Image processing onramp

★ Identify, discovery, monitoring. (extract info, take actions), edge detection, Remove noise.

★ Everywhere!

Import → preprocess → Segment → postprocess → classify

★ work with images

★ Segmenting

★ pre, post processing techniques

★ classification & batch processing.

---

working with images

"Import - assign it to a variable"

A = imread ("img file.jpg"); → import

→ display

imshow(A);

Display two images together.

imshowpair (A1, A2, "montage");

↓

side by side (A1 - left, A2 - right)

Save it to a file

imwrite (imgdata, "img.jpg");

"export imgdata to "img.jpg"

Greyscale & colour images

★ numerical array (300 x 400)

0 to 255 → Intensity.

Colour - RGB - 3 intensities

even though 2d image - stored as 3d array.

Green plane alone — Red

I(:,:,2); — I(:,:,1);

* most images use uint8 data type

Rmax = max(R,[],"all") → Across all values
↳ unused

Rmin = min(R,[],"all");

Split Colors

[R,G,B] = imsplit(A);

'display all at once'

montage({R,G,B});

---

* Grayscale → occupies → 1/3 space than RGB image.
* Less computation - simple - straight forward. (Info lost)
* Receipt: (black & white) → preserved!

Convert to grayscale

gs = im2gray(A)

Now only 2 dimensions (600 × 400)

"Intensity values of gray scale": weighted sum of RGB plane.

Save: imwrite(gs, 'gs.jpg');

Adjusting Contrast

Normalizing brightness - important step in preprocessing (even in grayscale the contrast is different. Especially when identifying black & white patterns of text in receipt images. An intensity histogram separates pixel into bins based on their intensity values. eg: How many pixels binned in the low end of the histogram. Bright regions have pixels binned high of the histograms. The histogram often suggests where simple adjustments can be made to improve the definition of image features.

```
I = imread ("Img_001.jpg");
I2 = imread ("Img_002.jpg");
gs = imagray(I);
gs2 = imagray(I2);
imshowpair (gs, gs2, "montage");
```

Investigate contrast using histogram:

```
imhist (gs);
imhist (gs2);
```

Intensity of histogram of gs2 shows lower contrast b/w the text & backgro most of the dark pixels have intensity values below 100. and not many bright pixels have intensity above 200. meaning contrast is about half of what it could be if the image used the full intensity range (0 to 255)

Increasing image contrast brightens brighter pixels & darkens darker pixels. We can use "imadjust" function to adjust the contrast of a grayscale image automatically.

```
gs2 adj = imadust (gs2)
imshow pair ( gs2, gs2adj, "montage");
imhist (gs2adj);
```

Note: imadjust → only for grayscale images
imlocalbrighten → For color image.

```
gs2 = imlocalbrighten(I2);
imshow (gs2)
```

---

* imshow * imread * imhist
* size() * imadjust

Image viewer → View image
→ Change histogram interactively

Open:

>> imtool (on Apps)

* Load an image (See info)
* pixel region tool – Inspect RGB values

Image viewer – only allows to adjust the contrast of a grayscale image.

* Adjust left slider – darken pixels to augment
* Right slide – brighten pixels adjust.

```
gs2 = imread ("gs2.jpg");
Sunset = imread ("Sunset.jpg");
```

>>imtool

load – image info, pixel region, Adjust contrast, Crop, measure distance, magnify, pan, zoom in & out.

'can't adjust contrast' in Color image.

Segmenting text

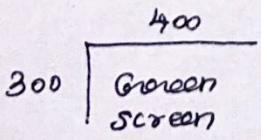
* Main feature of receipt images: darken text on a bright background!
* Leverage this!
* One method: seperate text from the background. Seperating an image into distinct part is called segmenting an image.

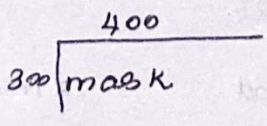
'Lanes, cars, Roads' – Edge detection, Text, more shapes, Colors, (boundary) size, etc..

* Idea = Identify the region of interest. (mask)
* Binary mask: Identifies the area of interest & replace with other images

↳ (Green Screen: Foreground (keep – img), background (green screen)).

↳ 'Logical array' indicates region of interest. Length & width of the mask are same as the original one





1 → keep
0 → Don't

---

* 'Replace the background'
* 'Identify objects', calculate.

'Create binary black & white img from a grayscale img by thresholding its intensity values. Values below the cutoff are assigned to 0, while others above are assigned to '1'.

eg: A grayscale image was segmented using a threshold of $\frac{1}{4}$ the maximum possible intensity of 255.

```
img = imread('IMG_001.jpg');
gs = im2gray(img);
gsAdj = imadjust(gs);
imshow(gsAdj);
```

'Setting threshold' (half)

```
BW = gsAdj > (1/2 * 255);
imshow(BW);
imhist(gsAdj);
```

Setting threshold (200)

```
BW = (gsAdj > 200);
imshow(BW);
```

Not good.

Repeating to 'Img_004.jpg' → good at $(\frac{1}{2} * 255)$.

Different lighting conditions: different contrast: different thresholds (needed)

* manually: impractical for 1000s of images

Automate!

eg: `gBinary = imbinarize(g);`

```
img = imread('Img_001.jpg');
gs = imagray(img);
gsAdj = imadjust(gs);
imshow(gsAdj);
```

'Text in the top - not clear as in bottom' — By default, imbinarize uses a global threshold value - the same threshold for every pixel.

'adaptive' → look at smaller portions & Choose the best threshold.

```
BWadapt = imbinarize(gsAdj, 'adaptive');
imshowpair(gsAdj, BWadapt);
```

'Not helped - worst - Text completely blacked out'

why?

By default; the foreground of an image is assumed to be bright & the

background dark. But does receipts - not true (foreground-dark text, bg: bright)

Set:

```
gBinary = imbinarize (g, "adaptive", "ForegroundPolarity", "dark");
```

Solution

```
BWadapt = imbinarize (gsAdj, "adaptive", 'ForegroundPolarity', 'dark');
imshowpair (gsAdj, BWadapt, 'montage');
```

Identifying text patterns

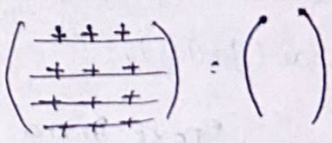
Improved the visibility of the text in a binary image. how to determine if the image is a receipt? → Answer lies in the pattern of text in the image.

* Rows of pixels having text have more 0 values & the rows b/w lines of texts have more 1s. If you sum the values across each rows, rows with text will have smaller textsums than rows without text.

```
I = imread ('Img_006.JPG');
gs = imagray(I);
gsAdj = imadjust(gs);
BW = imbinarize (gsAdj, 'adaptive', 'ForegroundPolarity', 'dark');
imshowpair (I, BW, 'montage');
```

'Img_005.JPG'

Sum across rows: 'Now 2d' - Grayscale. →(Columnwise answers) - Column vector.



```
S = sum (BW, 2);
plot(S);
```

```
S2 = sum(BW2, 2);
plot (S2);
```

Some receipt images have elements that distort the text patterns oscillations in the row sum plot, making them harder to classify.

Noise - making binary mask - appear as specs - distort

Remove noise:

Background: Large fluctuations in the row sum (distort: Smaller text pattern) - Removing background - make the text more noticible.

Pre- & post processing to improve Segmentation

'Noise removal'

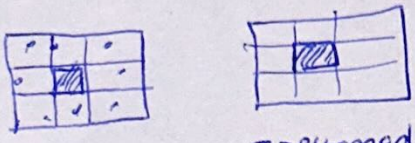
* Sometimes: No matter how we pick threshold - we can't seem to get binary image.
* Individual pixel has high intensity than neighbour pixels → Smoothing required.

'Spatial filtering' - Neighbourhood operation

* Each pixel changed - based on its neighbour.

Filter: acting like a mask - matrix - performs some operation.

eg: Linear filter: weighted average.



I

I Filtered

'move to next filter'

'sliding window operations'

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

→ Average of pixels & the eight pixels around it. — replaces each pixels with

'Removing noise & blurring edges' → useful.

You can increase the light sensitivity of a digital camera sensor to improve the brightness of a picture taken in low light. Many modern digital cameras (including mobile phone cameras) automatically increase the ISO (International Std. org - Standard). However this increase in sensitivity amplifies the noise picked up by the sensor, leaving image grainy.

```
I = imread ('Img_007.jpg')
gs = imread (I);
gs = imadjust (gs);
BW = imbinarize (gs, 'adaptive', 'Foreground polarity',
                                        'dark');
imshowpair (gs, BW, 'montage');
```

'Images taken in low light - often noisy due to ↑ in camera sensitivity used to capture the image' - polluting regions in the binarized img.

Reduce impact of noise — use Averaging filter.

* fspecial — n by n averaging filter. (creates)

H = fspecial ("average", 3);

Apply filter to an image

imfilter:

→ (Filter — averaging filter)

gsSmooth = imfilter (gs, H)

↳ Image (gray scale)

gsSmooth = imfilter (gs, H);

Binary image

BWSmooth = imbinarize (gssmooth, 'adaptive', 'foregroundPolarity', 'dark');

imshow (BWSmooth);

The filter introduced a small unwanted dark outline — to parts of the image border. (default: imfilter sets pixels outside the image to zero).

Adjust: 'replicate'

gsSmooth = imfilter (gs, H, 'replicate');

BWSmooth = imbinarize (gssmooth, 'adaptive', 'foreground polarity', 'dark');

imshow (BWSmooth);

Plot together: hold on, hold off

Isolating the image background

Receipt images that have a busy background are more difficult to classify because artifacts pollute the row sum. To mitigate this issue, you can isolate the background & then remove it by subtraction.

In a receipt - Background — Anything not text.

'Removing text' — Background

'morphological operations' — Remove text.

Morphological operations

* Type of sliding window operations (To look at neighbourhood)
* In linear filter - we change the value of the pixel by neighbourhood
* Here: change the output by changing the shape & size of the window

window - structuring element

"Shape is important - primarily two operations that can be performed"

max & min → Dilation & Erosion.

dilate (max):

Dilation takes the brightest pixel in the window

erosion (min)

Takes the darkest window

"These operations don't leave much room for customization" - To change the result - need to change the window.

Common: Dilation & erosion - performed in sequence rather than by themselves.

---

erosion followed by dilation : opening an image!

opening emphasizes & connects the dark parts of the image:

* Removes bright areas like spots, stripes (smaller & narrower than the structuring element.
* Leaves - ones that are longer / wider.

Reverse order (dilate → erode)

* closing an image! (opposite effect from opening).

* closing emphasizes bright parts & connects the bright parts
* Removes smaller / narrow dark areas
* Looking closely - we can observe the structuring elements pattern.
* Changing structural element - pattern also changes!

* making structural element smaller - we can see more details of the image. pattern becomes smaller.

Create structuring element

```
SE = strel ('diamond', 5)
                    ↳ size.
```

disc sized (8 - radius)

```
SE = strel ('disk', 8);
```

Perform closing operation

dilate → erode

```
Ibg = imclose (gs, SE) ;
          ↳ structural element
      ↳ Img
```

Remove noise - subtract background

```
Isub = I2 - I1 ;

gsIsub = gs - Ibg ; → sub background from
                      img.
imshow (gssub);
```

(or) subtract image from background

```
gsSub = Ibg - gs;
imshow (gsSub);
```

(gs - Ibg) → fully black

('No use')

Noise remove: Noise is due to (distortion) background image not useful. So we are closing the image (emphasize colour brighter parts) then subtract with original version.

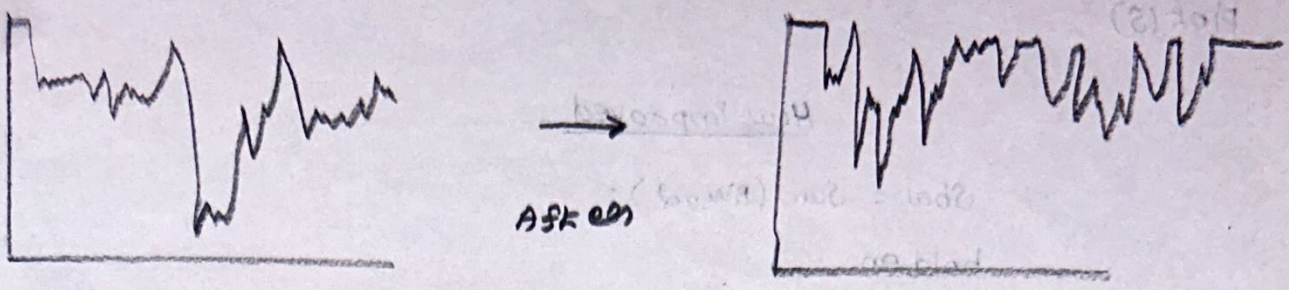
```
BWSub = imbinarize (gsSub);
BWSub = ~BWSub;
imshow (BWsub);
```

Inverting a binary image

1 → 0

0 → 1

Using noise removal use



'Before'

'Composing' - distortion - Reduced.

'we can use morphological operations to enhance the text in binary image & improve the row sum signal.

'Morphological opening - expands the dark text regions while closing - diminishes the dark text.

Increasing size of structural element ↑ these effects'.

To enhance a particular shape in an image, create a structural element in the shape of what you want to enhance. each row of text - should become a horizontal stripe (short, wide rectangle).

```
SE = strel ('rectangle', [10, 20]);
```

'Rectangle - short enough - to preserve the white space b/w rows of text is preserved - wide enough to connect adjacent letters'.

```
SE = ('rectangle', [3  25]);
```

Isolate background:

'Eliminate dark text by emphasizing bright regions'

'Job done closing' , emphasize dark - open

```
Iopened = imopen (I, SE);

BWstripes = imopen (BW, SE);
Im show (BWstripes);
```

S = Sum (BWstripes, 2);

→ Rowwise Sum.

Plot (S)

How Improved

Sbw = Sum (BW, 2);

hold on

Plot (S, 'b');

Plot (Sbw, 'r');

hold off

'Developing a metric for Receipt detection'

Local maxima: Live editor: task ⇒ 'tool box'

Prominence window: 25

Adjust min. prominence: Until 9 minima = 71.

(we don't need shallow minima) → we need exact ones.

Identify deep valleys in image 3

'minima'

---

* Receipt Classification algorithm: Count no. of minima in row sum
* Cut off value - seperates - Receipts from non-receipts.

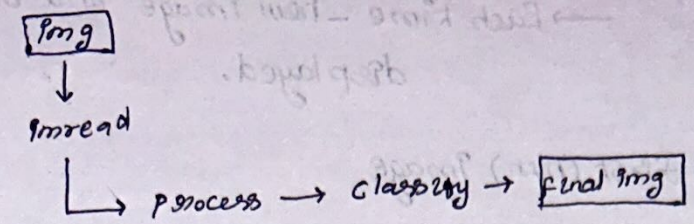
nnz → Finds number of (minimum) non-zero elements

nnz ([8 0 7 5 3 0]) → 4 (non-zero elements)

≥ 9 → 1 else 0

isReceipt = Min >= 9;

Batch processing with Image datastores



'Great for one / two images'

In practice: 1000's pictures

'Data store'

'Matlab variables - that acts as a reference to a data source'. (folders of image files). When you create an image datastore - matlab looks at the images & stores some basic meta information about them. 'Name, formats' - Not data → It imports when we need them. (Indiv / whole).

∴ while processing we can read from the datastore one at a time. Each time load a new file - we won't end up with a hundred images in MATLAB workspace.

create datastore - read in a loop & process

* create datastore: Image Datastore

Access datastores properties

* ds. Folders → access folders property

datafilenames = ds. Files → file names of all files as a cell.

Count number of images

nFiles = numel (datafilenames);

Read all images

readall (ds)

Read image by image

```
img = read(ds);
imshow(img);
```

→ Each time - new image will be displayed.

Read first (nth) image

```
img = readimage(ds, 1);
imshow(img);
```

classification - Algorithm

```
% Receipt = classifyImage(I)
```

→ User defined.

```
function %Receipt = classifyImage(I)
% This function processes an image - using algorithm - developed prev -
gs = im2gray(I);
gs = imadjust(gs);
mask = fspecial('average', 3);
gsSmooth = imfilter(gs, mask, 'replicate');
```

→ grayscale then filter it.

```
SE = Strel('disk', 8);
Ibg = imclose(gsSmooth, SE);
IbgSub = Ibg - gsSmooth;
Ibw = imbinarize(IbgSub);
SE = Strel('rectangle', [3, 25]);
Stripes = imopen(Ibw, SE);
Signal = Sum(stripes, 2);
```

→ Noise Cancellation

```
% Classification
minIndices = islocalmin(signal, 'MinProminence', 70, 'ProminenceWindow', 25);
Nmin = nnz(minIndices);
%Receipt = Nmin >= 9;
```

end.

```
ds = imageDatastore('test images');
nFiles = numel(ds.Files);
isReceipt = false(1, nFiles);

for k = 1:nFiles
    I = readimage(ds, k);                    → loop all files.
    isReceipt(k) = classifyImage(I)
end.
```

isReceipt = [0 1 1 0 0 1 ...]

↓ Receipt ↓ Not a receipt.

ds.Files(isReceipt) → gives the name of the files (img) which are receipts.

ds.Files([0 1 1]) → 2, 3 image alone displayed.

Display all images classified as Receipts

montage(receiptFiles)

∴ receiptFiles = ds.Files(isReceipt)