featureData.add( X\_totalWhitePixels );

featureData.add( X\_whiteWidth );

featureData.add( X\_whiteHeight );

featureData.add( X\_maxObjectWidth );

featureData.add( X\_maxObjectHeight );

featureData.add( X\_avgObjectHeight );

double X\_totalWhitePixels = totalWhitePixels( pic );

double X\_whiteWidth = whiteWidth( pic );

double X\_whiteHeight = whiteHeight( pic );

double X\_maxObjectWidth = maxObjectWidth( pic );

double X\_maxObjectHeight = maxObjectHeight( pic );

double X\_avgObjectHeight = avgObjectHeight( pic );

/\*\*

\* [B/W] Feature based on the ratio of the average object height (based on white pixels)

\* against the height of this picture

\*

\* @param pic The instance being looked at (the picture)

\* @return double The ratio of the average object height based on white pixels

\* divided by the height of the Picture

\*/

public double avgObjectHeight( Picture pic ) {

Pixel[][] pixels = pic.getPixels2D();

ArrayList<Double> heights = new ArrayList<Double>();

double totalHeight = pixels.length;

for( int col = 0; col < pixels[0].length; col++ ) {

double colHeight = 0.0;

double firstWhitePixelRow = pixels.length;

double lastWhitePixelRow = 0.0;

boolean firstWhitePixelFound = false;

for( int row = 0; row < pixels.length; row++ ) {

if( !firstWhitePixelFound && pixels[row][col].getColor().equals( Color.WHITE ) ) {

if( row < firstWhitePixelRow )

firstWhitePixelRow = row;

firstWhitePixelFound = true;

}

if( pixels[row][col].getColor().equals( Color.WHITE ) )

if( row > lastWhitePixelRow )

lastWhitePixelRow = row;

}

colHeight = lastWhitePixelRow - firstWhitePixelRow;

if( colHeight > 0 )

heights.add( colHeight );

}

int size = heights.size();

double sumOfHeights = 0.0;

for( int rep = 0; rep < size; rep++ )

sumOfHeights += heights.get(rep);

return (sumOfHeights / (double)size) / (double)totalHeight;

}

/\*\*

\* [B/W] Feature based on the ratio of the object width based on white pixels

\* against the width of this picture

\*

\* @param pic The instance being looked at (the picture)

\* @return double The ratio of the object width based on white pixels

\* divided by the width of the Picture

\*/

public double maxObjectWidth( Picture pic ) {

Pixel[][] pixels = pic.getPixels2D();

double maxWidth = 0.0;

double firstWhitePixelCol = pixels[0].length;

double lastWhitePixelCol = 0.0;

double totalWidth = pixels[0].length;

for( int row = 0; row < pixels.length; row++ ) {

double lineWidth = 0.0;

boolean firstWhitePixelFound = false;

for( int col = 0; col < pixels[0].length; col++ ) {

if( !firstWhitePixelFound && pixels[row][col].getColor().equals( Color.WHITE ) ) {

if( col < firstWhitePixelCol )

firstWhitePixelCol = col;

firstWhitePixelFound = true;

}

if( pixels[row][col].getColor().equals( Color.WHITE ) )

if( col > lastWhitePixelCol )

lastWhitePixelCol = col;

}

lineWidth = lastWhitePixelCol - firstWhitePixelCol;

if( lineWidth > maxWidth )

maxWidth = lineWidth;

}

return maxWidth / totalWidth;

}

/\*\*

\* [B/W] Feature based on the ratio of the object height based on white pixels

\* against the height of this picture

\*

\* @param pic The instance being looked at (the picture)

\* @return double The ratio of the object height based on white pixels

\* divided by the height of the Picture

\*/

public double maxObjectHeight( Picture pic ) {

Pixel[][] pixels = pic.getPixels2D();

double maxHeight = 0.0;

double firstWhitePixelRow = pixels.length;

double lastWhitePixelRow = 0.0;

double totalHeight = pixels.length;

for( int col = 0; col < pixels[0].length; col++ ) {

double colHeight = 0.0;

boolean firstWhitePixelFound = false;

for( int row = 0; row < pixels.length; row++ ) {

if( !firstWhitePixelFound && pixels[row][col].getColor().equals( Color.WHITE ) ) {

if( row < firstWhitePixelRow )

firstWhitePixelRow = row;

firstWhitePixelFound = true;

}

if( pixels[row][col].getColor().equals( Color.WHITE ) )

if( row > lastWhitePixelRow )

lastWhitePixelRow = row;

}

colHeight = lastWhitePixelRow - firstWhitePixelRow;

if( colHeight > maxHeight )

maxHeight = colHeight;

}

return maxHeight / totalHeight;

}

/\*\*

\* [B/W] Feature based on the ratio of black to white pixels

\*

\* @param pic The instance being looked at (the picture)

\* @return double The ratio of white pixels to total pixels

\*/

public double totalWhitePixels( Picture pic ) {

Pixel[][] pixels = pic.getPixels2D();

double totalWhite = 0.0;

double totalPixels = pixels.length \* pixels[0].length;

for( int row = 0; row < pixels.length; row++ )

for( int col = 0; col < pixels[0].length; col++ )

if( pixels[row][col].getColor().equals( Color.WHITE ) )

totalWhite++;

return totalWhite / totalPixels;

}

/\*\*

\* [B/W] Feature based on the ratio of the largest width of continuous

\* white pixels of the object detected against the width of this picture

\*

\* @param pic The instance being looked at (the picture)

\* @return double The ratio of the largest band of white pixels in a row

\* divided by the width of the Picture

\*/

public double whiteWidth( Picture pic ) {

Pixel[][] pixels = pic.getPixels2D();

double maxWidth = 0.0;

double totalWidth = pixels[0].length;

for( int row = 0; row < pixels.length; row++ ) {

double lineWidth = 0.0;

for( int col = 0; col < pixels[0].length; col++ ) {

if( pixels[row][col].getColor().equals( Color.WHITE ) )

lineWidth += 1.0;

}

if( lineWidth > maxWidth )

maxWidth = lineWidth;

}

return maxWidth / totalWidth;

}

/\*\*

\* [B/W] Feature based on the ratio of the largest height of continuous

\* white pixels of the object detected against the height of this picture

\*

\* @param pic The instance being looked at (the picture)

\* @return double The ratio of the largest band of white pixels in a column

\* divided by the height of the Picture

\*/

public double whiteHeight( Picture pic ) {

Pixel[][] pixels = pic.getPixels2D();

double maxHeight = 0.0;

double totalWidth = pixels[0].length;

for( int col = 0; col < pixels[0].length; col++ ) {

double colWidth = 0.0;

for( int row = 0; row < pixels.length; row++ ) {

if( pixels[row][col].getColor().equals( Color.WHITE ) )

colWidth += 1.0;

}

if( colWidth > maxHeight )

maxHeight = colWidth;

}

return maxHeight / totalWidth;

}