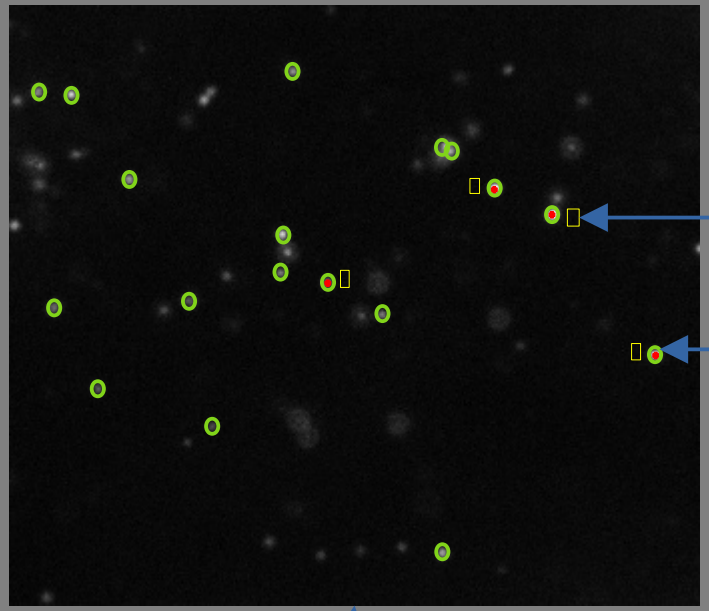
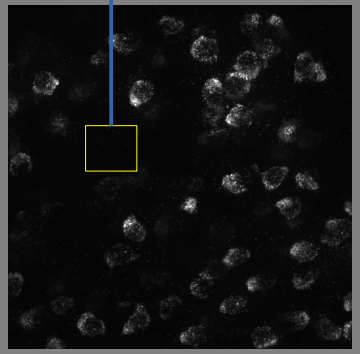
« Single mRNA » calibration





Xml file to describe dots positions

Roi file to describe dots rois background

Dots segmentation

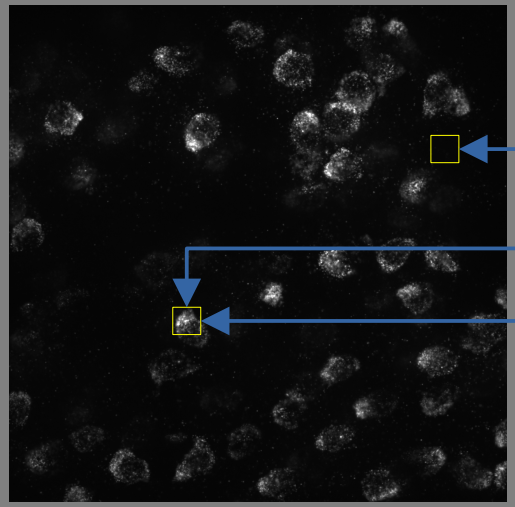
$$mean\ dot\ Int.\ bg = \frac{\sum_{dot\ zmin}^{dot\ zmax} roi\ Integ.\ Int.}{\sum_{dot\ zmin}^{dot\ zmax} roi\ area}$$

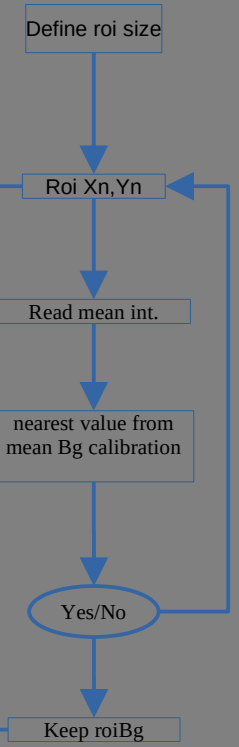
$$Corrected\ single\ mRNA\ Integrated\ Int. = Mean\left(dot\ Integ.\ Int. - mean\ dot\ bg\ Int. * dot\ vol\right)$$

Background detection

Find auto background

Stack gene image

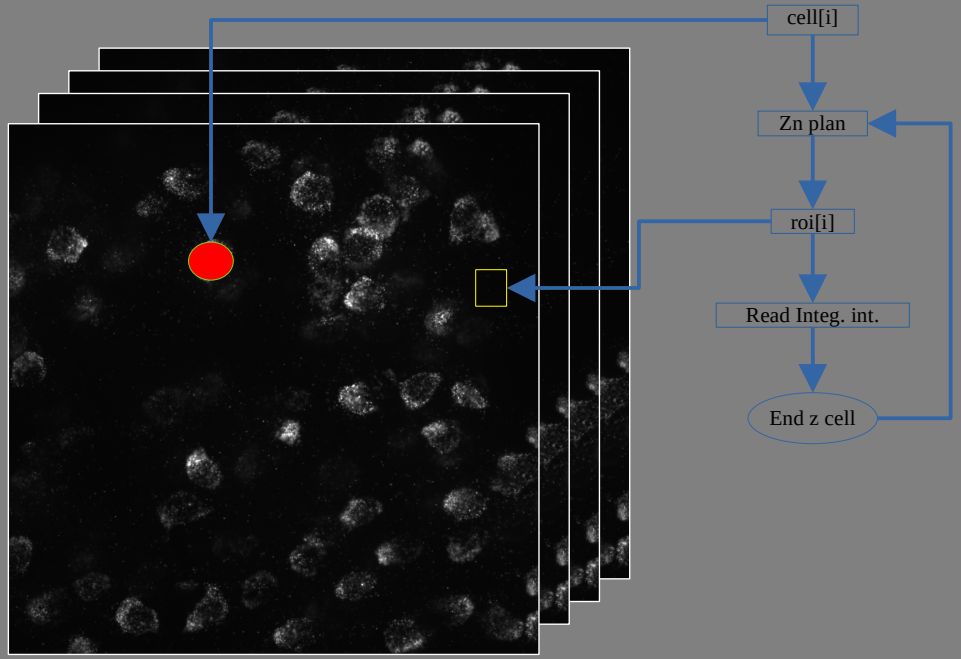





```
/**
 * Find nearest background roi
 * @param img
 * @param size
 * @return
 */
public static Roi findRoiBbackgroundAuto(ImagePlus img, double bgGene) {
    // scroll gene image and measure bg intensity in roi
    // take roi at intensity nearest from bgGene

    ArrayList<RoiBg> intBgFound = new ArrayList<RoiBg>();

    for (int x = 0; x < img.getWidth() - roiBgSize; x += roiBgSize) {
        for (int y = 0; y < img.getHeight() - roiBgSize; y += roiBgSize) {
            Roi roi = new Roi(x, y, roiBgSize, roiBgSize);
            img.setRoi(roi);
            ImagePlus imgCrop = img.crop("stack");
            double bg = find_background(imgCrop, 1, img.getNSlices());
            intBgFound.add(new RoiBg(roi, bg));
            closeImages(imgCrop);
        }
    }
    img.deleteRoi();
    // sort RoiBg on bg value
    intBgFound.sort(Comparator.comparing(RoiBg::getBgInt));

    // Find nearest value from bgGene
    double min = Double.MAX_VALUE;
    double closest = bgGene;
    Roi roiBg = null;
    for (RoiBg v : intBgFound) {
        final double diff = Math.abs(v.getBgInt() - bgGene);
        if (diff < min) {
            min = diff;
            closest = v.getBgInt();
            roiBg = v.getRoi();
        }
    }
    int roiCenterX = roiBg.getBounds().x+(roiBgSize/2);
    int roiCenterY = roiBg.getBounds().y+(roiBgSize/2);
    System.out.println("Roi auto background found = "+closest+" center x = "+roiCenterX+", y = "+roiCenterY);
    return(roiBg);
}
```

Background detection

Cell background

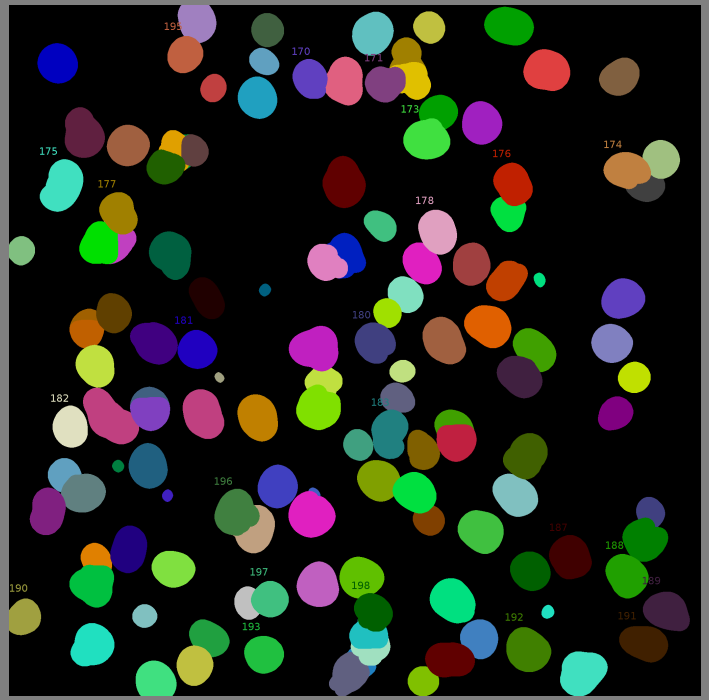


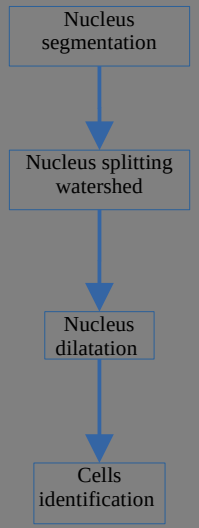
Gene X stack

```
/*
   * Get Mean of intensity in roi cropped stack
   */
   public static double find_background(ImagePlus img, int zMin, int zMax) {
      ResultsTable rt = new ResultsTable();
      Analyzer ana = new Analyzer(img, Measurements.INTEGRATED_DENSITY, rt);
      double intDen = 0;
      int index = 0;
      for (int z = zMin; z <= zMax; z++) {
         img.setSlice(z);
         ana.measure();
         intDen += rt.getValue("RawIntDen", index);
         index++;
      }
      double vol = img.getWidth() * img.getHeight() * (zMax - zMin + 1);
      double bgInt = intDen / vol;
      rt.reset();
      return(bgInt);
   }
```

$$mean\ bg\ Int.(cell[i]) = \frac{\sum_{cell\ zmin}^{cell\ zmax} roi\ Integ.\ Intensity}{\sum_{cell\ zmin}^{cell\ zmax} roi\ area}$$

Cells detection

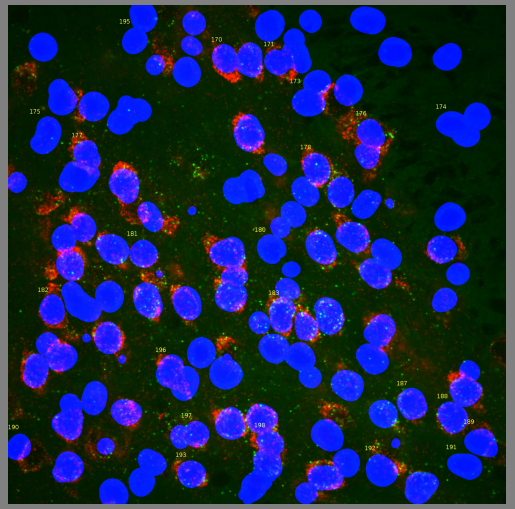





```
/**
 * Nucleus segmentation 2
 * @param imgNuc
 * @return cellPop
 */
public static Objects3DPopulation find_nucleus2(ImagePlus imgNuc) {
    ImagePlus img = new Duplicator().run(imgNuc);
    ImageStack stack = new ImageStack(img.getWidth(), imgNuc.getHeight());
    for (int i = 1; i <= img.getStackSize(); i++) {
        IJ.showStatus("Finding nucleus section "+i+" / "+img.getStackSize());
        img.setZ(i);
        img.updateAndDraw();
        IJ.run(img, "Nuclei Outline", "blur=20 blur2=30 threshold_method="+threshold+" outlier_radius=50 outlier_threshold=1
max_nucleus_size=100 "
                + "min_nucleus_size=10 erosion=5 expansion_inner=5 expansion=5 results_overlay");
        img.setZ(1);
        img.updateAndDraw();
        ImagePlus mask = new ImagePlus("mask", img.createRoiMask().getBufferedImage());
        ImageProcessor ip =  mask.getProcessor();
        ip.invertLut();
        for (int n = 0; n < 3; n++)
            ip.erode();
        stack.addSlice(ip);
    }
    ImagePlus imgStack = new ImagePlus("Nucleus", stack);
    IJ.showStatus("Starting watershed...");
    ImagePlus imgWater = WatershedSplit(imgStack, 8);
    closeImages(imgStack);
    imgWater.setCalibration(imgNuc.getCalibration());
    Objects3DPopulation cellPop = new Objects3DPopulation(imgWater);
    cellPop.removeObjectsTouchingBorders(imgWater, false);
    closeImages(imgWater);
    closeImages(img);
    return(cellPop);
}
public static  Objects3DPopulation findNucleus(ImagePlus imgNuc, ImagePlus imgGene) {
    Objects3DPopulation nucPopOrg = new Objects3DPopulation();
    nucPopOrg = find_nucleus2(imgNuc);
    System.out.println("-- Total nucleus Population :"+nucPopOrg.getNbObjects());
    // size filter
    Objects3DPopulation nucPop = new Objects3DPopulation(nucPopOrg.getObjectsWithinVolume(minNucVol, maxNucVol, true));
    int nbNucPop = nucPop.getNbObjects();
    System.out.println("-- Total nucleus Population after size filter: "+ nbNucPop);
    // create dilated nucleus population
    Objects3DPopulation cellsPop = new Objects3DPopulation();
    if (nucDil != 0)
        for (int o = 0; o < nucPop.getNbObjects(); o++) {
            Object3D obj = nucPop.getObject(o);
            //cellsPop.addObject(obj.getDilatedObject((float)(nucDil/cal.pixelWidth), (float)(nucDil/cal.pixelHeight), (float)(nucDil)));
            cellsPop.addObject(dilCellObj(imgNuc, obj));
        }
    else
        cellsPop = findCell(imgGene, nucPop);
    return(cellsPop);
}
```

mRNA quantifications

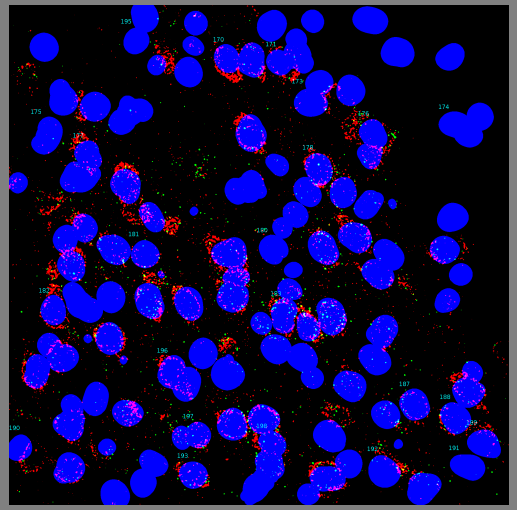
Cell intensity



$$Corrected\ cell\ integrated\ intensity = cell\ integrated\ intensity - (mean\ bg\ Int. * cell\ vol)$$

$$Total\ single\ mRNA\ per\ cell = \frac{corrected\ cell\ Integ.\ Int.}{corrected\ single\ mRNA\ Integ.\ Int.}$$

Dot intensity



$$Corrected\ dots\ integrated\ intensity = cell\ dots\ integrated\ intensity - (mean\ bg\ Int. * cell\ vol)$$

$$Total\ single\ mRNA\ per\ cell = \frac{corrected\ dots\ cell\ Integ.\ Int.}{corrected\ single\ mRNA\ Integ.\ Int.}$$

https://github.com/pmailly/RNA_Scope.git