Nome: Melissa Augusto Ribeiro
No. USP: 10850432

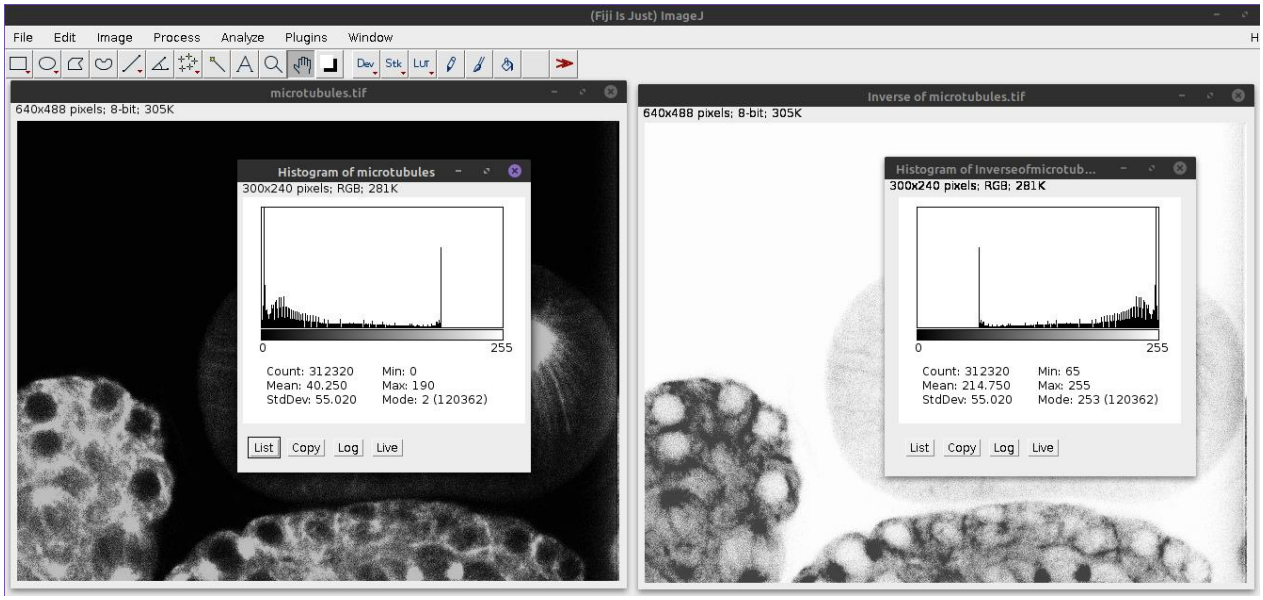
Relatório de Experiência da Atividade 1

1. Desenvolvimento de Plugin

O docente disponibilizou um arquivo .java contendo funções. Tais funções foram utilizadas como plugins na aplicação ImageJ através do pacote pointwise, que pôde ser acessado pelo Menu do programa. Como exercício, foi solicitado que adicionássemos linhas de códigos em alguns trechos do arquivo, para que plugin tivesse o funcionamento correto. Algumas dessas alterações estão listadas ao decorrer do relatório e o arquivo final será anexado ao formulário de entrega.

2. A transformação pontual para melhorar visualização

2.1. **Entendimento: inversão de contraste:** Exercício realizado com objetivo de compreender a rotina inverse(),visto que ela muda os valores de pixel da imagem f(x, y) para g (x, y) = 255 - f (x, y). Aplicou-se a rotina à imagem *microtubules.tif* chamando o plugin "Inverse". Abaixo, pode-se observar os histogramas das imagens original e contraste invertido.



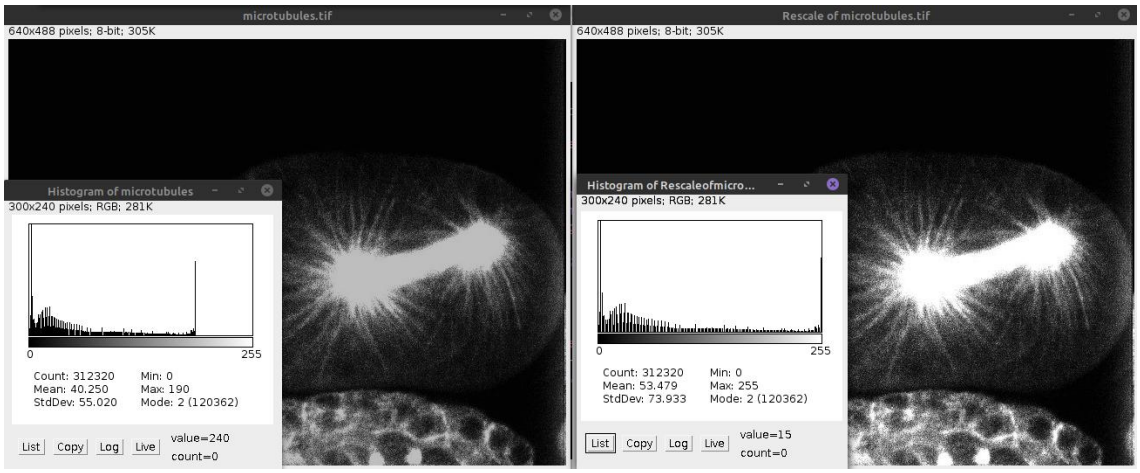
2.2. **Esticando e normalizando contraste:** A rotina rescale() foi escrita para alterar o valor de pixel da imagem, reescalonando g (x, y) a [0,255]. Após desenvolvida, a rotina foi aplicada à imagem de *microtubules.tif* com o plugin "Rescale". O código da rotina e o histogramas gerados podem ser observado abaixo.

f(x, y) para: $g(x,y) = \alpha (f(x,y) - \beta)$
$\beta = \min(f(x,y))$
$\alpha = 255 / (\max(f(x,y)) - \min(f(x,y)))$.

Insira o código “Rescale”

```
/**
* Question 2.2 Stretch normalized constrast
*/
static public ImageAccess rescale(ImageAccess input) {
    int nx = input.getWidth();
    int ny = input.getHeight();
    double max = input.getMaximum();
    double min = input.getMinimum();
    ImageAccess output = new ImageAccess(nx, ny);
    double pixel = 0.0;
    double alpha = 255/(max-min);
    double beta = min;

    for(int i=0; i<nx; i++){
        for(int j=0; j<nx; j++){
            pixel = input.getPixel(i,j);
            pixel = alpha*(pixel-beta);
            output.putPixel(i, j, pixel);
        }
    }
    return output;
}
```



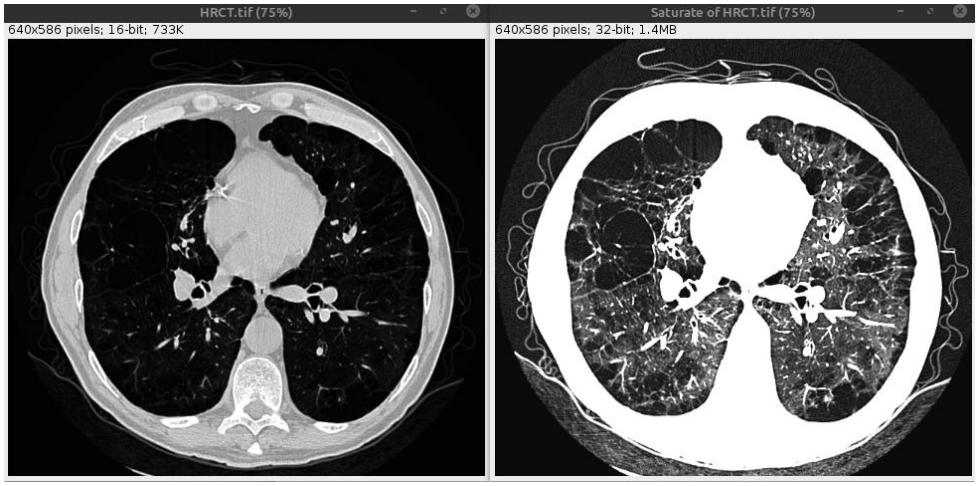
2.3. **Aplicação: Satura uma imagem médica:** A rotina saturate() foi escrita para por os pixels a 10.000 quando o valor de entrada é maior a 10.000. Após aplicação da rotina no arquivo *HRCT.tif* por meio do plugin "Saturate", um novo histograma foi gerado. O código, as imagens e o histograma podem ser verificados abaixo.

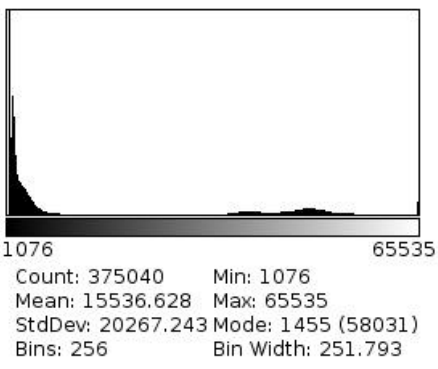
Insira o código “Saturate”

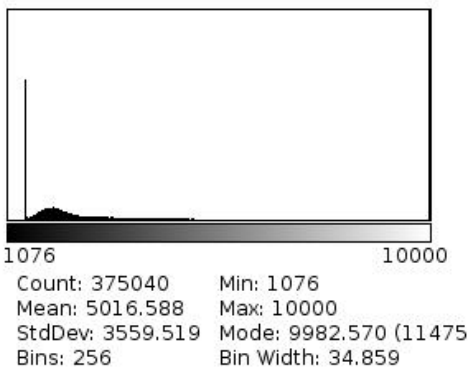
```
/**
* Question 2.3 Saturate an image
*/
static public ImageAccess saturate(ImageAccess input) {
    int nx = input.getWidth();
    int ny = input.getHeight();
    ImageAccess output = new ImageAccess(nx, ny);
    double pixel = 0.0;

    for(int i=0; i<nx; i++){
        for(int j=0; j<ny; j++){
            pixel = input.getPixel(i,j);
            if(pixel > 10000)
                pixel = 10000;

            output.putPixel(i, j, pixel);
        }
    }
    rescale(output);
    return output;
}
```





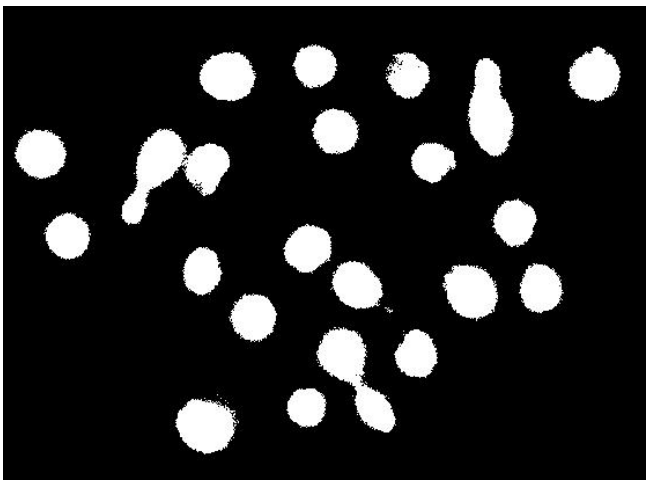


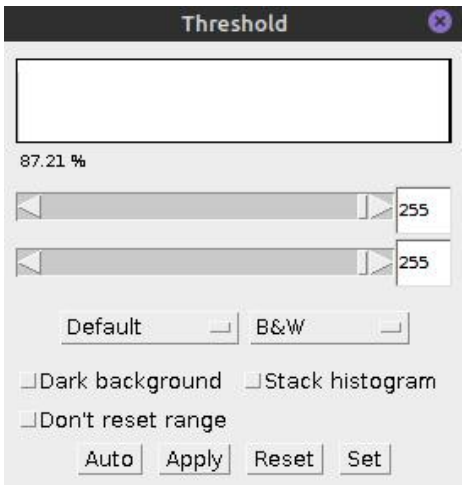
3. A Segmentação por Limiar: “thresholding”

3.1. **Contando partículas:** Usar o comando de "Limiar" de ImageJ, tenta segmentar as imagens: yeast1, yeast2, yeast3 e yeast4 para obter 25 objetos no analisador de partículas de ImageJ (Analisa -> Analisa Partículas). Se não for possível achar um limiar adequado, explique por que no arquivo de relatório.doc.

Insira a imagem resultante

- **yeast1:** Para obter 25 objetos no analisador de partículas do aplicativo ImageJ, foi necessário utilizar um *threshold* de 87,21% (B&W) ou 12,79% (invertido).



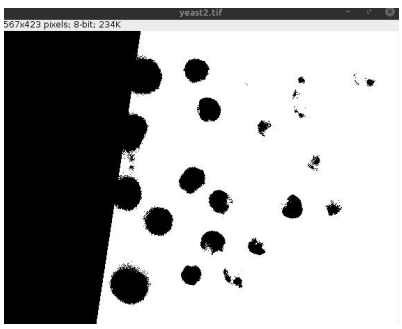


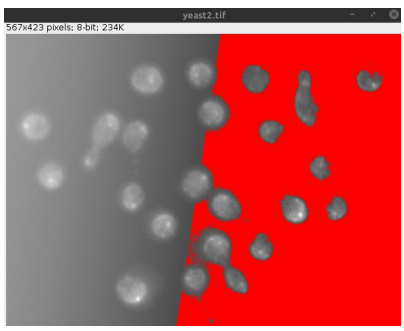
Summary

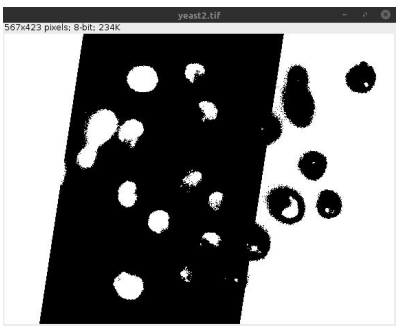
File Edit Font

Slice	Count	Total Area	Average Size	%Area	Mean
yeast1.tif	25	209174	8366.960	87.214	255

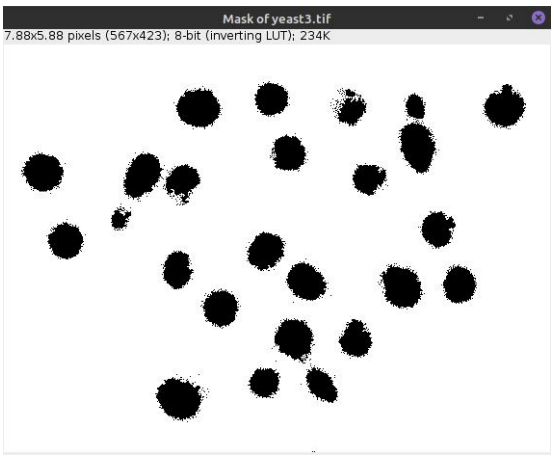
- **yeast2:** não foi possível definir um limiar nessa imagem para segmentar 25 objetos. Os pixels da vizinhança se sobrepõem às partículas que deveriam ser analisadas, conforme demonstrado nas imagens abaixo.

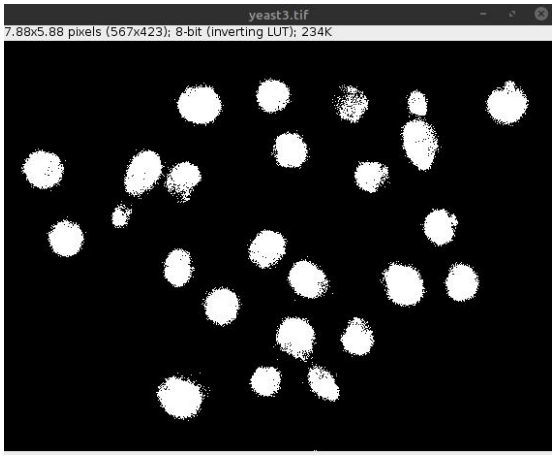


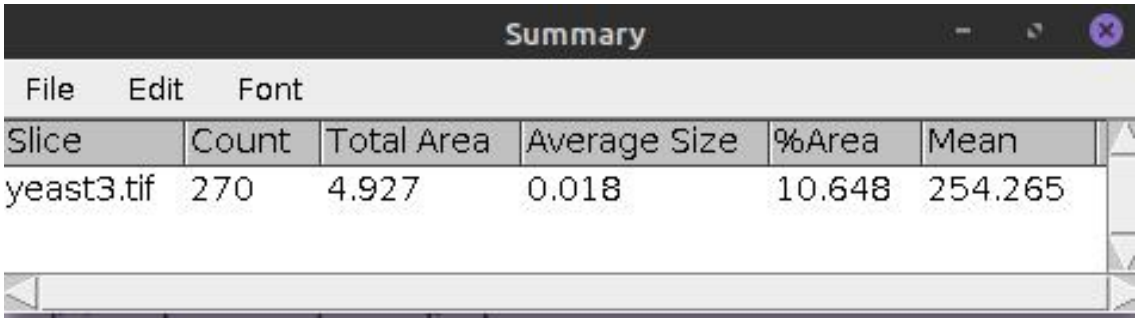




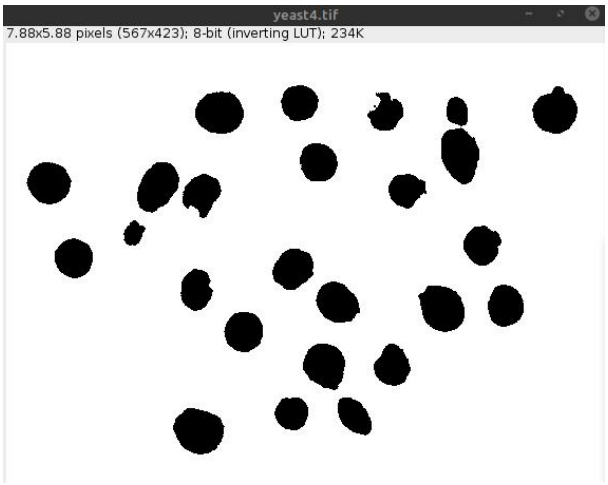
- **yeast3:** Apesar de conseguirmos observar a segmentação de 25 itens na imagem, não foi possível definir um limiar para que o contador de partículas fosse capaz de alcançar uma contagem satisfatória; existem ruídos (“granulado”) na imagem que acabam por ser adicionadas na contagem final.

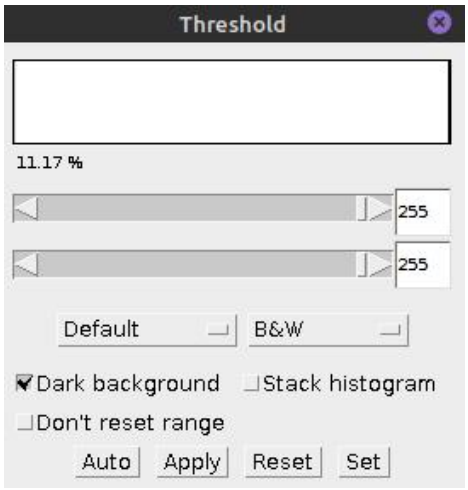


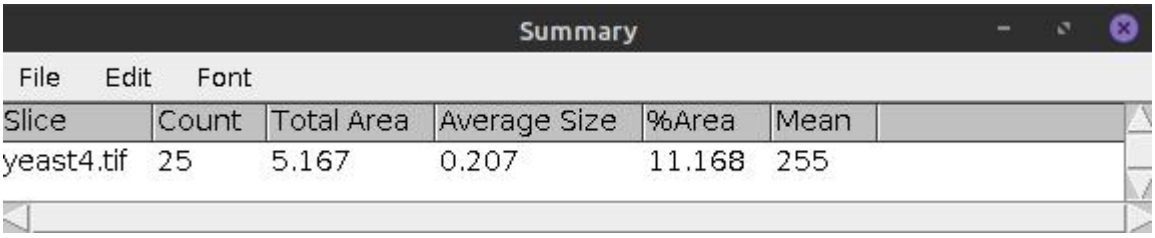




- **yeast4:** Para obter 25 objetos no analisador de partículas do aplicativo ImageJ, foi necessário utilizar um *threshold* de 11,17% (B&W) com configuração de background escuro.

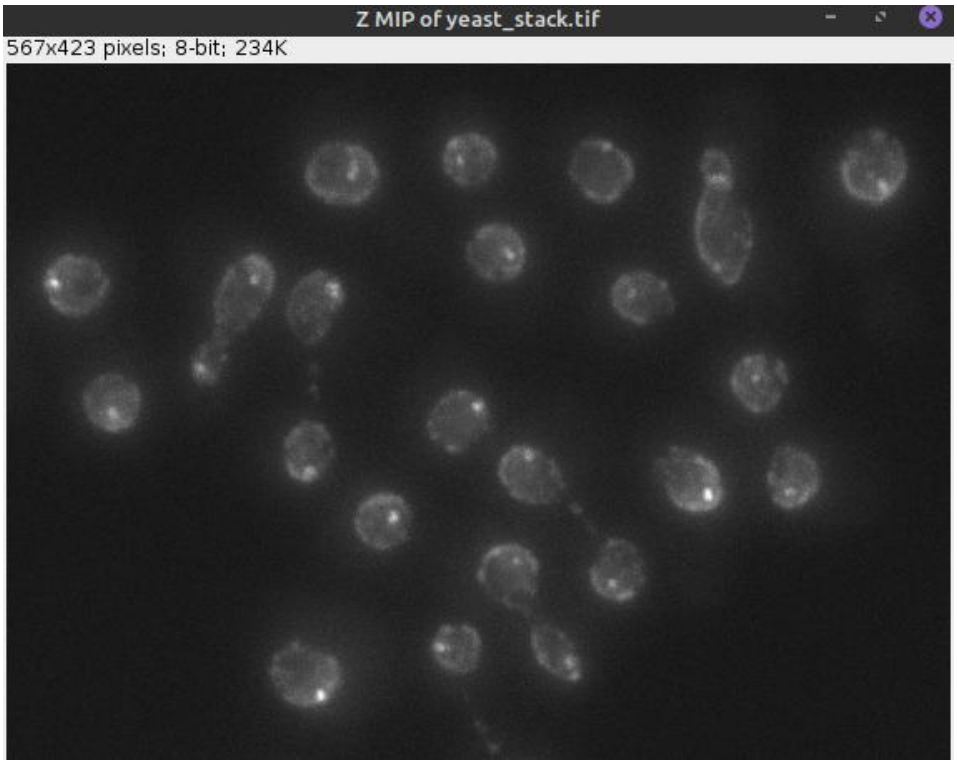




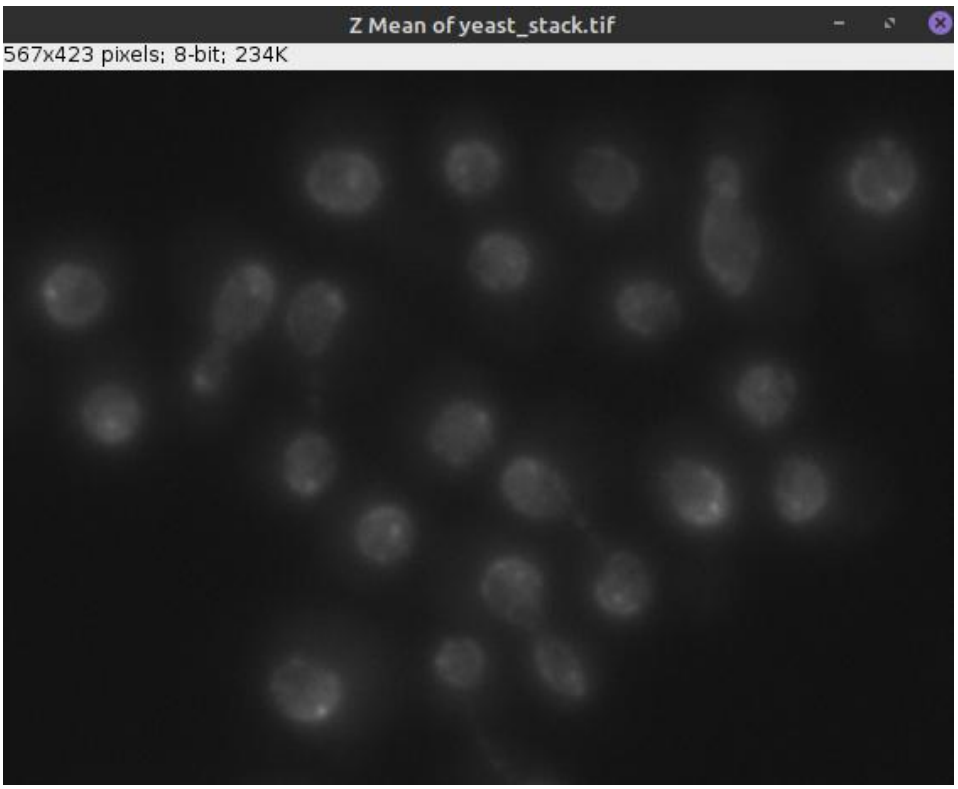


4. **Projeção de "Z-Stack"**

4.1. **Projeção de máxima intensidade:** Escreva a rotina zprojectMaximum() que gera a imagem máxima de projeção de intensidade de uma pilha de imagens. Aplique a rotina ao *yeard_stack*. tif por chamar o plugin ZMIP.



4.2. **Projeção de intensidade média:** Escreva a rotina zprojectMean() que gera a imagem média de uma pilha de imagens. Aplique a rotina ao *yeard_stack.tif* por chamar o plugin ZMean.



4.3. **Apresentação de imagens de z-stack:** Usar dois plugins préviamente, o comando "Brilho & Contraste", o comando "Image Calculator" → "Color Merge" do ImageJ na tentativa de obter uma imagem composta de cor onde claramente podemos distinguir a célula de fermento, o núcleo de GFP-tagged e o telomere de GFP-tagged (mancha). A fonte é uma z-stack de imagens fluorescentes chamada *yeast_stack. tif* e a imagem de fase é *yeast_phase. tif*. Na imagem acima está um exemplo de que nós podemos esperar. Coloque a imagem resultante no arquivo relatório.doc.

Insira a imagem resultante