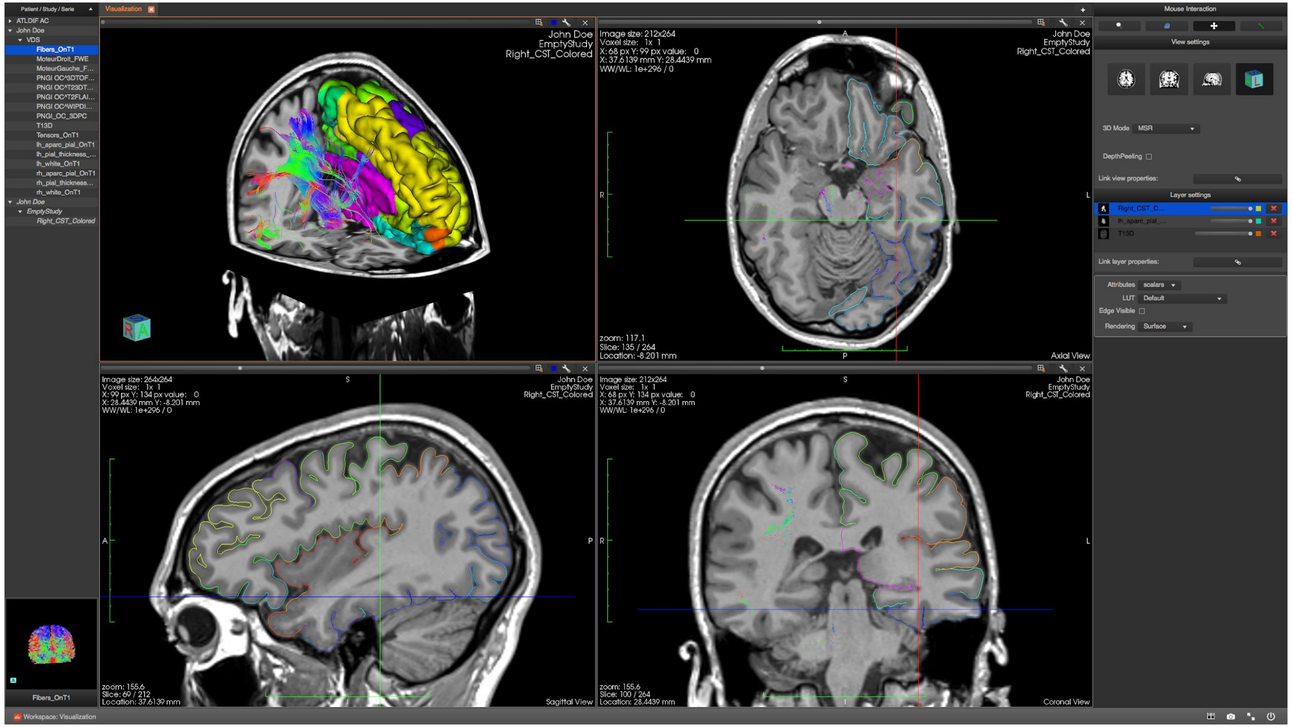
Developing medical image analysis tools in Python with Scikit-image

Frank Longford

PyData Cambridge, 15th November 2019

Medical Image Analysis



John Doe
EmptyStudy
Right_CST_Colored

Image size: 212x264
Voxel size: 1x 1
X: 68 px Y: 99 px value: 0
X: 37.6139 mm Y: 28.4439 mm
WW/WL: 1e+296 / 0
zoom: 117.1
Slice: 135 / 264
Location: -8.201 mm

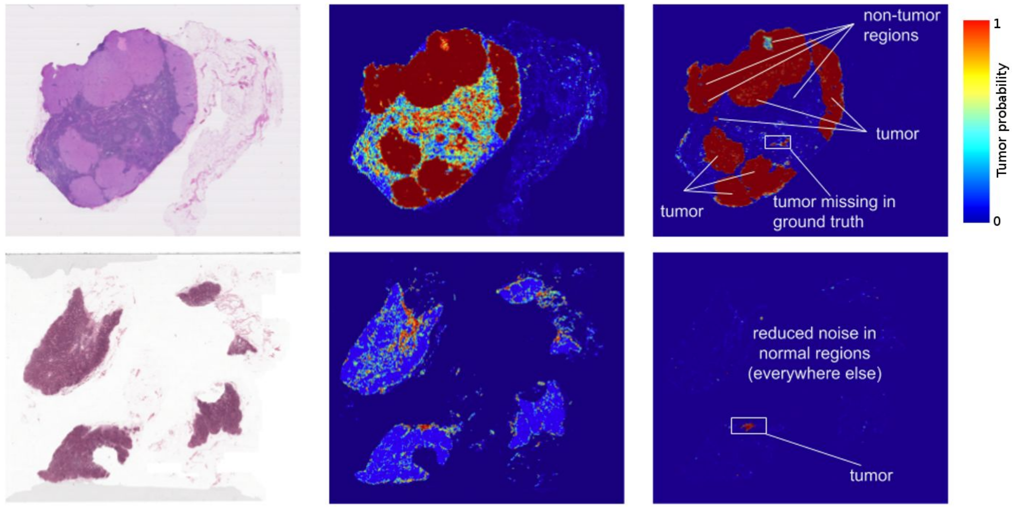
Image size: 264x264
Voxel size: 1x 1
X: 99 px Y: 134 px value: 0
X: 28.4439 mm Y: -8.201 mm
WW/WL: 1e+296 / 0
zoom: 155.6
Slice: 69 / 212
Location: 37.6139 mm

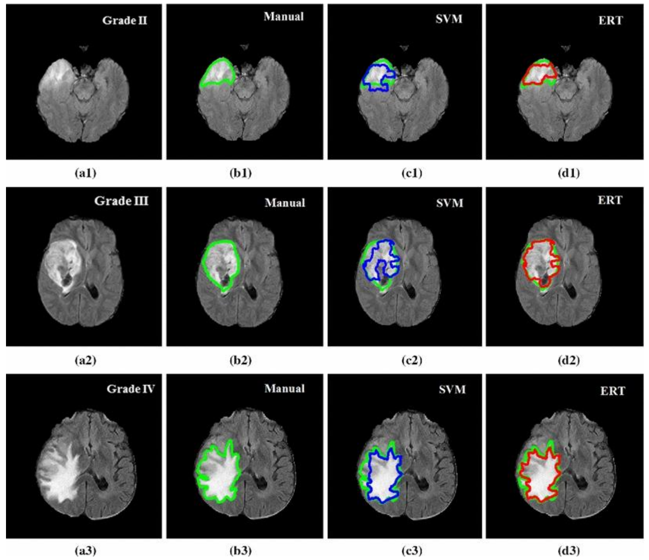
Image size: 212x264
Voxel size: 1x 1
X: 68 px Y: 134 px value: 0
X: 37.6139 mm Y: -8.201 mm
WW/WL: 1e+296 / 0
zoom: 155.6
Slice: 100 / 264
Location: 28.4439 mm

3D Mode: MSR
DepthPeeling
Link view properties:
Link layer properties:
Attributes: scalars
LUT: Default
Edge Visible
Rendering: Surface

https://team.inria.fr/empenn/files/2019/02/medInria-visu.png

Automated / Augmented Analysis

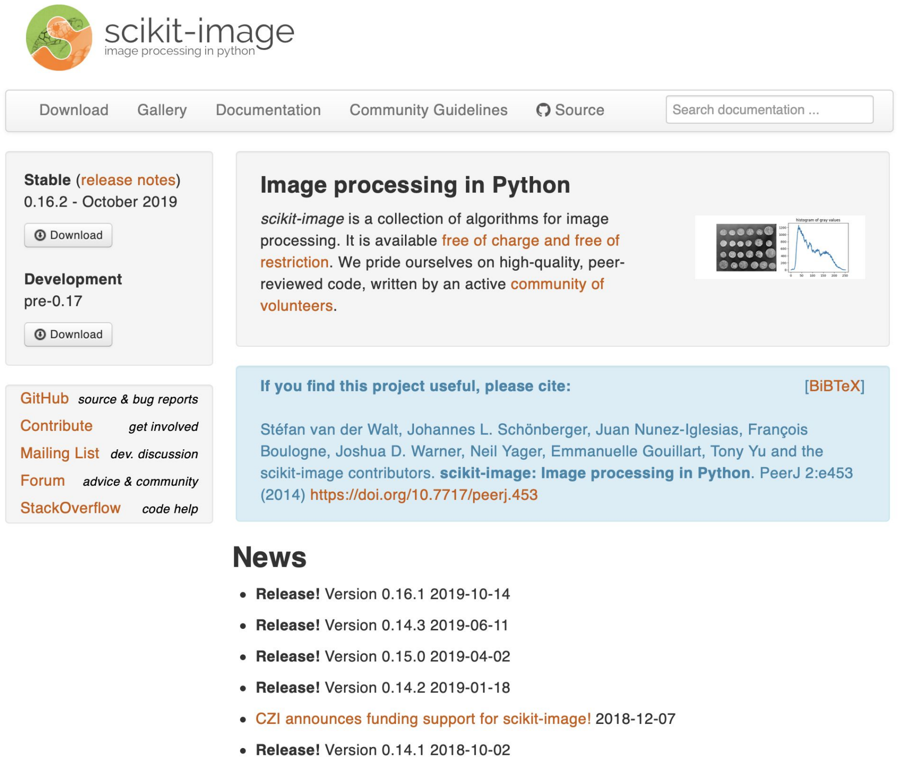

non-tumor regions
tumor
tumor
tumor missing in ground truth
1
Tumor probability
0
reduced noise in normal regions (everywhere else)
tumor



https://ai.googleblog.com/2017/03/assisting-pathologists-in-detecting.html
Soltaninejad, M., Yang, G., Lambrou, T. et al. *Int J CARS* (2017) **12**: 183. https://doi.org/10.1007/s11548-016-1483-3

Scikit-image

- SciKits add-on package
- NumPy / SciPy backbone
- Open-source (> 330 contributors)
- Well documented API
- Regular updates



Stable (release notes)
0.16.2 - October 2019
Download

Development
pre-0.17
Download

GitHub *source & bug reports*
Contribute *get involved*
Mailing List *dev. discussion*
Forum *advice & community*
StackOverflow *code help*

Image processing in Python

scikit-image is a collection of algorithms for image processing. It is available free of charge and free of restriction. We pride ourselves on high-quality, peer-reviewed code, written by an active community of volunteers.

If you find this project useful, please cite: [BiBTeX]

Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. **scikit-image: Image processing in Python**. PeerJ 2:e453 (2014) https://doi.org/10.7717/peerj.453

News

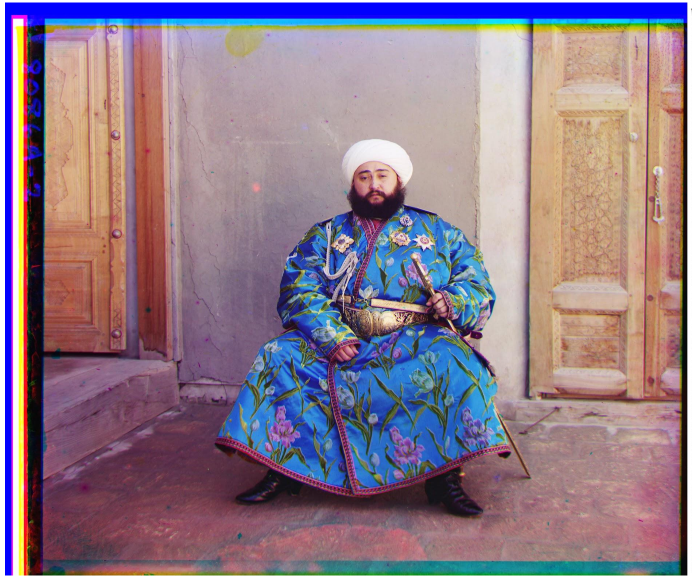
- **Release!** Version 0.16.1 2019-10-14
- **Release!** Version 0.14.3 2019-06-11
- **Release!** Version 0.15.0 2019-04-02
- **Release!** Version 0.14.2 2019-01-18
- CZI announces funding support for scikit-image! 2018-12-07
- **Release!** Version 0.14.1 2018-10-02

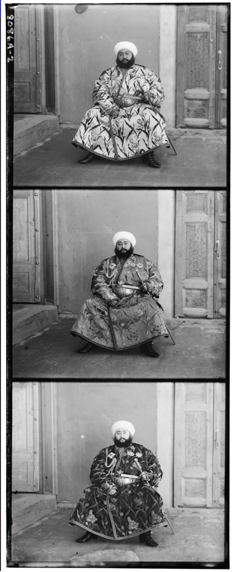
Image Analysis 101

ENTHOUGHT

Images are Vectors

$f(x, y, c)$





Blue $b(x, y)$

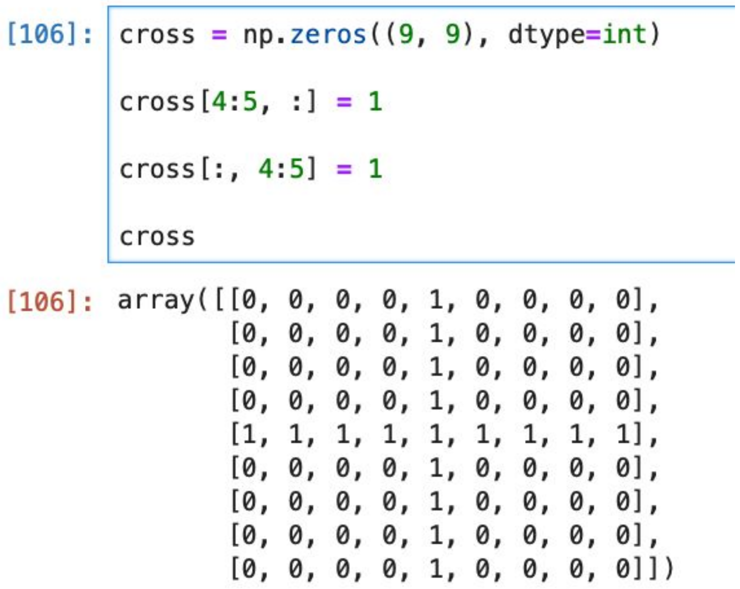
Green $g(x, y)$

Red $r(x, y)$

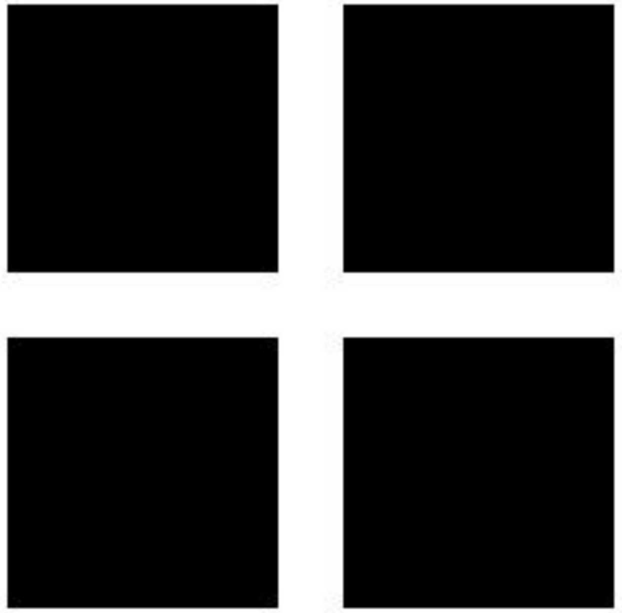
By Sergei Prokudin-Gorskii - Taken from the Library of Congress' website and converted from TIFF to PNG.TIFF file from LOC, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1470606

ENTHOUGHT

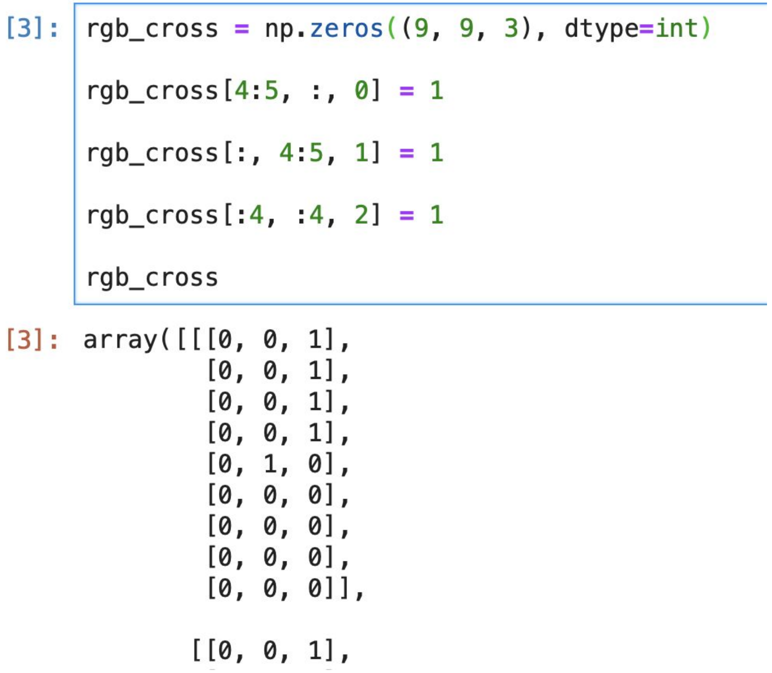
Digital Images are Arrays



```
[106]: cross = np.zeros((9, 9), dtype=int)

       cross[4:5, :] = 1

       cross[:, 4:5] = 1

       cross

[106]: array([[0, 0, 0, 0, 1, 0, 0, 0, 0],
              [0, 0, 0, 0, 1, 0, 0, 0, 0],
              [0, 0, 0, 0, 1, 0, 0, 0, 0],
              [0, 0, 0, 0, 1, 0, 0, 0, 0],
              [1, 1, 1, 1, 1, 1, 1, 1, 1],
              [0, 0, 0, 0, 1, 0, 0, 0, 0],
              [0, 0, 0, 0, 1, 0, 0, 0, 0],
              [0, 0, 0, 0, 1, 0, 0, 0, 0],
              [0, 0, 0, 0, 1, 0, 0, 0, 0]])
```

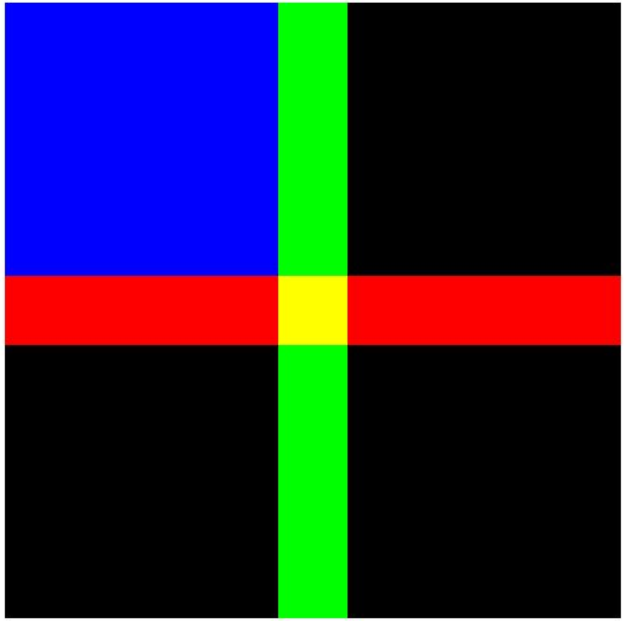


Digital Images are Arrays



```
[3]: rgb_cross = np.zeros((9, 9, 3), dtype=int)

     rgb_cross[4:5, :, 0] = 1

     rgb_cross[:, 4:5, 1] = 1

     rgb_cross[:4, :4, 2] = 1

     rgb_cross
```

```
[3]: array([[[0, 0, 1],
             [0, 0, 1],
             [0, 0, 1],
             [0, 0, 1],
             [0, 1, 0],
             [0, 0, 0],
             [0, 0, 0],
             [0, 0, 0],
             [0, 0, 0]],

            [[0, 0, 1],
```



A Note on Representations...

Hexadecimal format: #FF0000

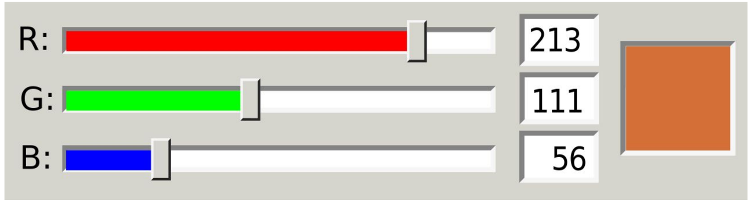
Contain 256 possible values for each channel

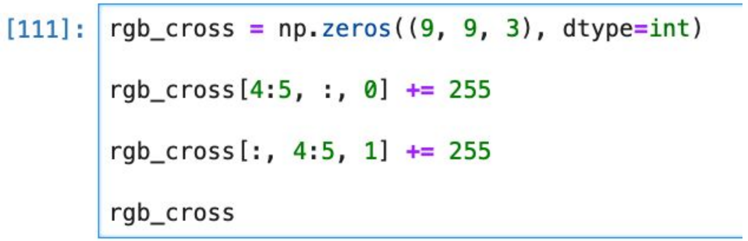
8-bit representation (2^8 permutations)

Scikit-image expected following RGB formats:

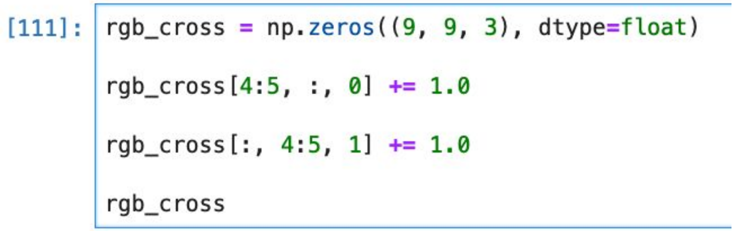
- Integer arrays must be in 8-bit format (0.. 255)
- Floating point arrays must be normalised (0..1)

RGBA (alpha = opacity) formats are also acceptable





```
[111]: rgb_cross = np.zeros((9, 9, 3), dtype=int)

       rgb_cross[4:5, :, 0] += 255

       rgb_cross[:, 4:5, 1] += 255

       rgb_cross
```



```
[111]: rgb_cross = np.zeros((9, 9, 3), dtype=float)

       rgb_cross[4:5, :, 0] += 1.0

       rgb_cross[:, 4:5, 1] += 1.0

       rgb_cross
```

"Not a snapshot, but an artistic pixel art representation.", Public Domain, https://en.wikipedia.org/w/index.php?curid=27756679

A Note on Reality...

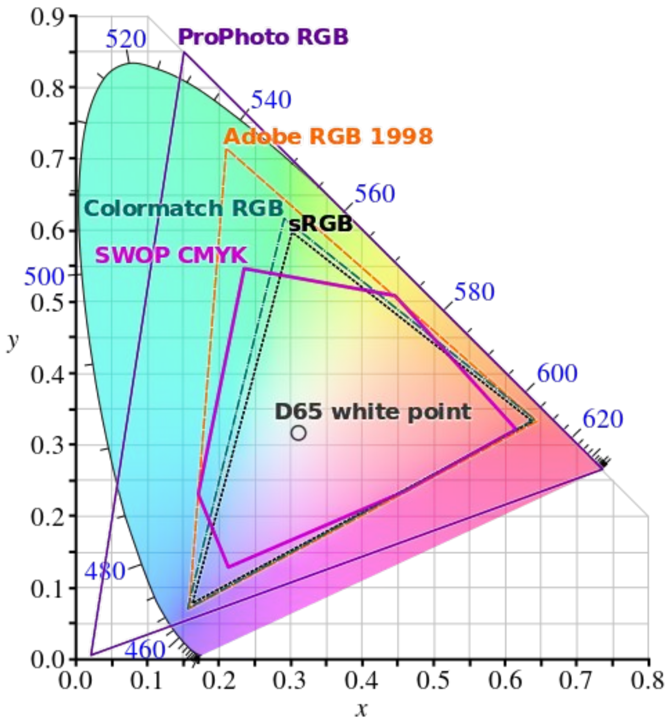
How to convert RGB units to colour?

Need a RGB Colour Space

1. Define the gamut
   - A complete subset of colours

1. Define mapping to wavelengths
   - CIE 1931 colour space standard

1. Define the white point
   - Sets chromaticity for (1, 1, 1)



Comparison of some RGB and CMYK colour gamut on a CIE 1931 xy chromaticity diagram, based on http://commons.wikimedia.org/wiki/File:CIE1931xy_blank.svg

Confidential and Proprietary

ENTHOUGHT