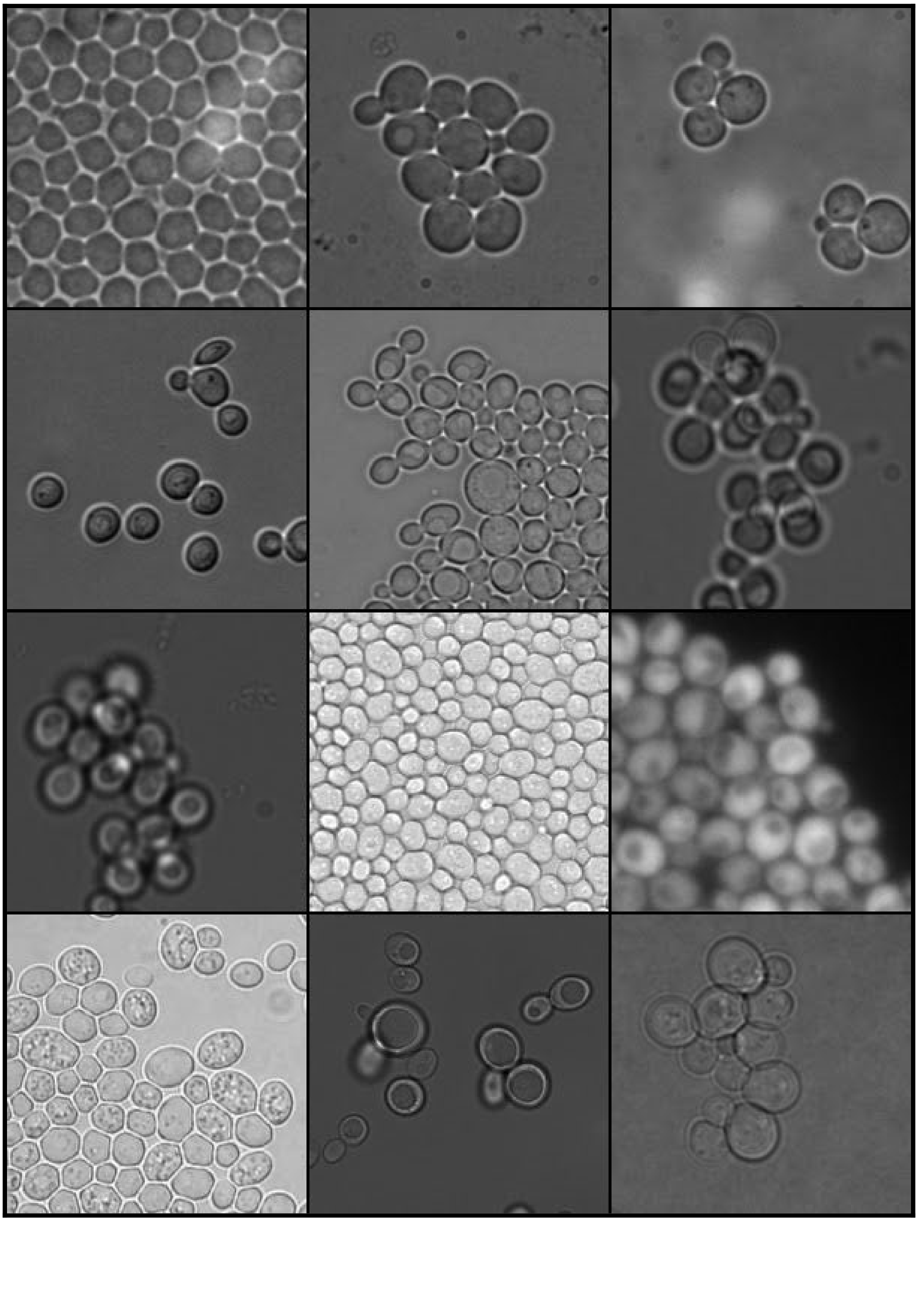
CellStar example usage guide v.1.9

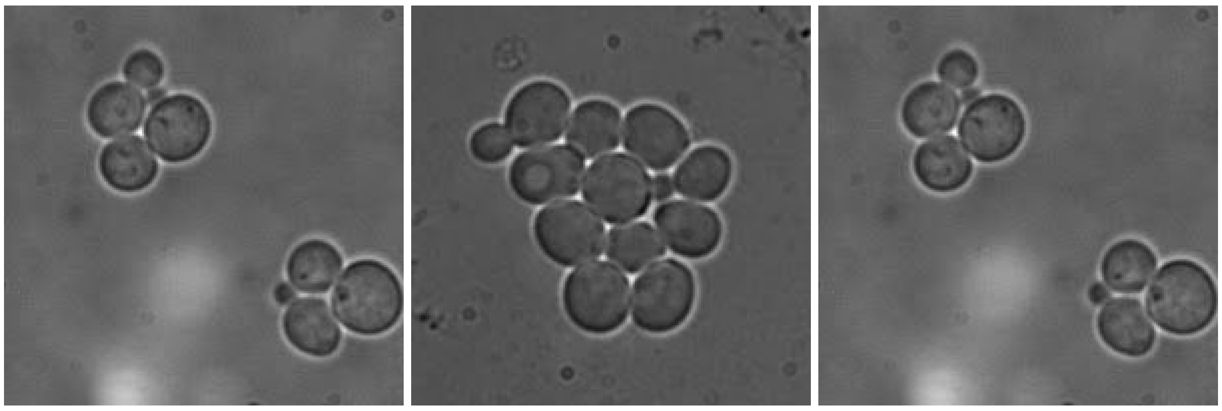


Introduction

The purpose of this document is to present in a convenient way all the imagery that has been so far properly segmented (to various extent) by CellStar algorithm combine with the power of CellProfiler. It complements the **Examples** folder found in **CellStar package** as most of the description can be also found in **Readme** files in the corresponding example folder. The pipelines that are refered to in the description can also be found in these folders.

More documents can be found in _Manuals and _Analyses folders in Examples folder.

1. Strong edges and dark interiors



Input imagery features:

In these images cells have distinctive bright edges and their interiors are significantly darker than the background.

Initially CellStar was designed to handle these type of images so no specific configuration is necessary. However in some cases it may be beneficial to increase number of random seeds to 1.5.

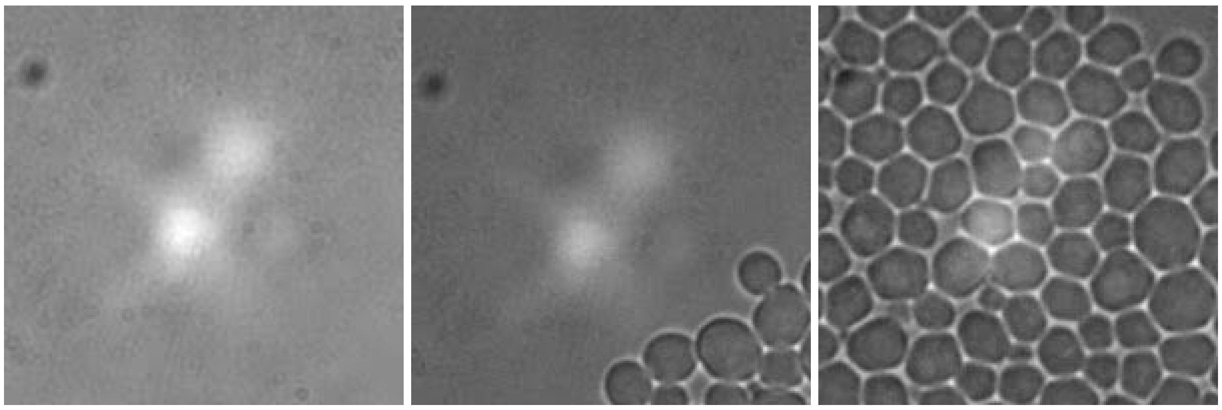
Pipeline:

1. LoadImage
2. IdentifyYeastCells
   - default parameters
   - precision 2

For ready to use pipeline go to:

- **1_StrongEdgesAndDark/StrongEdgesAndDark.cpproj**

2. Inhomogeneous background



Input imagery features:

As in Example Usage 1 cells have distinctive bright edges and their interiors are significantly darker than the background.

The only difference is that background is inhomogeneous and can therefore interfere with segmentation. In this case the cells that lie on the "spot light" are missed by the pipeline from Example Usage 1.

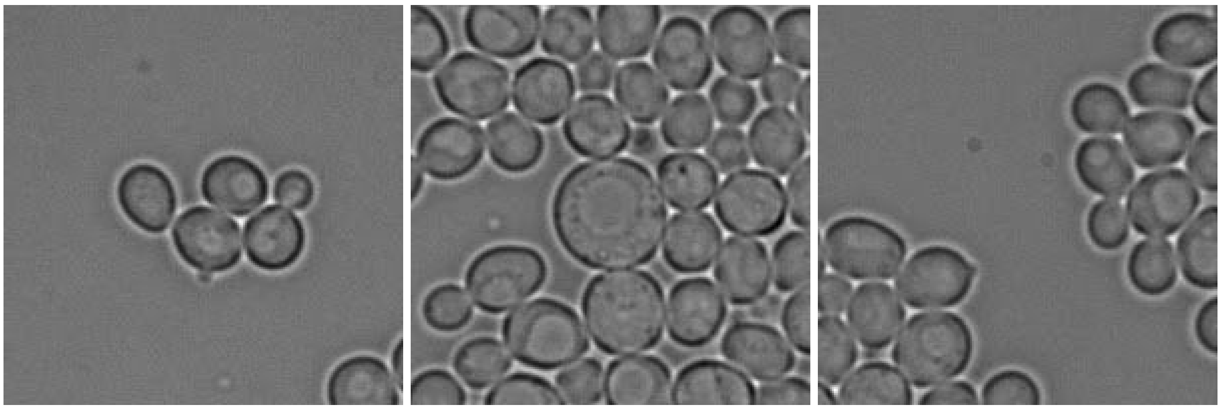
In order to fix that problem we can provide background image which will improve segmentation quality.

Pipeline:

1. LoadImage
   - Input
   - Single background for all other input
2. IdentifyYeastCells
   - default parameters
   - precision 2
   - background from file

For ready to use pipeline go to:
- **2_InhomogeneousBackground/2_InhomogeneousBackground.cpproj**

3. Strong edges and gray interiors



Input imagery features:

In these images cells have distinctive bright edges and their interiors are of similar intensity as the background. Also cell internals are visible which can cause over-segmentation.

In order to improve the results we can use auto-adaptation to find better fitting parameters.

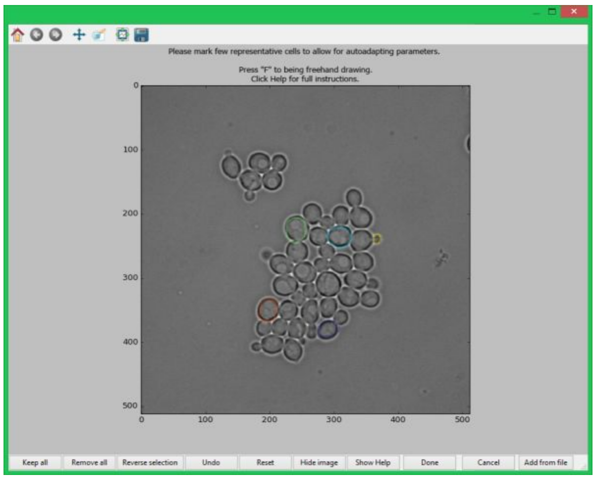
Auto adaptation:

Given input image and the expected partial-labeling the module can search for the best parameters. It is done in a number of independent iterations (steps) specified in IdentifyYeastCells module. At the end of every iteration best parameters found so far are saved to the pipeline.

In order to start the process set desired number of iterations (start from 1) and press "Autoadapt parameters" button. Depending on the module settings you will have to provide all images used in the segmentation (such as background or ignore mask).

The next step is hand-drawing contours on the input image. Try to follow below guidelines:

- choose contours from different parts of
- choose contours that are closely clumped together so CellStar can learn how to declump
- choose small and big but do not pick cells that are not important



Please mark few representative cells to allow for autoadapting parameters.

Press "F" to being freehand drawing.
Click Help for full instructions.

Alternatively you can prepare labels earlier and save it as file <input_image_and_extension>.lab.png (see Labels folder). Module will load that as an initial labeling.

There is also a button 'Add from file' for loading label from other location

When autoadaptation is finished you can run module and check the results on the new set of parameters

If adaptation was not successful

- try more adaptation iterations
- use different set of labels or input image

Pipeline:

[StrongEdgesAndGray_1_Default]

Default pipeline as in Example Usage 1:

1. LoadImage
2. IdentifyYeastCells
    - default parameters
    - precision 2

[StrongEdgesAndGray_2_Adapted]

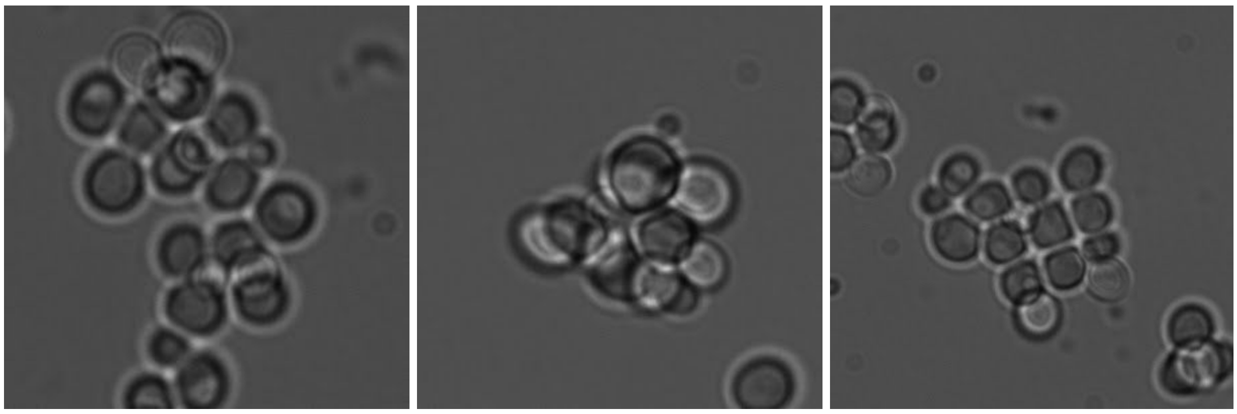
Pipeline after auto adaptation using TS6_BF_frame004.tif.lab.png labels.
Only difference is in 'Autoadapted Parameters':

1. LoadImage
2. IdentifyYeastCells
    - default parameters
    - autoadapted cell star parameters
    - precision 2

For ready to use pipeline go to:

- **3_StrongEdgesAndGray/StrongEdgesAndGray_2_Adapted.cpproj**

4. Dark cells with overlapping



Input imagery features:

In these images cells have distinctive bright edges and their interiors are darker than the background.

However in this case there is no effective enforcement of monolayer growth and therefore cells can grow and stay on top of each other which can interfere with CellStar segmentation. Two main problems arise:

Cell appearance change:

The appearance of yeast cells in bright-field imagery depend on the focus setting and their distance to focal point. This effect can be clearly see in FocusChange.gif where the distance is getting smaller. It means that CellStar can properly segment only a part of the cells and miss cells that are because of their Z-position are not recognized (high False Negative).

False Negative number may be reduced using autoadaptation (see Example Usage 3) however in this case it is not necessary and may enlarge the other problem.

Cell overlapping:

Overlapping occurs where more than one cell occupies the same region in the image. In this case it may be impossible to use data collected in fluorescent channels as it may yield false information (an unknown aggregation of all cells in the stack). CellStar may find some cells in these stacks but it is highly unwanted (high False Positive).

However we can still use CellStar to correctly extract a significant number of cells and with careful filtering reduce the number of False-Positives resulting in high precision which will allow to extract reliable research data.

Basic filtering:

There are 3 basic measurements type that we can use to improve precision:

- Area (filtered by CellStar)
- Quality (calculated by CellStar)
- Shape (calculated by CellProfiler)

The area filtering should be done in CellStar module as it can use that information to yield better segmentation. The threshold can be set high (e.g. 70% of average area) because in this case we accept higher False Negative number (see pipeline DarkButOverlapping_1_DefaultBig).

In order to find thresholds for Quality and Shape we need to calculate them for a sample of our imagery and visualize. We can view a histogram, see basic statistics or display measurements values on object in the image. The last method seems to be the most useful as it allows to identify which value respond to correct or incorrect cell (see pipeline DarkButOverlapping_2_Thresholds).

Quality measure can vary slightly from one image to another so it important to check a few to find a proper cut off. This measure should filter out under-segmentation and contours that cover background.

In shape measurement the most useful are Compactness and Eccentricity which describe how much circular the object is. It should allow us to filter out cells that are partial or incorrect contour found between other cells.

The filtering should significantly improve precision of the segmentation (see pipeline DarkButOverlapping_3_BasicFilter).

Advanced filtering using CPA classifiers (CPA version 2.2.1):

If False-Positive is too high after basic filtering we can use CellProfiler Analyst and its classifiers to improve filtering. First we need to export all the measurements (more than used in basic) and overlays to database. For more detailed description of CPA usage see general Manuals folder.

The measurements to export include:

- Shape
- Intensity
- Texture

Because the intensity measures can depend on the illumination of a particular image then it is beneficial to remove that factor. This can be achieved using Edge-Image of input for measurements calculation. LoG filter seems to be well suited for the task as it introduce less noise than other methods.

Overlays can be very useful in ground truth preparation and they can be saved to database using SaveImages with "Record file and path..." option on.

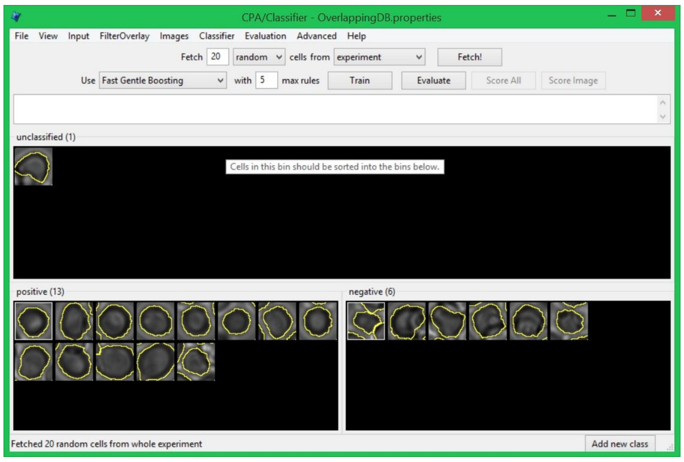
It is important to export only one set of objects to database (see simple configuration in pipeline DarkButOverlapping_4_ExportToCPA).

After finished analysis and export we should first edit *.properties file to ensure that object location data is not used in classification.
The ignored columns line should look like:

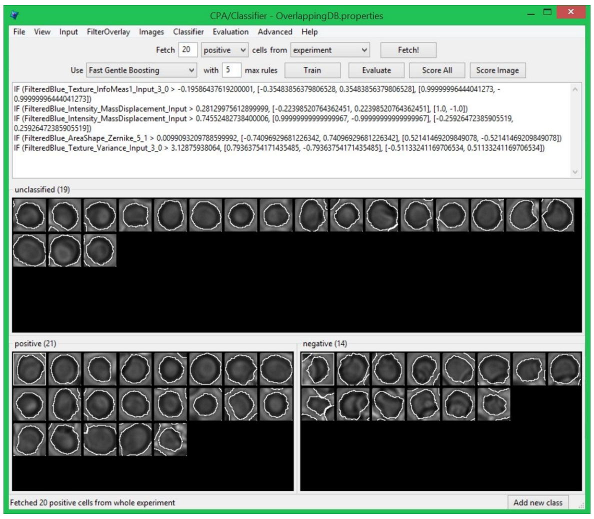
```
classifier_ignore_columns = table_number_key_column, image_number_key_column, object_number_key_column, ImageNumber, ObjectNumber, .*Parent.*, .*Children.*, .*Center_.*, .*Location.*, .*_Metadata_.*
```

This work can be automated with script that can be found in general Manuals folder.

Now we can run CPA and point it to created properties file. The idea is to train classifier based on manual classification (positive, negative) and then export as a set of rules that can be used in CP. First lets switch to FastGentleBoosting classifier as it is the one compatible with CellProfiler.


CPA/Classifier - OverlappingDB.properties
File View Input FilterOverlay Images Classifier Evaluation Advanced Help
Fetch 20 random cells from experiment Fetch!
Use Fast Gentle Boosting with 5 max rules Train Evaluate Score All Score Image
unclassified (1)
Cells in this bin should be sorted into the bins below.
positive (13)
negative (6)
Fetched 20 random cells from whole experiment
Add new class

The process should start with a number of random fetches of cells which we distribute to positive/negative bins. Then we can train classifier and use it to present us with random fetches from presumable positive or negative classes which allows us to quickly find and correct objects that are misclassified. It is important to have at least 20 various examples of both classes in final classification set. Another important factor is to have a number of images from different time points of the experiments so that the resulting classifier does not depend on the specific lighting conditions.


CPA/Classifier - OverlappingDB.properties
File View Input FilterOverlay Images Classifier Evaluation Advanced Help
Fetch 20 positive cells from experiment Fetch!
Use Fast Gentle Boosting with 5 max rules Train Evaluate Score All Score Image
unclassified (19)
positive (21)
negative (14)
Fetched 20 positive cells from whole experiment
Add new class

```
IF (FilteredBlue_Texture_InfoMeas1_Input_3_0 > -0.19586437619200001, [-0.35483856379806528, 0.35483856379806528], [0.99999996444041273, -
0.99999996444041273])
IF (FilteredBlue_Intensity_MassDisplacement_Input > 0.28129975612899999, [-0.22398520764362451, 0.22398520764362451], [1.0, -1.0])
IF (FilteredBlue_Intensity_MassDisplacement_Input > 0.74552482738400006, [0.99999999999999967, -0.99999999999999967], [-0.25926472385905519,
0.25926472385905519])
IF (FilteredBlue_AreaShape_Zernike_5_1 > 0.0099093209788599992, [-0.74096929681226342, 0.74096929681226342], [0.52141469209849078, -0.52141469209849078])
IF (FilteredBlue_Texture_Variance_Input_3_0 > 3.12875938064, [0.79363754171435485, -0.79363754171435485], [-0.51133241169706534, 0.51133241169706534])
```

The look of the tiles and be changed using menu - for example overlays can be turned on or off.

The complementing way to find and manually classify examples is to right click on tile and show full image. Then we can classify all objects in the image and drag-and-drop them from the image to the correct bins.

The training set (positive, negative bins) can be saved and loaded so it is not lost after closing CPA.

After every training the rules used for classification are shown and its length depend on the max rules setting which should not be higher than 20.

The result of the training should be rules.txt file with rules which have to manually copied from text box. They can be then used in CellProfiler FilterObject module. It is important to note that these rules include the name of the objects (in this case FilteredBlue) so if it is different then the file has to be modified.

- For our detailed manual for CPA classificator usage see: CPA_Usage.pdf in Manuals folder.
- For more information and guidelines for classificator training see: http://cellprofiler.org/cpa/5_classifier.html.
- For training video for classifactor training watch: https://www.youtube.com/watch?v=XMKgiRGb4IY.
- For more information and instruction about CPA see: http://cellprofiler.org/cpa/.

The final pipeline will consist of two filtering methods: basic and rules based (see pipeline DarkButOverlapping_5_Classifier).

Pipelines:

[DarkButOverlapping_1_DefaultBig]

1. LoadImage
2. IdentifyYeastCells
    - default parameters
    - precision 2 and (1.2 for random seeds)
    - filtering by area: 700 < area < infinite

[DarkButOverlapping_2_Thresholds]

As in [DarkButOverlapping_1_DefaultBig] but with measurements calculation and visualization.

1. LoadImage
2. IdentifyYeastCells
    - default parameters
    - precision 2 and (1.2 for random seeds)
    - filtering by area: 700 < area < infinite
3. MeasureObjectSizeShape
4. DisplayDataOnImage x 3 (quality, compactness, eccentricity)

[DarkButOverlapping_3_BasicFilter]

As in [DarkButOverlapping_2_Thresholds] but with filtering instead of visualization.

1. LoadImage
2. IdentifyYeastCells
   - default parameters
   - precision 2 and (1.2 for random seeds)
   - filtering by area: 700 < area < infinite
3. MeasureObjectSizeShape
4. FilterObjects

[DarkButOverlapping_4_ExportToCPA]

As in [DarkButOverlapping_3_BasicFilter] but then recalculating measurements for LoG image and exporting in CellProfiler Analyst compliant format.

1. LoadImage
2. IdentifyYeastCells
   - default parameters
   - precision 2 and (1.2 for random seeds)
   - filtering by area: 700 < area < infinite
3. MeasureObjectSizeShape
4. FilterObjects
5. EnhanceEdges (LoG of filtered)
6. MeasureObjectSizeShape (for filtered)
7. MeasureObjectIntensity (for filtered on LoG)
8. MeasureTexture (for filtered on LoG)
9. SaveImages (overlays to database)
10. ExportToDatabase

[DarkButOverlapping_5_Classifier]

As in [DarkButOverlapping_3_BasicFilter] but with more measurements and additional filtering step using rules calculated in CPA.

1. LoadImage
2. IdentifyYeastCells
   - default parameters
   - precision 2 and (1.2 for random seeds)
   - filtering by area: 700 < area < infinite
3. MeasureObjectSizeShape
4. FilterObjects (basic)
5. EnhanceEdges (LoG of filtered)
6. MeasureObjectSizeShape (for filtered)
7. MeasureObjectIntensity (for filtered on LoG)
8. MeasureTexture (for filtered on LoG)

9. FilterObjects (cpa rules)

Ready to use solution

The above discussion and pipelines are designed to show how the certain imagery and problems can be approached and solved. However they are not necessarily the best out-of-the-box solution. For example in this case the classifier was trained using only a few images and small ground truth. If you look for more flexible and general pipelines see _Solution folder.

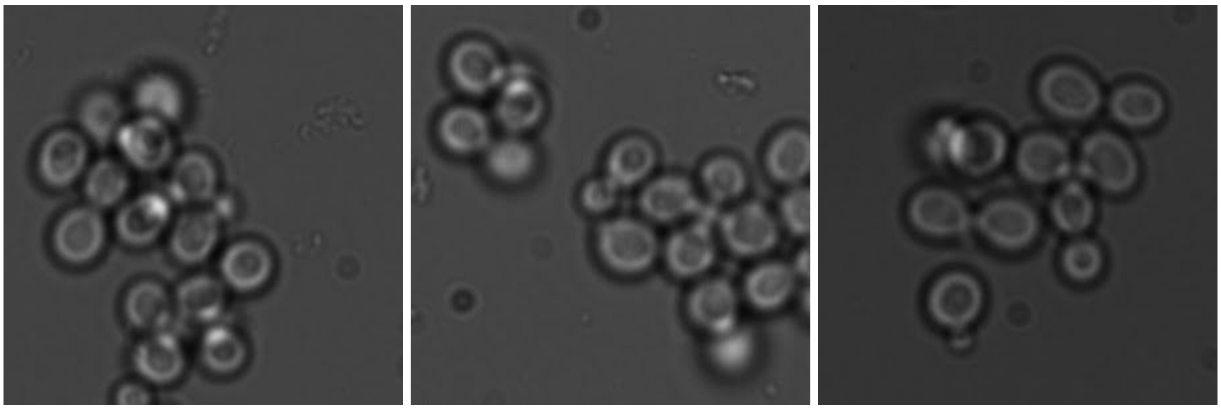
Random Forests classifiers

In CPA there is a number of available classifiers from which the most promising seem to be RandomForests which can yield better results than FastGentleBooting. However its results cannot be used in CP (as the export is not ready). Therefore one can use if it does not plan to move analysis back to CellProfiler.

For ready to use pipeline go to:

- **4_DarkButOverlapping/_Solution**
- **4_DarkButOverlapping/DarkButOverlapping_5_Final.cpproj**

5. Bright cells with overlapping



Input imagery features:

In these images cells have distinctive dark edges and their interiors are brighter than the background. CellStar has a switch which allow to properly find cells in this type of imagery.

However in this particular experiment there is no effective enforcement of monolayer growth and therefore cells can grow and stay on top of each other which can interfere with CellStar segmentation.

Imagery is somewhat similar to Example Usage 4 [EU4] (actually comes from the same experiment) the difference is the focus setting which makes the cells look quite different. Therefore in this description we will often refer to EU4 to avoid redundancy.

As in EU4 two main problems arise but now with different flavour:

Cell appearance change:

Focus setting (cells are closer than focal point) already makes cells to have out of focus look (bright with dark edges). This means that the change in appearance is less drastic than in EU4 (see FocusChange.gif). The cells that grow on top of each other are simply more blurred up to being just bright blobs. It can result in CellStar missing a significant number of cells.

False Negative number may be reduced using autoadaptation (see Example Usage 3) however it may also enlarge the overlapping problem.

Cell overlapping:

Overlapping as in EU4 makes it impossible to use data collected in fluorescent channels thus such cells have to be filtered out. However in this setting the problem is not that big as CellStar usually do not find any cells in regions with heavy overlap. On the other hand some overlapping cells may be so blurred so that they go unnoticed.

It is also notably that (contrary to EU4) overlapping regions are brighter than the rest of the image so there may be a way to find them and exclude from processing using IgnoreMask image.

However we can still use CellStar to correctly extract a significant number of cells and with careful filtering reduce the number of False-Positives resulting in high precision which will allow to extract reliable research data.

Basic filtering (as in EU4):

There are 3 basic measurements type that we can use to improve precision:

- Area (filtered by CellStar)
- Quality (calculated by CellStar)
- Shape (calculated by CellProfiler)

The threshold setting should be calculated as:

- Area -- high (e.g. 70% of average area)
- Quality -- imagery dependent (see below) can vary especially after parameter fitting
- Shape -- imagery dependent (see below) but these values are generally good:
    - Eccentricity < 0.7
    - Compactness < 1.1

In order to find thresholds for Quality and Shape we need to calculate them for a sample of our imagery and visualize. We can view a histogram, see basic statistics or display measurements values on object in the image. The last method seems to be the most useful as it allows to identify which value respond to correct or incorrect cell (see pipeline BrightButOverlapping_2_Thresholds).

See EU4 description for more details.

Ignoring overlapping regions:

If False-Positive is too high after basic filtering we can improve it by ignoring the regions where overlapping occurs. In this imagery, contrary to EU4, regions with overlapping cells is usually much brighter than the rest of the image. Therefore it can be found using thresholding and then applied as a IgnoreMask to CellStar.

CellStar marks IgnoreMask pixels as 'no-go zones' and it will not create any cells there and trim the close ones.

Flexible thresholding can be done in CellProfiler using ApplyThreshold module with Background method. It should return the pixels that much higher intensity than the rest of the image. In order to get rid of the noise we can erode that mask. Next step is dilation of the mask to ensure that all overlapping cells are effectively rejected.

The final pipeline will contain that ignore mask calculation as a first step and it should improve False-Positive number (see pipeline BrightButOverlapping_4_Ignoring).

Pipelines:

[BrightButOverlapping_1_DefaultBig]

1. LoadImage
2. IdentifyYeastCells
    - default parameters
    - 'Is area without cells brighter than cell interior' set to false
    - precision 2 and (1.2 for random seeds)
    - filtering by area: 700 < area < infinite

[BrightButOverlapping_2_Thresholds]

As in [BrightButOverlapping_1_DefaultBig] but with measurements calculation and visualization.

1. LoadImage
2. IdentifyYeastCells
    - default parameters
    - 'Is area without cells brighter than cell interior' set to false
    - precision 2 and (1.2 for random seeds)
    - filtering by area: 700 < area < infinite
3. MeasureObjectSizeShape
4. DisplayDataOnImage x 3 (quality, compactness, eccentricity)

[BrightButOverlapping_3_BasicFilter]

As in [BrightButOverlapping_2_Thresholds] but with filtering instead of visualization.

1. LoadImage
2. IdentifyYeastCells
    - default parameters
    - 'Is area without cells brighter than cell interior' set to false
    - precision 2 and (1.2 for random seeds)
    - filtering by area: 700 < area < infinite
3. MeasureObjectSizeShape
4. FilterObjects

[BrightButOverlapping_4_IgnoreOverlap]

Estimate overlap mask and give it to CellStar module. Rest of a pipeline as in [DarkButOverlapping_3_BasicFilter].

1. LoadImage
2. ApplyThreshold (background, no smooth)
3. Morph (erosion-3)
4. Morph (dilation-11 x 3)
5. IdentifyYeastCells
   - default parameters
   - 'Is area without cells brighter than cell interior' set to false
   - Select ignore mask: MorphThresholdMask
   - precision 2 and (1.2 for random seeds)
   - filtering by area: 700 < area < infinite
6. MeasureObjectSizeShape
7. FilterObjects

Further improvements

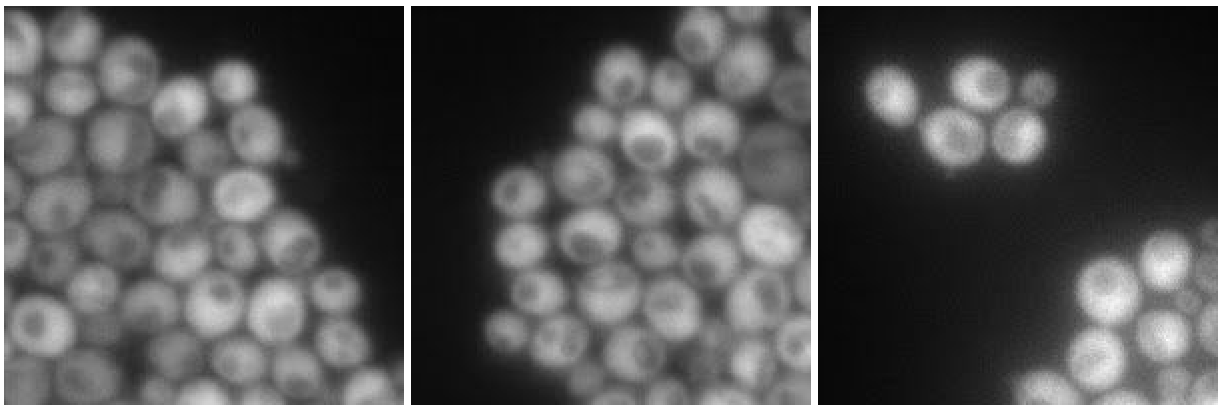
In order to further improve the quality of the segmentation, the number of changes have to be investigated:

- use parameter fitting to improve False-Negative number, however this will also produce more False-Positive (notably in overlapping areas)
- find overlapping regions will more precision using Ilastik module which classify pixels using user provided ground truth
- for better filtering train classifier as in EU4

For ready to use pipeline go to:

- **5_BrightButOverlapping/BrightButOverlapping_4_IgnoreOverlap.cpproj**

6. Fluorescent imagery



Input imagery features:

The images in fluorescent channel with cells designed so that their entire interior respond.

This is not data that CellStar was designed for and there exist a number of other dedicated software for segmentation in fluorescent imagery (notably in CellProfiler). However in some dense colony scenarios it may be difficult for them to properly disjoint nearby cells causing under-segmentation.

Therefore CellStar was extended to try to adapt to this type of imagery. This option is in proof of concept state so it may not work well for all datasets. The resulting segmentation should be filtered to remove incorrect contours.

We can also use autoadaptation to try further improve segmentation. See Example Usage 3 for more details.

Smoothing contours

The resulting contours are irregular and sharp which should not be the case for yeast cells. It can also interfere with some of the shape measures.

Simple way to achieve smooth contours is shrinking and expanding objects (see pipeline FluorescentImagery_1_Smoothed).

Filtering:

There are 3 basic measurements type that we can use to improve precision:

- Area (filtered by CellStar)
- Quality (calculated by CellStar)
- Shape (calculated by CellProfiler)

However in this case we will use only Area and Shape. The reasoning is that Area as well as basic Shape parameters can be easily configured as they are fairly constant in yeast cells population. Therefore the default ones should be good in filtering.

See Example Usage 4 if want to tune the filtering parameters.

Pipelines:

[FluorescentImagery_1_Smoothed]

1. LoadImage
2. CorrectIlluminationCalculate (fit polynomial)
3. CorrectIlluminationApply (divide)
4. ExpandOrShrinkObjects (shrink)
5. ExpandOrShrinkObjects (expand)
6. IdentifyYeastCells
    - higher average cell diameter than actual 40 (real around 30)
    - 'Do you want to segment brightfield images' set to No
    - 'Is area without cells brighter than cell interior' set to No
    - precision 2 and (1.5 for random seeds, 3 cell size variants)

[FluorescentImagery_2_BigShapeFilter]

As in [FluorescentImagery_1_FluBig] but with shape measuring and filtering.

1. LoadImage
2. CorrectIlluminationCalculate (fit polynomial)
3. CorrectIlluminationApply (divide)
4. IdentifyYeastCells
    - higher average cell diameter than actual 40 (real around 30)
    - 'Do you want to segment brightfield images' set to No
    - 'Is area without cells brighter than cell interior' set to No
    - precision 2 and (1.5 for random seeds, 3 cell size variants)
    - filtering by area: 300 < area < infinite
5. MeasureObjectSizeShape
6. FilterObjects (compactness, eccentricity)

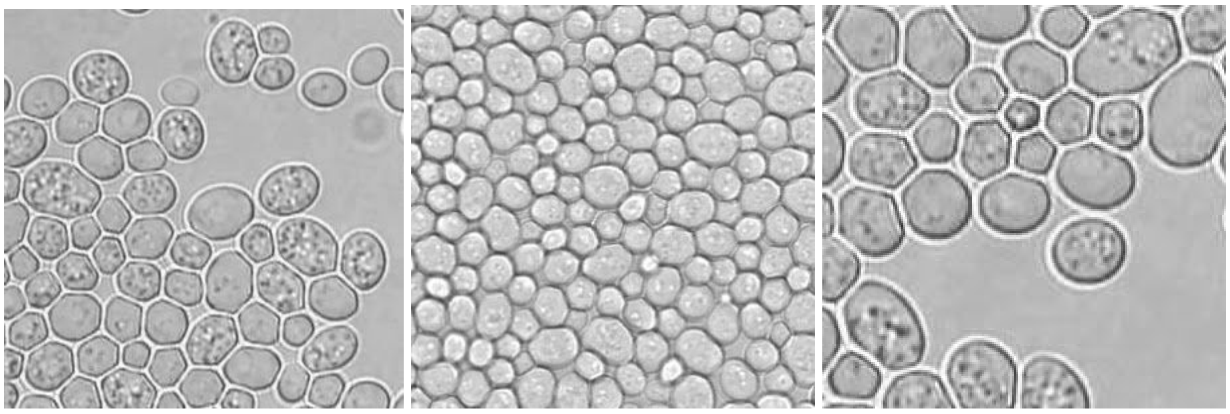
Ready to use solution

The above discussion and pipelines are designed to show how the certain imagery and problems can be approached and solved. However they are not necessarily the best out-of-the-box solution. If you look for a better performing pipeline see _Solution folder.

For ready to use pipeline go to:

- **6_FluorescentImagery/_Solution**
- **6_FluorescentImagery/FluorescentImagery_2_BigShapeFilter.cpproj**

7. White cells



Input imagery features:

In these images cells have distinctive edges (bright or white) and their interiors are white and similar intensity as the background. Also cell internals are visible which can cause over-segmentation. The general features are quite similar to Example Usage 3 (*StrongEdgesAndGray*).

It is important to note that although both input images look similar they required different switch state.

The initial parameters may not yield satisfactory results so in order to improve we can use auto-adaptation to find better fitting parameters (see Example Usage 3 for details).

Pipelines:

[StrongEdgesAndWhite_1_Default]

1. LoadImage
2. IdentifyYeastCells (for bright edges)
    - default parameters
    - precision 2
    - 'Is area without cells brighter than cell interior' set to Yes
3. IdentifyYeastCells (for bright edges)
    - default parameters
    - precision 2
    - 'Is area without cells brighter than cell interior' set to No

[StrongEdgesAndWhite_2_Adapted]

Pipeline with auto-parameters after auto adaptation. Only difference to [StrongEdgesAndWhite_1_Default] is in 'Autoadapted Parameters'.

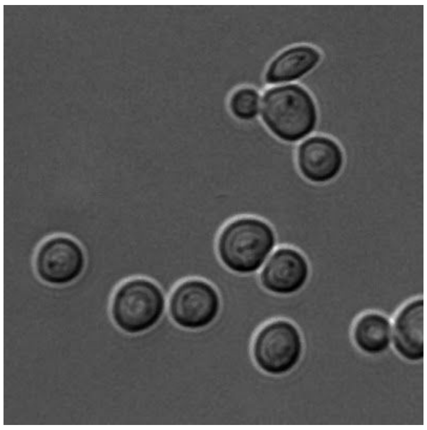
Ready to use solution

The above discussion and pipelines are designed to show how the certain imagery and problems can be approached and solved. However they are not necessarily the best out-of-the-box solution. If you look for a better performing pipeline see _Solution folder.

For ready to use pipeline go to:

- **7_StrongEdgesAndWhite_Solution\BrightEdgesAndWhite_2_Adapted.cpproj**
- **7_StrongEdgesAndWhite_Solution\DarkEdgesAndWhite_2_Adapted.cpproj**

A. Various parameter fitting



Input imagery features:

In this category we will place any other imagery which can be solved using parameter auto-adaptation.

Ready to use pipelines:

[DarkWithMoreInternals_Adapted]

Pipeline for segmentation of DarkCloser1.tif

1. LoadImage
2. IdentifyYeastCells
    - autoadapted parameters
    - precision 2 (and 1.2 random seeds)
    - 'Is area without cells brighter than cell interior' set to Yes