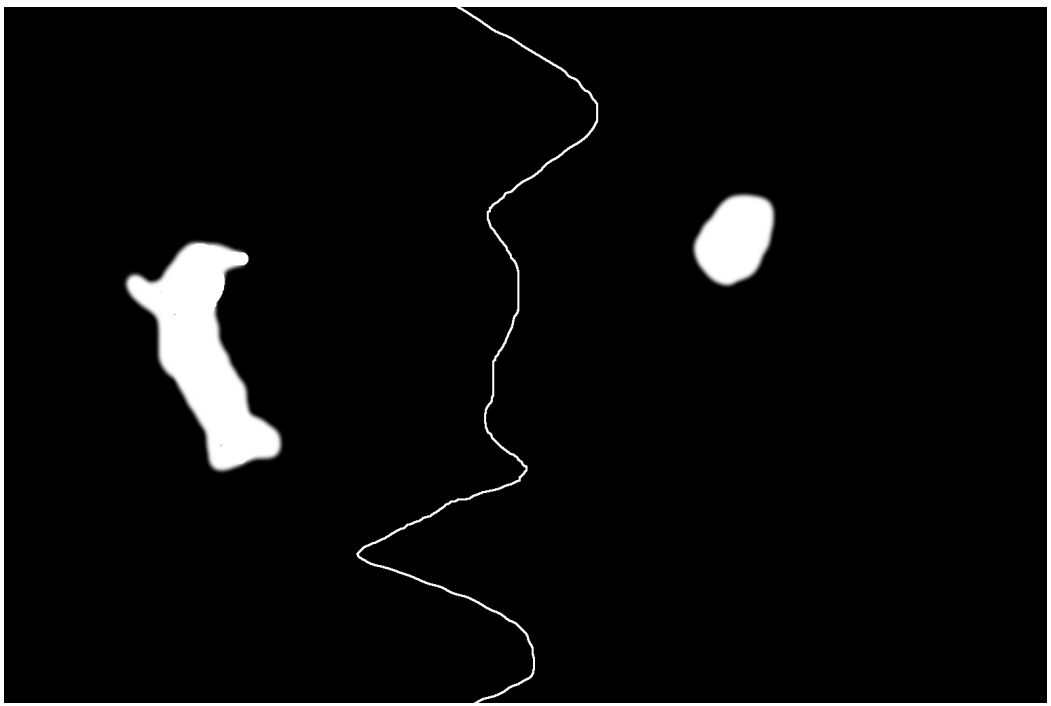
One final issue has to be solved.

What is missing in this expansion approach?

Expanding the images several times *always* lead to this error.
Either the first seam or the new values have the lowest cost, therefore the image will be expanded at the same place again and again.

To avoid always having seams at the same location, you can update the mask.

Whenever you expand at a seam, increase slightly the mask values at the corresponding locations $((x_i, i)$ and $(x_i + 1, i)$. New seams at the same location will then be less likely.



Hint: also do this when downscaling the images; the results will look better.

Pattern Recognition Lab

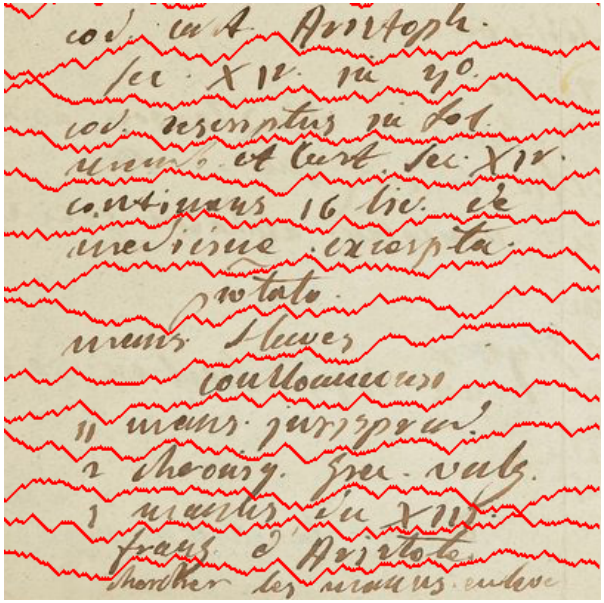
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Line Segmentation

Seam Carving is also much used for text line segmentation.

- Handwritten text is not straight
- Some letters (e.g., t or h) have ascenders,
- Other (e.g., p, q) have descenders
- Text lines can touch each others
- High quality needed? Forget about bounding boxes

Line segmentation:



Two difficulties:

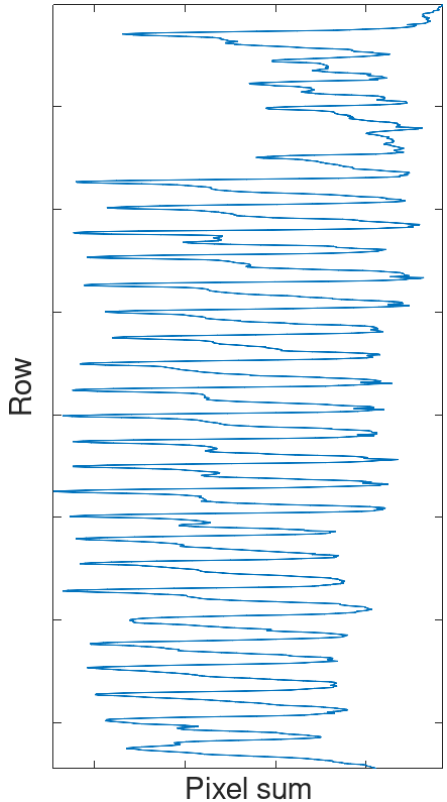
- Where do lines start?
- Where is the ink?

Sum rows of pixel costs to find text lines locations. Local minimas are typically between text lines.

To find line start only, consider only the left side of the text block. You can carve from there to the right side.

Typically, line ends are also computed, and matched to the starts.

Finding line starts:



Two difficulties:

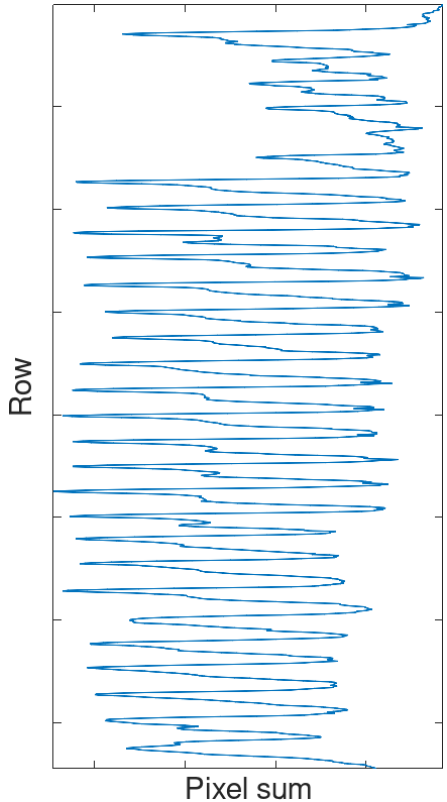
- Where do lines start?
- Where is the ink?

Sum rows of pixel costs to find text lines locations. Local minimas are typically between text lines.

To find line start only, consider only the left side of the text block. You can carve from there to the right side.

Typically, line ends are also computed, and matched to the starts.

Finding line starts:



Two difficulties:

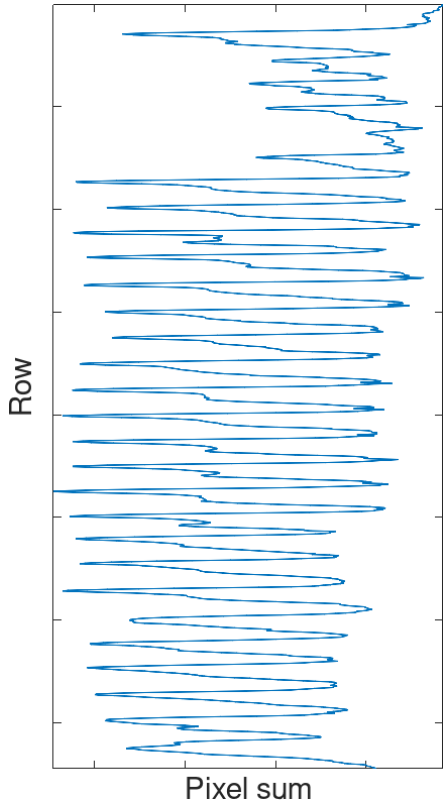
- Where do lines start?
- Where is the ink?

Sum rows of pixel costs to find text lines locations. Local minimas are typically between text lines.

To find line start only, consider only the left side of the text block. You can carve from there to the right side.

Typically, line ends are also computed, and matched to the starts.

Finding line starts:



Two difficulties:

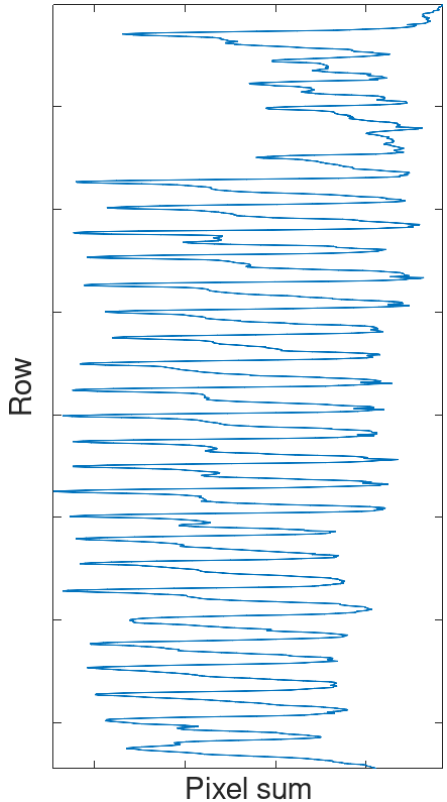
- Where do lines start?
- Where is the ink?

Sum rows of pixel costs to find text lines locations. Local minimas are typically between text lines.

To find line start only, consider only the left side of the text block. You can carve from there to the right side.

Typically, line ends are also computed, and matched to the starts.

Finding line starts:



Two difficulties:

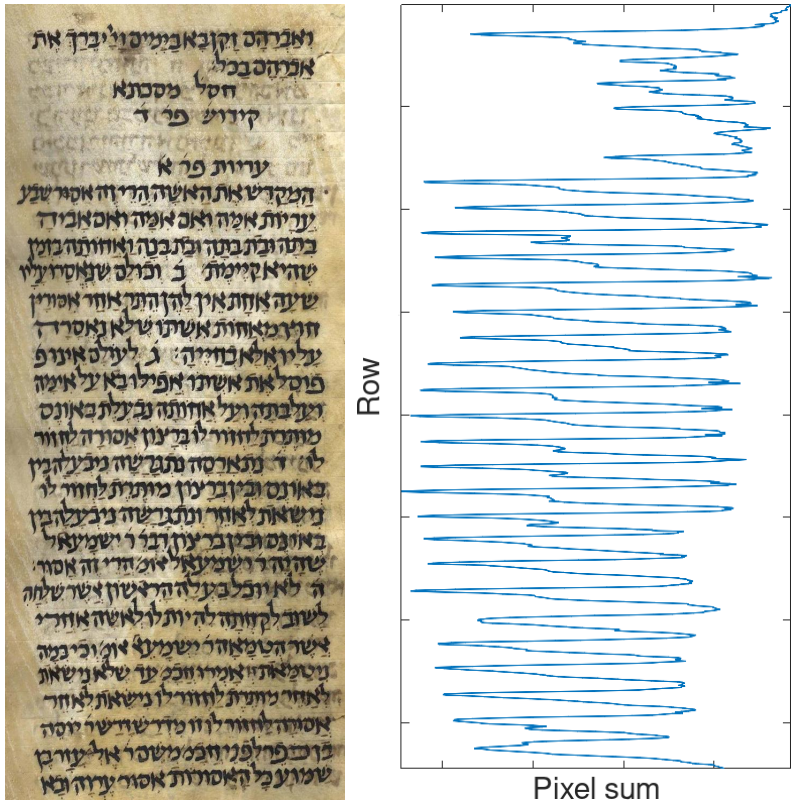
- Where do lines start?
- Where is the ink?

Sum rows of pixel costs to find text lines locations. Local minimas are typically between text lines.

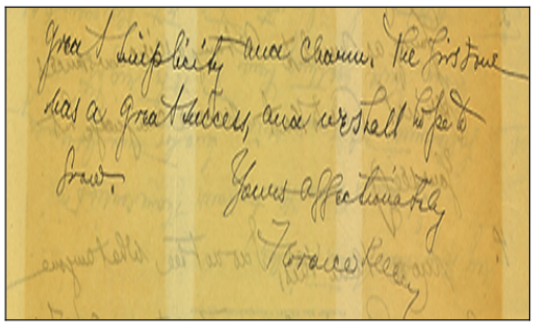
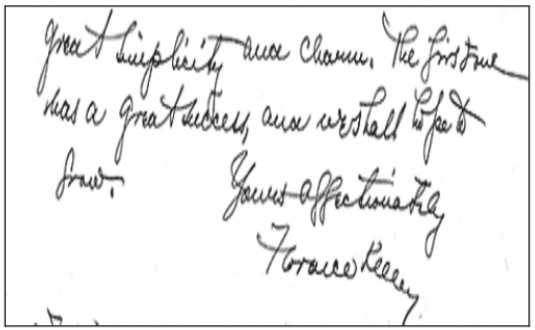
To find line start only, consider only the left side of the text block. You can carve from there to the right side.

Typically, line ends are also computed, and matched to the starts.

Finding line starts:



Finding the ink location is a challenging task. This is typically called “binarization”, and can be used to compute the pixels’ cost.

a)  b) 

(Bezmaternykh et al., 2019)

In this lecture, the considered documents are rather clean. While it can help to get better results, binarization will not be necessary. As cost function, use $255 - M$, where M is the mean value of the three channels of the image.

For motivated students: you can try Otsu and Sauvola, two common binarization methods :-)

Conclusion

Any question?