# Cosmos Segmentation

by Keshav Balaji

Balancing Margins

- Takes in input binary map and map of original image, and balance possibly unequal margins
- Two pointers (left margin at column i and right margin at column width - i)
  - Move each margin boundary inward while it contains only whitespace
- Checks absolute difference between left and right margin boundaries
  - If left margin width is less than right margin width, slices image and bmap to include only the columns from first column till the column at index width - diff
  - Otherwise, slices image and bmap to include only the columns from first column till the column at index width - diff

Balancing Margins Example

Before balancing margins: **width = 1166**
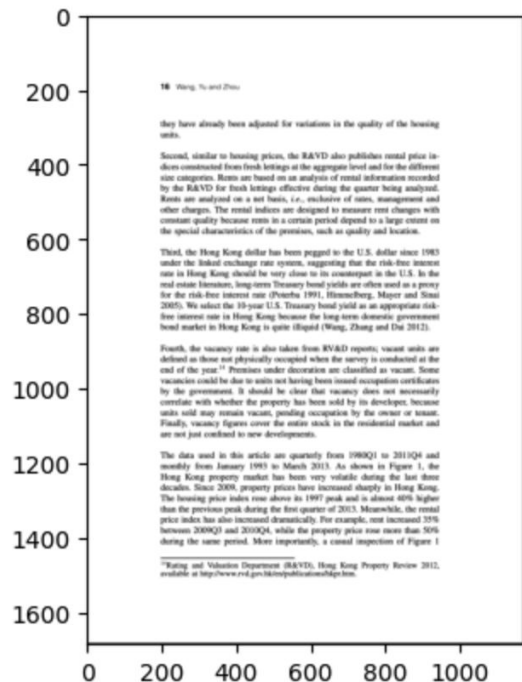
After balancing margins: **width = 1140**
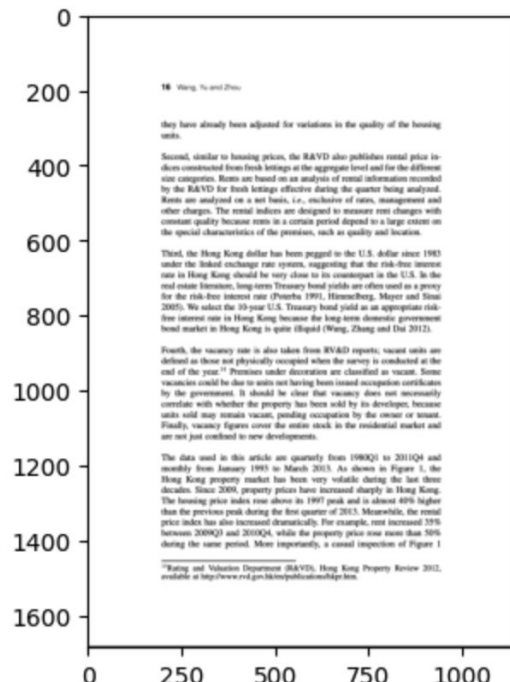
Get Blank Rows

- Objective: create list of blank rows that mark the end of a region (ie. the beginning of another region)
- Defines a blank row (array of 0's) of size img's width (# columns)
- Defines curr_top and curr_bot variables to track top and bottom of current row
- Iterates with curr_bot from row 0 to row img_height - 1 (top to bottom)
  - If current row (row from curr_top to curr_bot) is equal to the blank row:
    - If list is empty, adds curr row to list and set curr_top and curr_bot to next row
    - If row index of the last white row in the list is greater than the current row's top index, set the last white row's bottom index to curr_bot (finding white row closest to end of region)
    - Otherwise, append curr_bot to list
  - If curr_top is first row, or if curr_bot is last row, adds the row to the list

Get Blank Rows Example

Get Proposals

- Balances margins of image by calling balance_margins()
- Gets the row indices of each start of region with get_blank_rows()
- Stores bmap's horizontal slices (areas between consecutive starts of regions) in a list 'rows' (setting top boundary)
- Iterates through 'rows' list
  - Performs same white row detection on transpose of each row (column)
  - Sweeps sideways to get vertical slices of the horizontal slices, stores each vertical slice in a variable spl, whose transpose is then added to the list 'blocks'
  - Iterates through 'blocks' list, and adds its horizontal slices to list 'rows2'
    - Iterates through rows2 list, and establishing right and bottom boundary for each region with np.argwhere(), which retains only non-white pixels
    - Adjusts coordinates of final bounding box to page's absolute coordinates

Detecting White Rows (Visual Representation)
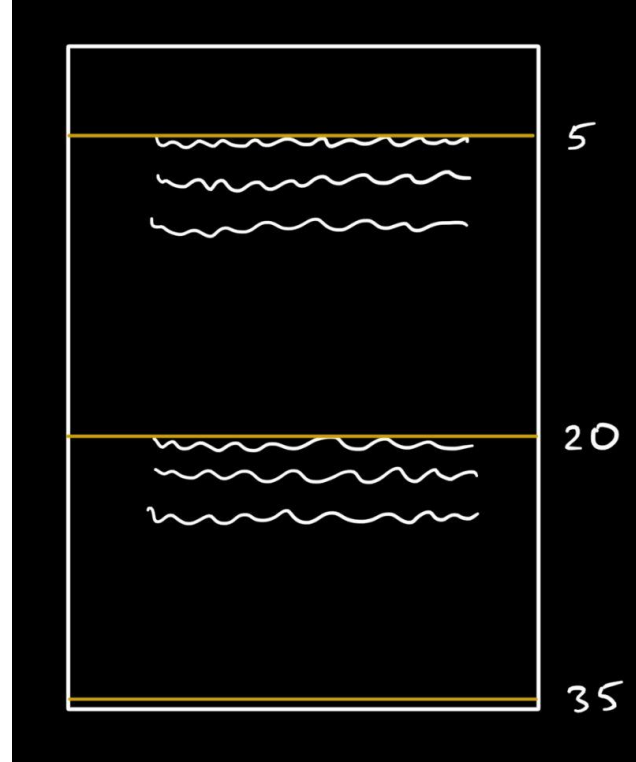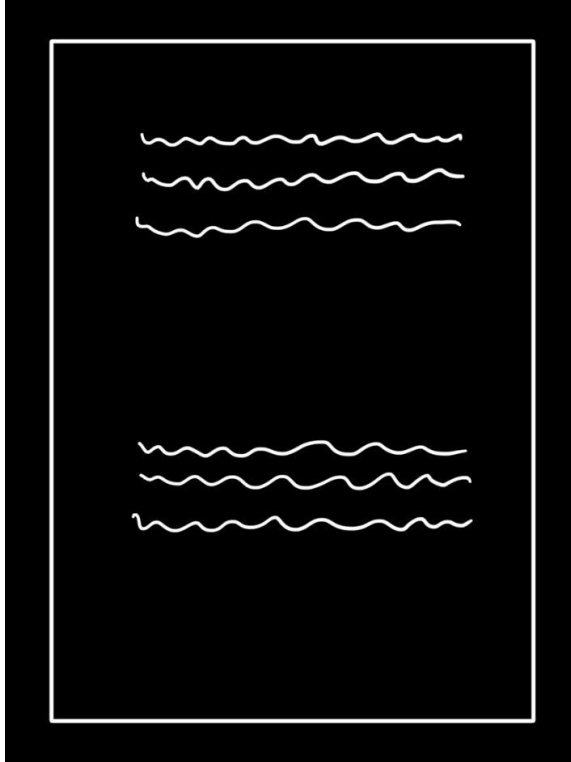
Forming 'rows' List
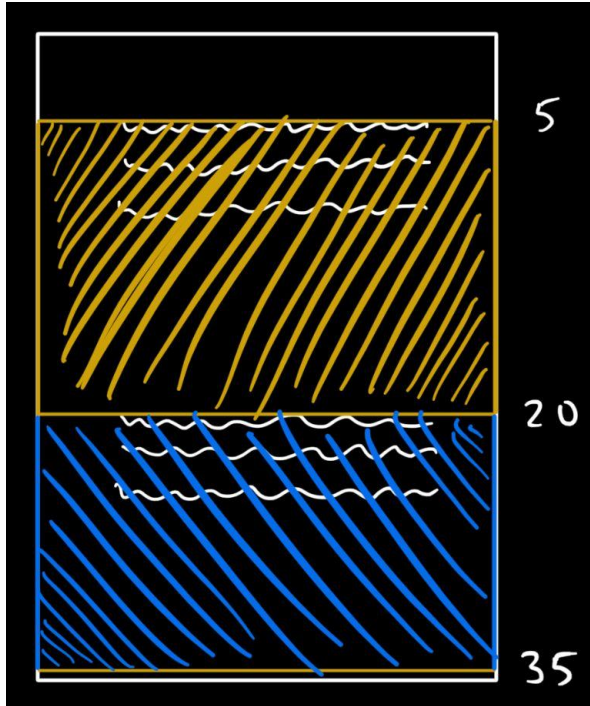
- Top boundary of regions set

rows = { slice from (5,20) , slice from (20,35) }

Column Detection

For now, let's focus on row[0] as row

<u>row.T : Visual representation</u>
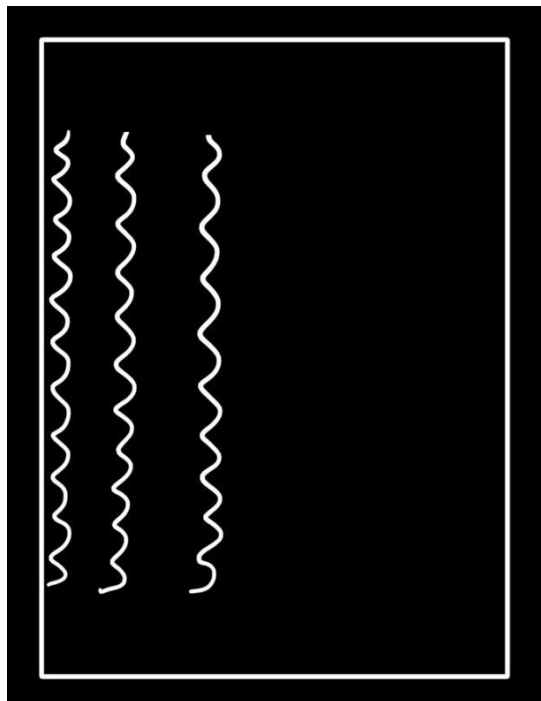
<u>row.T : Visual representation</u>

$$\text{row } T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ a & e & i & 0 & 0 \\ b & f & j & 0 & 0 \\ c & g & k & 0 & 0 \\ d & h & l & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$(6 \times 5)$$

Getting white_cols
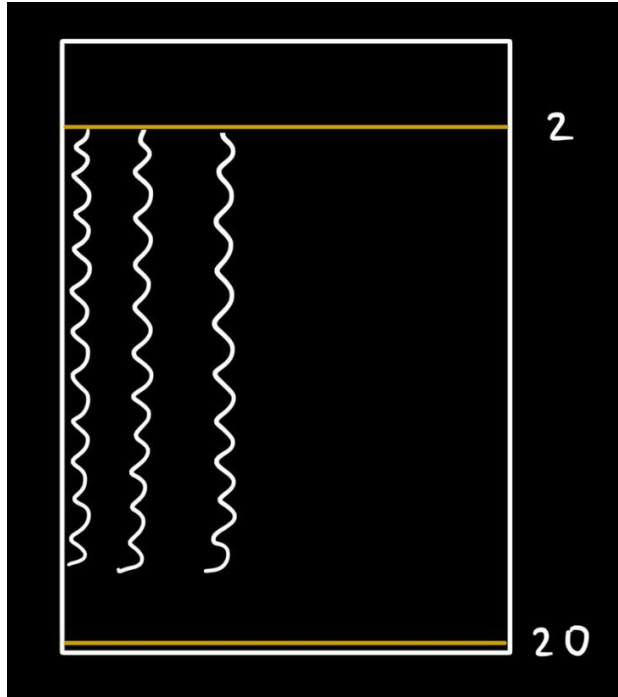
$$\text{row } T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ a & e & i & 0 & 0 \\ b & f & j & 0 & 0 \\ c & g & k & 0 & 0 \\ d & h & l & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(6×5)

Setting left boundary

$$Sp1 = \begin{bmatrix} a & e & i & 0 & 0 \\ b & f & j & 0 & 0 \\ c & g & k & 0 & 0 \\ d & h & l & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(5×5)
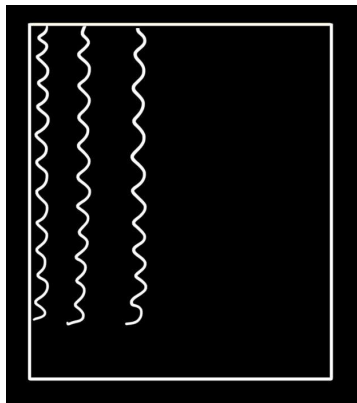
$$Sp1 = Sp1.T = \begin{bmatrix} a & b & c & d & 0 \\ e & f & g & h & 0 \\ i & j & k & l & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(5×5)

(left boundary set)
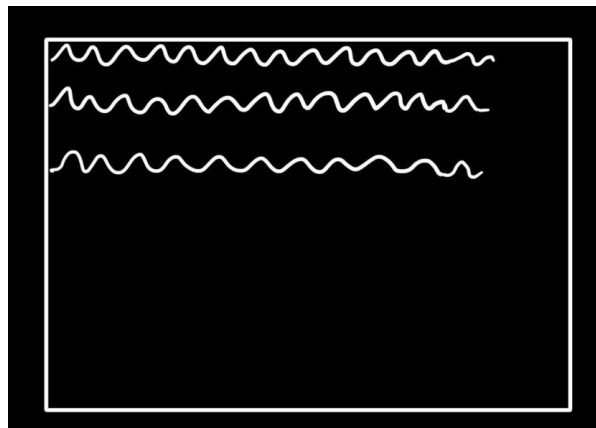
Finishing Bounding Box Creation

Adding to blocks and rows2 lists

$$\text{blocks} = \text{rows2} = \left\{ \begin{bmatrix} a & b & c & d & 0 \\ e & f & g & h & 0 \\ i & j & k & l & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right\}$$

Finishing Box Creation by Finding Nonzero Elements

<u>Finding Nonzero Elements with argwhere()</u>

$$\text{one\_inds} = \{ [0,0], [0,1], [0,2], [0,3], [1,0], [1,1], [1,2], [2,0], [2,1], [2,2], [2,3] \}$$

<u>Splitting coordinates by axis (row vs. col) using hsplit()</u>

$$\text{h\_one\_inds} = \{ \{ [0], [0], [0], [0], [1], [1], [1], [1], [2], [2], [2], [2] \},$$
$$\{ [0], [1], [2], [3], [0], [1], [2], [3] \} \}$$

Bounding Box Coordinate Generation

- Taking minimum x and y coordinates of nonzero elements for top left corner
- Taking maximum x and y coordinates of nonzero elements for bottom right corner

$$x1 = np.min(h\_one\_inds [1]) = 0$$
$$y1 = np.min(h\_one\_inds [0]) = 0$$
$$x2 = np.max(h\_one\_inds [1]) = 3$$
$$y2 = np.max(h\_one\_inds [0]) = 2$$

- Relative Coordinates

$(0,0)$ →

a	b	c	d
e	f	g	h
i	j	k	l

→ $(2,3)$