1. Objective:

The objective of our project is to extract urban areas in aerial images captured from Google Maps by a simple screenshot. The proposed image processing is suitable for images with the same resolution as the images given as examples, that is 1 pixel represents 5,932 m on the ground. With a different resolution, the parameter values will be adjusted. A sample image is shown in Figure 1.

Figure 1: Input Image

2. Introduction:

Urbanization often occurs in an unplanned and uneven manner, resulting in profound changes in patterns of land cover and land use. Understanding these changes is fundamental for devising environmentally responsible approaches to economic development in the rapidly urbanizing countries of the emerging world. One indicator of urbanization is built-up land cover that can be detected and quantified at scale using satellite imagery and cloud-based computational platforms.

3. Tools and Platforms:

We have used MATLAB as our working platform. Being a general purpose programming language it provides many important advantages for image processing. It ensures the image processing steps used are completely documented, and hence can be replicated. In general, the source code for all image processing functions are accessible for scrutiny and test. The MATLAB code of our project is added at the end of the report as appendix.

4. Process to solution:

The method is based on the assumption that urban areas are characterized by large areas .Thus, the algorithm is straightforward:

1. Reducing noise and JPEG artifacts,
2. Detecting true edges ,
3. Localizing regions with high density of edges ,
4. Forming the candidate regions ,
5. Removing spurious regions ,
6. Superimposition of the urban area boundaries to the original image.

The details of our work will be discussed here:

4.1 Reducing noise and JPEG artifacts:

The color is not an intrinsic characteristic of urban areas. For example, the house roofs can be of any color. In image processing, one very important task is to remove white noise, all the while maintaining salient edges .This can be a contradictory task - white noise exists at all frequencies equally, while edges exist in the high frequency range .In traditional noise removal via filtering, a signal is low pass filtered, which means that high frequency components in the signal are completely removed. But if images have edges as high frequency components, traditional laplacian filter will also remove them, and visually, this manifests itself as the edges becoming more smudged .To remove noise, but also preserve high frequency edges we will use **Gaussian low pass filtering.** So, we only use the lightness component of the colors. To remove noise, but also preserve high frequency edges we have used Gaussian low pass filtering.

At first we convert our aerial image into grayscale image. Then we have used Gaussian filter on our image . Gaussian filters are generally isotropic, that is, they have the same standard deviation along both dimensions. An image can be filtered by an isotropic Gaussian filter by specifying a scalar value for sigma. Here we have used sigma=0.5 .After using Gaussian filter we get this image :

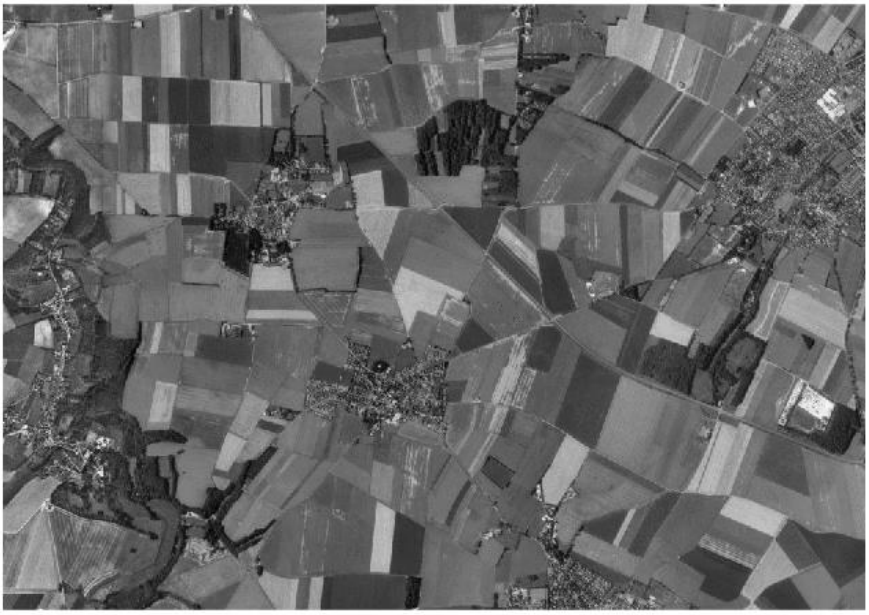

Figure 2: After applying Gaussian filter with sigma=0.5.

4.2 Detecting true edges:

After removing noise from image ,now we have to detect edges in our image .Urban areas are regions with a high density of edges due for example by house or street borders, while other regions of the image (eg. fields, water) are smoother. We have threshold edge magnitudes to produce binary image. If we use low threshold, we get lots of edge point if we use high threshold we get lower edge points. But the problem is using a high threshold, we get edge points that are more than one pixel wide along a contour. So what we would really like is for a line in the scene to have a one pixel line edge in the edge. So for this purpose, we have used simple image gradient followed by threshold based on inter-class variance maximization to detect the contours. After using gradient filter we have got the following image :

Figure 3: Edge detection.

4.3 Localizing regions with high density of edges:

To do this we used basic morphological operation Dilation and Erosion. After edge detection we have converted the grayscale image into binary image. Then using Dilation we enlarged the foreground and shrank the background first to expand the regions. It smooths object boundaries and closes the holes and gaps. Here we have used 5*5 structuring element of diamond shape. After that we used Erosion to shrink the size of foreground. In erosion we used 7*7 disk shape structuring element. It removes small objects which are really not urban areas. But all small objects are not removed yet.

We consider that two houses have borders at a maximum distance of 5 pixels in the image. So a simple dilation of 3 pixels on both side of the edges is sufficient to merge two edges at a distance <= 5 pixels. To locate the regions from the blobs (Binary Large Object), it is necessary to separate the blobs from each others by removing the links between them (e.g : street, edge).After applying dilation and erosion we get the following image:

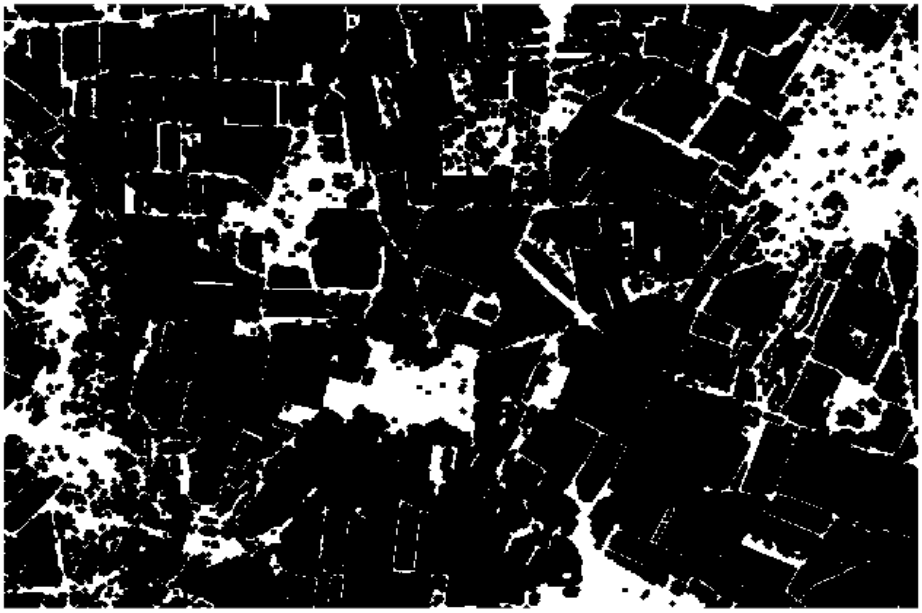

Figure 4: After dilation and erosion

After getting the eroded image, we apply top hat filter on that image. We have used top hat filter to remove uneven background illumination from the image . Now we subtracts the top hat filtered image from our eroded image to remove small unnecessary objects.

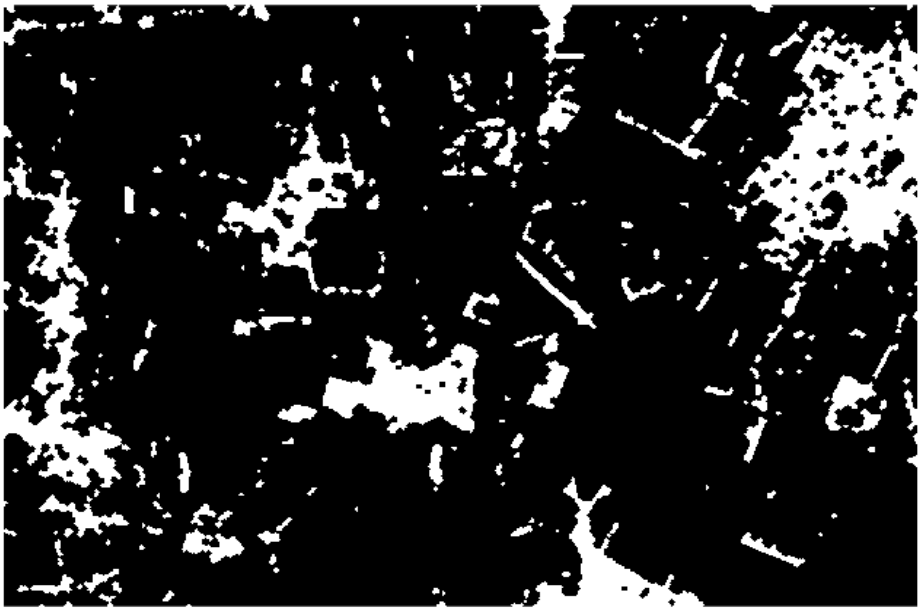

Figure 5: Subtracted image

As we have seen from Figure 5, many un-necessary edges have been removed.

4.4 Forming the candidate regions:

Regions are formed with blob in which the holes are filled. However, only regions with area surface less than 10 x 10 pixels (60 x 60 m^2 on the ground) are filled. Generally, larger holes correspond to green spaces or football stadiums within the urban area that should not be agglomerated in the urban area.

To complete these tasks, we once again perform dilation then erosion. In other words, we have performed closing operation. After performing the closing operation, we get the following image :

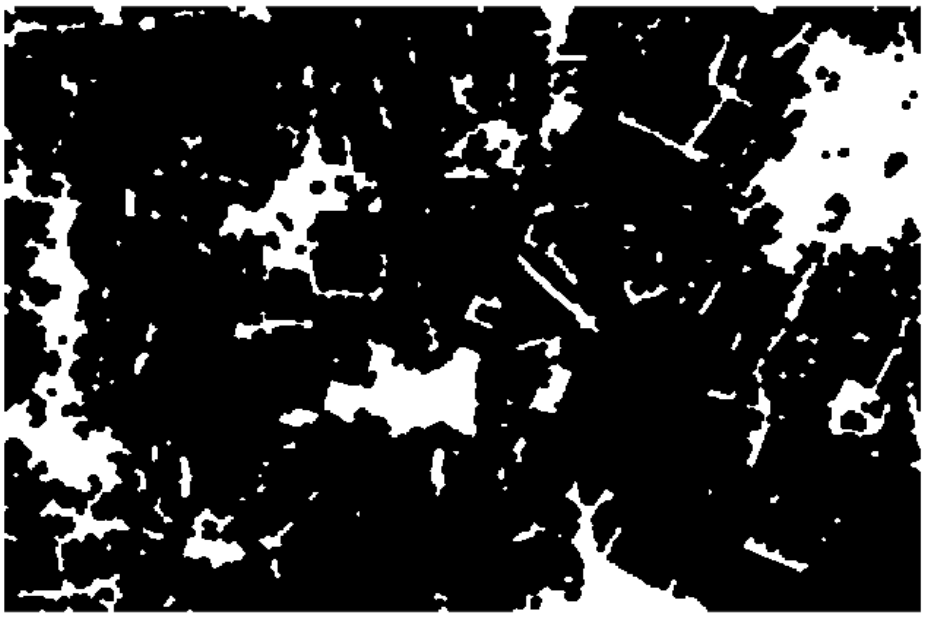

Figure 6 : After performing closing operation.

From figure 6 we can see that, many small black regions is removed from the candidate regions.

4.5 Removing Spurious Regions:

Small regions do not correspond to urban areas. So we will eliminate them using size criteria. We have used bwareaopen() to remove these small region. bwareaopen(BW,P) removes all connected components (objects) that have fewer than P pixels from the binary image BW. The default connectivity is 8 for two dimensions.We remove connected components that have fewer than 4000 pixels. After removing small we got the following image:

Figure 7: After removing small regions.

4.6 Superimposition of the urban area boundaries to the original image:

After removing small regions using size criteria , we get our final urban areas. This final step includes the boundaries in the original image.We have traced the exterior boundaries of objects, as well as boundaries of holes inside these objects, in the binary image. Then we have given some color to the regions.

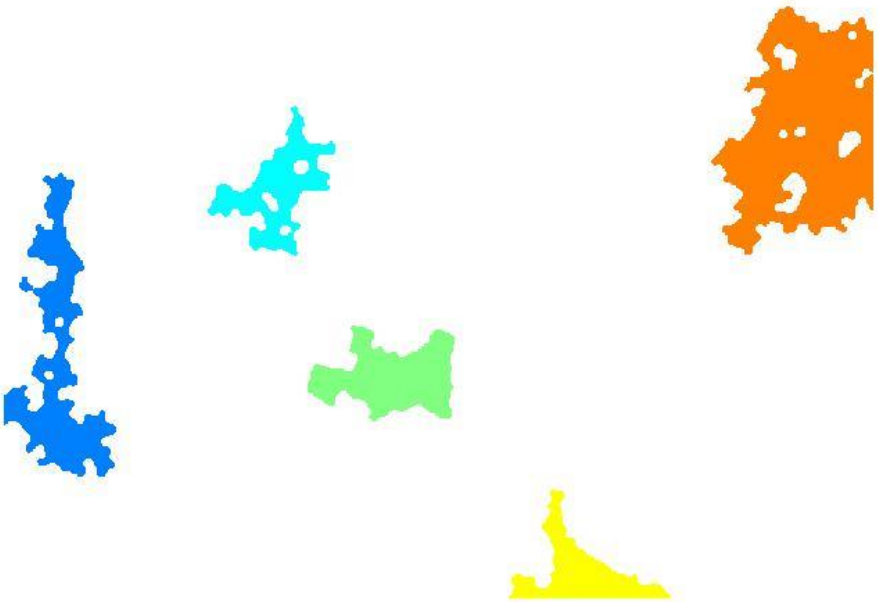

Figure 8: Selected urban areas.

After that we have plotted this image onto our main image using imshowpair().It creates a visualization of the differences between input image and boundary image.

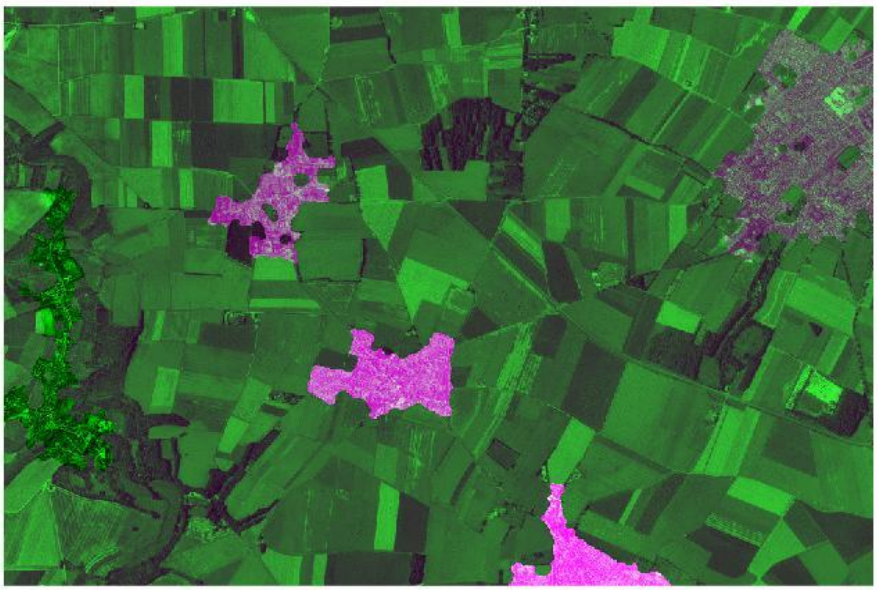

Figure 9: Superimposition of boundaries to the original image

5. Conclusion:

The increasing availability of satellite imagery at different spatial and temporal resolutions has shifted urban research towards the use of digital, multispectral images. Using digital image processing, It is possible to expand the spatial and temporal units of analysis and to investigate urbanization processes over larger areas and over longer periods of time. In our project we will try to detect the urban areas as accurately as possible. There are several stage left. After that our program can detect urban areas from Google map aerial images.

APPENDIX

MATLAB CODE

```
I = im2double(rgb2gray(imread('IP1.png')));

subplot(3,3,1);
imshow(imread('IP1.png'));
title('Input Image');

subplot(3,3,2);
I1 = imgaussfilt(I,0.5);
imshow(I1);
title('Gaussian filterd Image');

subplot(3,3,3);
I2=imgradient(I1);
imshow(I2);
title('Gradient image');

subplot(3,3,4);
BW = im2bw(I2,0.4);
imshow(BW);
title('Coverting to binary image using threshold');

se = strel('diamond',2);
dilatedBW = imdilate(BW,se);


 se1=strel('disk',4);

 erodedBW = imerode(dilatedBW,se1);


 sn=strel('disk',2);
 topht=imtophat(erodedBW,sn);
 newimg=erodedBW-topht;

 subplot(3,3,5);
 imshow(newimg);
 title('After First round of dilation and erosion');

sd2=strel('disk',5);
dilatedBW2 = imdilate(newimg,sd2);

se2=strel('disk',5);
erodedBW2 = imerode(dilatedBW2,se2);


sn=strel('octagon',3);
topht=imtophat(erodedBW2,sn);


newimg2=erodedBW2-topht;
```

```
 subplot(3,3,6);
 imshow(newimg2);
title('After second round of dilation and erosion');

final = bwareaopen(newimg2 ,4000);

subplot(3,3,7);
imshow(final);
title('Removing small regions');

[B,L] = bwboundaries(final,'noholes');
BB=label2rgb(L, @jet, [.5 .5 .5]);

subplot(3,3,8);
imshow(label2rgb(L, @jet, [1 1 1]));
hold on
title('Coloring the regions');
%figure;
for k = 1:length(B)
   boundary = B{k};
   plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end

subplot(3,3,9);
imshowpair(I,BB);
title('Output image');

[row,col]=size(final);
totalSize=row*col;
WhiteReg=sum(final(:));
urbanArea=(WhiteReg*100)/totalSize;
fprintf('Urban area: %d percent',urbanArea);
```