Using masks in the phasor analysis

Instruction

While in the phasor page, double click in the image that you want to use to generate the mask. When the bitmap library popups, right click on the image. Select the shape and adjust position and shape. In the Tools menu click on “get ROI”. This process generates the mask. In the phasor page menu in the tools click on show image in phasor 2. The phasor should be calculated only in the region of the mask when you press the “Recalculate” button. You can continue to move the ROI in the bitmap library popup to select different regions of the image, but remember to press the get ROI button in the tools menu to activate the mask. Then you can keep moving the ROI and just calculate the average and add the point. You don’t need to press recalculate. Recalculate is only used to show the selected area in the image series.

IMPORTANT: I added a check near the “Average in the cursor” button to indicate that the average must be calculated in the mask rather than in the cursor. In the previous editions of SimFCS this capability was missing so it was not possible to calculate the average in the ROI but only in the cursor.

Examples:

Given a number of images select a region of the image and calculate the average phasor in the region for the image series

The following image shows the placements of 3 ROIs.
The boolean function is OR, which means that all the pixels selected will be added. After the getROI, selecting the mask check and then recalculate the images look like in the following images

Now pressing the average in cursor with the scan all images set, you should get the following memo which contains the analysis of the different ROIs

You can copy and paste the results in Excel or any other program to plot and statistically analyze the result.
For example, importing the above data in excel, sorting according to the first column for “image #” and plotting the fraction of bound (whatever that means for this data set) we get

![Bar chart showing the fraction of bound for different regions.]

Explanations: image is the average if all regions that satisfy the Boolean condition, while Mask is the analysis in each individual ROI.

Columns 1 and 2 are the G and S average coordinates of each ROI

Column 3 is the average intensity in the ROI

Column 4 is the “fraction of bound” according to the values used for the lifetime of the free and bound species

Column 5 is the number of pixels in the ROI that were used for the analysis (that means all pixels that are in threshold intensity range in that ROI)

How to use the Boolean conditions?

AND means the pixels that are in common in all ROI’s used

OR means all the pixels in each ROI

XOR mean pixel that are only in one ROI but not in any other.

You can use the XOR function to produce a “ring like” mask as shown below
Technical details

The function used by the SimFCS to process an image only in a selected mask is called in_cursor3.

```pascal
function in_cursor3(i,j:integer; var k:longword):boolean;
var radius,radius2:double;
begin
  result:=true;
  if lifetimeform.CheckBox3.Checked and frameform.showROI1.checked then
    begin
      radius:=frameform.bubbleseries3.radiusvalues[0];
      radius2:=2*frameform.bubbleseries3.radiusvalues[0];
      case frameform.bubbleseries3.pointer.style of
        psrectangle: if (i>=frameform.series3.xvalue[0])
          and (i<=frameform.series3.xvalue[0]+radius2)
          and (j>=frameform.series3.yvalue[0])
          and (j<=frameform.series3.yvalue[0]+radius2)
          then result:=true else result:=false;
        pscircle: if (sqrt((i-frameform.bubbleseries3.xvalue[0])^2+
          (j-frameform.bubbleseries3.yvalue[0])^2)<=radius)
          then result:=true else result:=false;
        pshexagon: if maskb0^[j,i]<>-1 then
          begin
            result:=(maskb0^[j,i] and $80000000)>0;
            k:=maskb0^[j,i] and $7fffffff;
          end
          else
            begin
              result:=false;
              k:=$FFFFFFFF;
            end;
        end;//of case
  if defaultsform.checkbox1.checked
    then result:=not result;
  end; //of if
end;
```
This procedure sets the amsk in vmask

```
procedure tBmform.set_vmask(Sender:Tobject);
var aux,i,j,k,side,m,n:integer;
rx,ry:integer;
x,y:single;
px,py:array[0..32,0..32] of single; //local store for the sides of the polygons j index and for each polygon k index
begin
  fillchar(vmask,sizeof(mydatasingle),#0);
  for k:=0 to series4.Shapes.count-1 do //save all the polygon series and clear
    begin
      if k>31 then continue;//no more than 31 symbols
      sides[k]:=series4.shapes.Polygon[k].Points.Count-1;
      for j:=0 to sides[k] do { add the XY coordinates }
        begin
          px[k,j]:=Series4.Shapes.Polygon[k].points.XValue[j];
          py[k,j]:=Series4.Shapes.Polygon[k].points.yValue[j];
        end;
      series4.shapes.Polygon[k].Points.clear;
    end;
  for k:=0 to series4.Shapes.count-1 do //restore one by one
    begin
      if k>31 then continue;//no more than 31 symbols
      for j:=0 to sides[k] do { add the XY coordinates }
        Series4.Shapes.Polygon[k].AddXY(px[k,j],py[k,j]);
      series1.clear;
      for j:=0 to sides[k] do series1.addxy(px[k,j],py[k,j]);
      for j:=0 to sizey do
        for i:=0 to sizex do
          begin
            rx:=(i*chart1.Width) div 256;
            ry:=(j*chart1.height) div 256;
            if symb[k]=2 then //must check for the distance for teh ellipses
              begin
                x:=(series1.MAXXValue-series1.MinXValue)/2;
                y:=(series1.MAXyValue-series1.MinyValue)/2;
                if ((sqr(sizey-j-series2.yValue[k])/sqr(y)) + (sqr(i-series2.xValue[k])/sqr(x)))<=1
                  then m:=k
                else m:=-1;
              end
            else m:=series4.clicked(rx,ry); //this is for arbitrary shape
            if m<0 then continue;
            l:=addr(vmask[j,i]);
            l^:=l^ or (1 shl m); //mark as binary number up to 31, cannot have more that 32 symbols
          end;
      series4.shapes.Polygon[k].Points.clear;//clear in the loop
    end;
```
for k:=0 to series4.Shapes.count-1 do //restore all
begin
  if k>31 then continue;//no more than 31 symbols
  series4.shapes.Polygon[k].Points.clear;//clear in the loop
  for j:=0 to sides[k] do { add the XY coordinates }
  Series4.Shapes.Polygon[k].AddXY(px[k,j],py[k,j]);
end;
series1.clear;
for j:=0 to sides[polygonindex] do series1.addxy(px[polygonindex,j],py[polygonindex,j]);
showseries3(polygonindex);
chart1.refresh;
end;

This procedure save the mask to be used by the rest of the program
procedure Tbmform.Savemaskinmaskc01Click(Sender: TObject);
var i,j:integer;
begin
  for j:=0 to sizey do
  for i:=0 to sizex do
    begin
      maskc0^[j,i]:=trunc(vmask[j,i]);
      if trunc(vmask[j,i])>=0
      then maskb0^[j,i]:=0
      else maskb0^[j,i]:=-1;
    end;
  frameform.bubbleseries3.pointer.style:=pshexagon;
  frameform.showROI1.checked:=true;
end;

This procedure get the mask to be saved for the rest of the program
procedure Tbmform.GetROI1Click(Sender: TObject);
var  i,j,x,y:integer;
data:mydatasingle;
begin
  ShowROI1Click(Sender);//this set the mask with boolean
  move(plotarray,data,sizeof(mydatasingle));
  x:=sizex;
  if x>256 then x:=256;
  y:=sizey;
  if y>256 then y:=256;
  for j:=0 to y-1 do
  for i:=0 to x-1 do
    begin
      l:=addr(vmask[j,i]);
      if l^=-1 then data[j,i]:=data[j,i]*0.6; //only test if the pixel has a mask
    end;
  drawdata(chart1.BackImage.bitmap,min,max,@data,x,y);
  chart1.refresh;
Save mask in mask C01 Click(Sender); // this saves the masks
end;

procedure TForm.ShowROI1Click(Sender: TObject); // this routine sets the boolean condition by
marking the 31 bit
var  i,j,k,kk,x,y,m:integer;
    data:mydatasingle;
    h,lo:single;
    test:array[0..32] of boolean;
    b:boolean;
begin
  set vmask(Sender);
  x:=sizex;
  if x>256 then x:=256;
  y:=sizey;
  if myshowroi then h:=8 else h:=0;
  lo:=-1;
  if y>256 then y:=256;
  if booleancondition in [0] then for j:=0 to 32 do test[j]:=true;
  if booleancondition in [1,2] then for j:=0 to 32 do test[j]:=false;
  for j:=0 to y-1 do
    for i:=0 to x-1 do
      begin
        l:=add r(vmask[j,i]);
        if l^<0 then data[j,i]:=lo else
          begin
            m:=l^;
            for k:=0 to kk-1 do
              test[k]:=(m and (1 shl k))>0;
            case booleancondition of
            0:begin
                b:=true;
                for k:=0 to kk-1 do b:=test[k] and b;
              end;
            1:begin
                b:=false;
                for k:=0 to kk-1 do b:=test[k] or b;
              end;
            2: begin
                b:=false;
                for k:=0 to kk-1 do b:=test[k] xor b;
              end;
          end;//of case
        if b
          then data[j,i]:=h
          else data[j,i]:=lo;
      end;
end;
if data[j,i]=h then
  \( l^\land := l^\land \) or \( $80000000 \); // set bit 31 to mark that the condition is met
end;
end;
begin
  drawdata(chart1.BackImage.bitmap,lo,h,@data,x,y);
  chart1.refresh;
end;
end;

The function used by the SimFCS to process an image only in a selected mask is called in_cursor3.

For the cases psrectangle and pscircle, a geometrical region is used with a given center and radius. For the arbitrary shape, the region is defined by the pshexagon contour which is written in the maskb0\(^\land\) mask image by writing 0 in the region selected and -1 in the region not selected.

The mask is set in the bitmap library by using the “Save mask in maskc0” button in the tools menu of this library. In this routine both the maskc0 and the maskb0 are set starting form vmask[j,i]. The vmask (mydatadataset) is a public variable of bitmaplibrary (bmform) while the variables maskb0 and maskc0 are global variables of type image integer 32 of mdiframe.

The “Get ROI” button is used to set the vmask variable in the bitmap library starting from a pattern defined by pressing the right click mouse button while in the library and selecting the region of the mask.