Calibration of beam size monitors

Scott Williams

April 9, 2020

Introduction

- Working on calibration of optical system used in beam size monitors using USAF1951 microscope resolution test chart, and repurposing Canon/Tamron EF mount lens' autofocus system
- Software component –
  - Detecting and measuring elements of the microscope test chart
  - Calculating resolution by reference to the smallest element of test chart seen
  - Calculating relation between pixels and microns on test chart plane by reference to largest element of test chart
- Hardware component –
  - Using an Arduino, emulate a Canon camera body and instruct a Tamron telephoto lens to adjust focusing position
  - Delayed due to COVID-19

Pictures of USAF1951

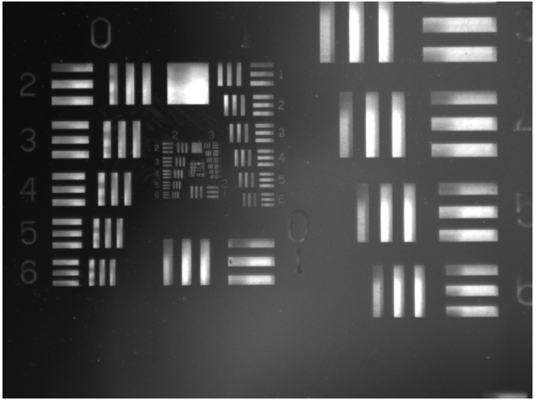


Figure 1: Example of USAF1951, close up

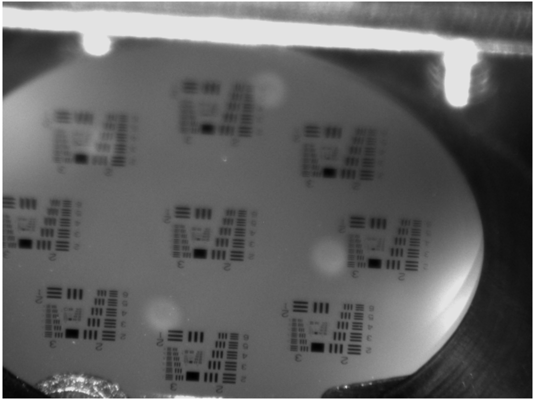


Figure 2: Image as seen on camera

A set of three equally sized vertical and horizontal bars forms an *element*, numbered 1-6. A set of six elements in sequence forms a *group*. In the figure on the left, five elements of group are on the far left.

Software part

Software component

Basic process:

- Find Regions Of Interest (ROIs) in an image that should analysed
- Analyse each region of interest, finding the smallest and largest set of bars
- Calculate the resolution/resolving power of the optical system (in line pairs per mm) by finding smallest group and element
- Calculate the transformation from pixel space to geometrical space on the microscope target
- Compare ROIs against and reject those with unreasonable values (eg. Too small, too large, not enough components detected, etc)
- Write to file calibration data

ROI detection

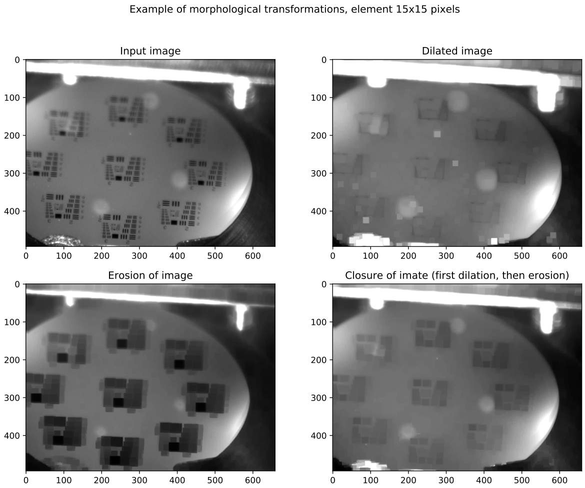
- In the current camera setup we can have one or more targets clearly visible, and some not visible at all.
- Can also be dealing with uneven illumination, depth of field effects, etc
- Previously found regions of interest by looking and clustering for hotspots of gradients, square shaped objects, but this can easily require too much fine tuning.
- Readapted method proposed for barcodes using morphological image transformations, which are similar to image convolutions, but using logical operations instead of multiplication and summation. See Katona and Nyúl 2012

Morphological transforms

In morphological transforms we take an image and a *structuring element* which serves as a mask to define the window to operate on. However instead of multiplying element wise and taking a sum as in convolution with an image mask, we'll do logical operations. Some examples are:

- Dilation: For each pixel in the original image, place the window on top. The pixel is set to the maximum value within the window. This tends to expand white parts of the image
- Erosion: Instead of taking the maximum, take the minimum. This tends to shrink the white parts/expand the black parts.
- Closure: Dilation followed by erosion, which tends to fill in dark gaps of the image smaller than the structuring element.
- Black hat: Difference of closure and original image. This tends to highlight black elements of image that are smaller than the structuring element.

Morphological operations



Example of morphological transformations, element 15x15 pixels

Input image

Dilated image

Erosion of image

Closure of imate (first dilation, then erosion)

Figure 3

Morphological operations

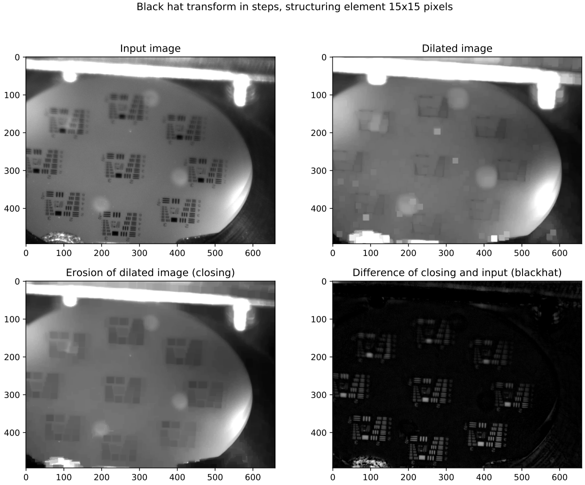

Black hat transform in steps, structuring element 15x15 pixels
Input image
Dilated image
Erosion of dilated image (closing)
Difference of closing and input (blackhat)
0
100
200
300
400
500
600

Figure 4

Morphological operations

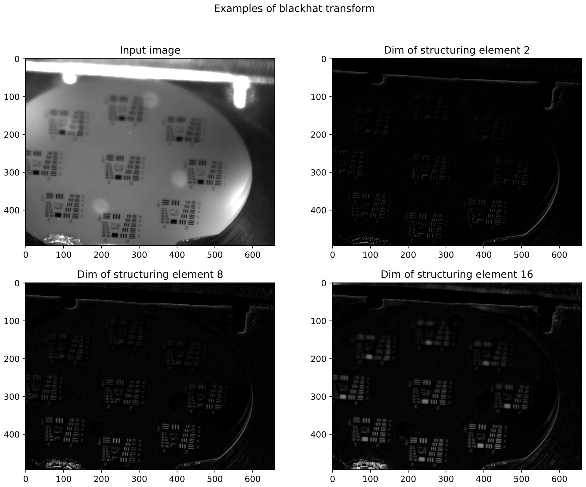
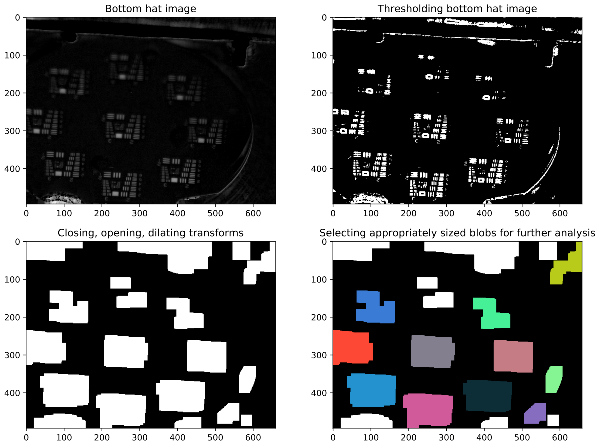

Examples of blackhat transform
Input image
Dim of structuring element 2
Dim of structuring element 8
Dim of structuring element 16
0
100
200
300
400
500
600

Figure 5

ROI detection

Overall algorithm flow



Bottom hat image

Thresholding bottom hat image

Closing, opening, dilating transforms

Selecting appropriately sized blobs for further analysis

Figure 6: Successive morphological operations are used to discover regions of interest, and then we investigate those that are roughly square and a reasonable size

ROI detection

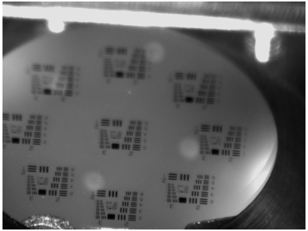


Figure 7: Input image

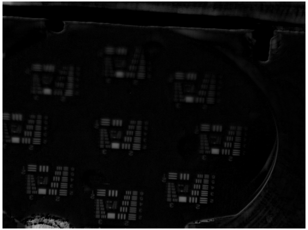


Figure 8: Bottom hat transformed image

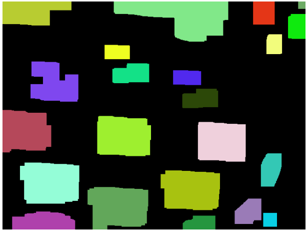


Figure 9: After repeated morphological transforms

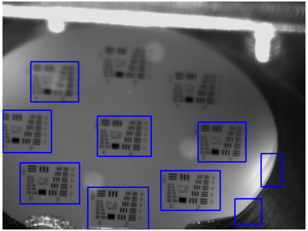


Figure 10: Detected regions of interest

ROI analysis

Each set of bars is manufactured to be a specific width and length, and these dimensions can be looked up in a table given the group and element number. To analyse the ROI we:

- Find the smallest paired set of three bars possible; this indicates the resolution/smallest features clearly distinguishable.
- Find the largest paired set of three bars possible; this is used to calculate the pixel/μm correspondance.

ROI analysis - graphically

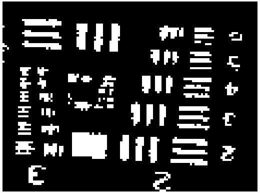


Figure 11: Binarise the ROI

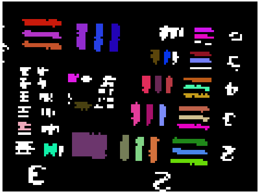


Figure 12: Detect rectangular shapes

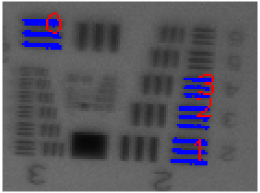


Figure 13: Cluster groups of 3 similarly sized horizontal rectangles

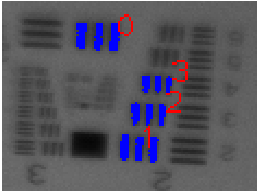


Figure 14: Cluster groups of 3 similarly sized vertical rectangles

Other comments

- Current the resolution/resolving power of the system appears to be limited by the resolution/number of pixels of the image. The bars are only a couple of pixels wide in many cases, which makes precise measurement of width and orientation difficult.
- The modulation transfer function, which describes the ability of the system to transfer contrast at various resolutions, is found by measuring the contrast of each element, and plotting it as a function of the resolution of each element. In this case the contrast is the maximum pixel intensity minus the minimum pixel intensity, and the line pairs per mm of the element. Unfortunately due to the size of the target in the image the number of datapionts is very small, ie. often at most 3-4
- Often many of the group and element number symbols are hard to read or distinguish due to lighting and resolution. Thus the group and element numbers associated with the sets of bars are deduced by counting down from the largest group ane element present, which is preset.

Still to be done

- Test with other cameras, lighting configurations, other configurations of aperture/lens zoom/lens focus
- Integrate with rest of control room systems and Tango interface
- Calculate/measure variance of results of script
- Add more documentation

Hardware part

Hardware component

- Using an Arduino and bitbashing (manually toggling a pin to communicate each bit of message), we will emulate a Canon camera body that gives instructions to an autofocusing Tamron telephoto lens
- Currently we can query the lens for lens info, serial number, and issue commands to move the focusing position backwards and forwards.
- Have also developed short Arduino program for controlling via sending bytes over USB
- Testing delayed, implementation of autofocus routine delayed due to COVID-19
- Still to do, implement autofocus routine

Current status and future plans of hardware part

- Currently the lens and Arduino are set up in the basement of building 208, under lockdown.
- Could replicate this in Australia at home, currently investigating whether similar lens could be found locally.
- Still need to implement a software autofocus algorithm and test this with the lens.

End of presentation

References I

Katona, Melinda and László G Nyúl (2012). “A novel method for accurate and efficient barcode detection with morphological operations”. In: *2012 Eighth International Conference on Signal Image Technology and Internet Based Systems*. IEEE, pp. 307–314.