CSE - 527 Final Project Report

Alok Katiyar , Onkar Pednekar, Amit Sahasrabudhe

Group Members:

1) Amit Sahasrabudhe (108738814)
2) Onkar Pednekar (108601495)
3) Alok Katiyar (108943744)

Project Description

Shadow detection is an important computer vision field since shadows can be used to deduce useful cues for determining the 3D properties of objects like shape, depth etc. Different kinds of light sources that illuminate objects make it difficult to derive a unified shadow detection algorithm for all circumstances. We have implemented the methodology devised by Huang et al in their paper for shadow detection in outdoor scenes illuminated by only bright sunlight. Such detection can be quite useful in the analysis of outdoor pictures as in the case of video surveillance. The detailed paper can be found at the location:

http://www.cs.northwestern.edu/~xhu414/papers/11iccv_shadow.pdf

The plan of action has been to first analyze a number of training images containing shadows created by the blocking of direct sunlight. From these images we plan to extract a number of features which can effectively distinguish a pixel of lying on the boundary of a shadow from a nom shadow pixel. We then modeled all these features in the form of a feature vector. Feature vectors are formed for both shadow and non-shadow boundary pixels. These pixels were then fed into the Naïve Bayes classifier to train a model for distinguishing shadow boundary pixels from non-shadow boundary pixels.

Procedure Used

We begin the process with applying the canny edge detection algorithm to filter out the candidate shadow edge points. These points were then matched with the ground truth given in the XML's provided in the dataset and then classified into shadow and non- shadow points using the Naïve Bayes classifier. We used WEKA toolkit for training the classifier. We trained the classifier using the feature points extracted from the images. The entire procedure can be subdivided into two phases: 1) Feature extraction 2) Training the classifier. The detail of feature point extraction is given below:

Phase 1: Feature Extraction

Selecting visual features that can distinguish shadow boundary pixels from non-boundary pixels is of utmost importance. Four such visual features have been defined in the paper that can be used to train the classifier using the shadow boundary pixels from the images of the database. These features will also be used once we try to segment shadows in our input image.

Implementation details for feature point extraction:

As a first step in computing the features we find out the x and y components of the gradient at each and every pixel location of the image as a whole (by converting it to grayscale) and also of the individual colored channels. The function that computes this is: “color_gradient(in)”, where “in” is the image that has been read into the matlab. Thereafter all the four features are computed using the function:

returnColorChannelWeight(img , xGrad, yGrad,Hx, Hy, Ix, Iy, Jx, Jy,out,Rgrad,Ggrad,Bgrad)

where img : is the image that has been received into matlab

xGrad : ‘xGrad’ is a matrix containing the x component of the image gradient at every point in the image.

yGrad : ‘yGrad’ is a matrix containing the y component of the image gradient at every point in the image.

Hx : ‘Hx’ is a matrix containing x component of the image gradients at each and every point in the red channel of the image

Hy : ‘Hy’ is a matrix containing y component of the image gradients at each and every point in the red channel of the image

Ix : ‘Ix’ is a matrix containing x component of the image gradients at each and every point in the green channel of the image

Iy : ‘Iy’ is a matrix containing y component of the image gradients at each and every point in the green channel of the image

Jx : ‘Jx’ is a matrix containing x component of the image gradients at each and every point in the blue channel of the image

Jy : ‘Jy’ is a matrix containing y component of the image gradients at each and every point in the blue channel of the image

out : ‘out’ is a matrix containing the candidate shadow edges

Rgrad : Is a matrix containing the magnitude of the gradient at each and every point in the Red channel of the image.

Ggrad : Is a matrix containing the magnitude of the gradient at each and every point in the Green channel of the image.

Blue : Is a matrix containing the magnitude of the gradient at each and every point in the Blue channel of the image.

For calculation of gradient we have used the 'Sobel' Operator

1) For calculating the first visual feature we have calculated the pixel intensity values across the edge. We compute the Gaussian weighted average of the pixels on both the sides(the light side and the dark side) of the edge for the red, green and the blue channel. Say the intensity value is H_r , H_b , H_g for the brighter side and L_r , L_b ,L_g for the lighter side . Then we compute the intensity ratio t_r, t_g, t_b as $(L_r / H_r, L_g / H_g , L_b / H_b)$. For a shadow it is reasonable to assume that it is illuminated by only the sky and not the sun so it will be more blue than the brighter side so in case of a shadow boundary pixel

$$t_r > t_g > t_b$$

Finally, we compute the features to be fed into the vector as
$(t, t_{rg}, t_{gb}) = ((t_r + t_g + t_b)/3, t_r/t_b, t_g/t_b)$.

To compute this feature we have adopted the following methodology in our code. We compute direction of the image gradient at the candidate edge pixel and take a window in the direction of the gradient as well as the negative gradient direction and compute the Gaussian Weighted average on the brighter and the darker side of each of the colored channel.

2) The second visual feature is devised by calculating the gradient at each candidate edge pixel . We compute the gradient for all the three colors at a candidate edge pixel. Let these gradients be (s_r, s_g, s_b) . We compute the second feature by calculating the ratio $(\delta r, \delta g, \delta b) = (s_r/H_r, s_g /H_g, s_b / H_b)$

In our code we compute the gradient magnitude at the particular candidate edge for every colored channel and then dividing it by H_r which is computed using the methodology as given in 1 above.

3) The third visual feature is the gradient direction for all the three colors at a candidate edge pixel. For a possible shadow edge pixel all the three gradients are collinear and in the same direction. The difference of the gradient for each pair of color is to be fed into the SVM which will uses this data to classify the candidate edge pixels from a test image into shadow edge pixel and non-shadow edge pixel. The feature is calculated as $(\gamma_{rg}, \gamma_{gb}, \gamma_{br})$, where $\gamma_{rg} = \min(|\gamma_r - \gamma_g|, 2\pi - |\gamma_r - \gamma_g|)$.

This feature is also extracted by taking the direction of the gradient at a candidate edge pixel for each of the three colored channels and then performing the computations as mentioned in three above.

4) Fourth visual feature is derived from the edge width value for each color channel (w_r, w_g, w_b) A shadow edge would have the same width foe all the three channels. We compute this fourth feature vector as (w, w_{rg}, w_{rb}), where $w = (w_r + w_g + w_b)/3$, $w_{rg} = |w_r/w_g|$, wrb = $|w_r/w_b|$.

To compute the fourth feature we resolve gradient direction of each of the colored channel into the following directions in order to facilitate computation in a discrete setting:

1) -22.5 to 22.5 will be treated as lying along the horizontal
2) 22.5 to 67.5 will be considered lying along the positive 45 degree line
3) 67.5 to 112.5 will be considered lying along the positive vertical
4) 112.5 to 157.5 will be considered lying along the 135 degree line
5) 157.5 to -157.5 will be treated as lying along the horizontal
6) -22.5 to -67.5 will be considered lying along the negative 45 degree line
7) -67.5 to -112.5 will be considered lying along the negative vertical
8) -112.5 to -157.5 will be considered lying along the negative 135 degree line

Note that the term negative over here implies measurement in the clockwise direction.
After classifying the directional category we take a strip along the particular direction and then we find out the pixel up to which the intensity continuously increases (let the number of pixels be a) in that direction. Then in the opposite direction we compute the pixel upto which the intensity continually decreases (let the number of pixels be b). We add the the two values a and b, then get the edge width for a particular color channel.

For each edge pixel we build a feature vector at three scales, the vector contains the following elements: $(t, t_{rb}, t_{gb}, \delta_r, \delta_g, \delta_b, \gamma_{rg}, \gamma_{gb}, \gamma_{br}, w, w_{rg}, w_{rb})$.

Note that scaling over here implies taking a different Gaussian Kernel and not a scaled down image. For computing the 36D feature vector we use the 3×3, 5×5 and 7×7 kernel

We have used the training data set provided by Zhu et al to train the Naïve Bayes classifier for shadow boundary detection using the features extracted for both shadow and non shadow edges.

Observations regarding the edge width (the fourth feature)

As asked by Tomas, we have utilized our method of calculating the edge width. For this we have made the function “widthCalculator” .

Instructions for using the function:

The following two files have been provided in the bundled project package

1) 1.txt : Corresponds to the image ‘zhu-003.jpg'
2) 3.txt : Corresponds to the image ‘zhu-009.jpg'

Both these files contain the x and y co-ordinates of the shadow edge pixels fetched from the XML’s provided from the database. Our function computes the edge width at these pixels if they are currently detected in the canny applied on our image.

Note that the 4th column is the y co-ordinate of the image and the fifth one is the ‘x’ co-ordinate of the image.

Entering the following commands at the command line would generate a matrix B which can be computed for any image , by preparing files similar to ‘1.txt’ and ‘2.txt’. The commands are:

```
A = imread('zhu-003.jpg');
[out Hx Hy Ix Iy Jx Jy gradX gradY Rgrad Ggrad Bgrad]=color_gradient(A);
[ B ] = widthCalculator( '1.txt',A , gradX, gradY,Hx, Hy, Ix, Iy, Jx,
Jy,out,Rgrad,Ggrad,Bgrad );
```

We have imported the matrices into two separate ‘.mat’ files and bundled them in the “MAT files” folder. The .mat files are named ‘zhu03.mat’ and ‘zhu09.mat’

From our observation we feel that the authors claim is correct at most of the shadow edge points at least using our calculation methodology. The edge width is the same or nearly the same for the three color channels. Though, there are some points where there is a huge aberration, but these points are say 3 in every 10.

Please find below the way each column of the results is to be read:

1) Column 1: Image width in the red channel
2) Column 2: Image width in the green channel
3) Column 3: Image width in the blue channel
4) Column 4: Y co-ordinate of the image
5) Column 5: X co-ordinate of the image

Brief description of all the functions used in feature calculation:

1) **color_gradient:** This function returns the ‘x’ and ‘y’ component of the overall gradient as well as the ‘x’ and ’y’ components of the gradient in each of the three separate color channel. This function uses the “Sobel” operator to compute the gradient.

2) **returnColorChannelWeight:** This function returns the features at each of the candidate edge pixels. These features are returned in a matrix ‘B’ which is basically m×38 matrix , where m depends on the number of candidate edge pixels.

This matrix is later parsed in a ‘.txt’ file using the following code snippet :

```
f='zhu-labelme_0059.txt';
fid=fopen(f,'w');
for a=1:size(B,1)
for b=1:size(B,2)
fprintf(fid,'%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f
%f %f %f %f %f %f %f %f %f %f %f %f %f %f\n',B(a,b,:));
end
end
fclose(fid);
```

It is these text files that are parsed and fed into the classifier.

All the codes used are bundled in the material attached with the final submission.

Phase 2: Training the classifier

Our Feature vector consisted of 36D features for each pixel (shadow and non shadow pixel). We used the Naïve Bayes classifier to train and generate the model for classification. The tool kit used for classification was WEKA. WEKA is a collection of machine learning algorithms for data mining tasks. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. We used it for classification task. The input to classifier is an arff file which contains data in a specific format. Arff file contains the 36D features as the attributes and each pixel point is the data.

Generation of arff file

Our dataset consisted of 135 images, but for implementation feasibility we considered 62 images. We divided the 62 image dataset into training and testing data. 42 images were used for training and 20 images for testing. We generated arff file for both the training and testing corpus.

Elements of arff file:

- **Attributes:** Both the training and testing arff files contain same 36D features as attributes. Attribute names were assigned serially from 1-36. Format of the attributes was kept numeric.

- **Data:** Each tuple of the dataset consists of 36 features for each edge pixel (shadow and non-shadow). The shadow pixels were directly extracted from the xml data file which was provided in the predefined dataset.

  The shadow pixel values in the xml file were as follows:
  <1.5,272.5> <2.5,272.5> <3.5,272.5> <4.5,272.5> <5.5,272.5> <6.5,272.5> <6.5,273.5>

  Whereas the corresponding pixel values that we obtained using canny edge detector were as follows:
  <2.0,273.0> <3.0,273.0> <4.0,273.0> <5.0,273.0> <6.0,273.0> <7.0,273.0> <8.0,274.0>

  (i.e. all the pixel values were integer values, since in MATLAB images are matrices and matrices can only be represented by integer value)

  Hence, in order to match the corresponding pixels, we rounded off the pixel values in the xml file to the next integer value.

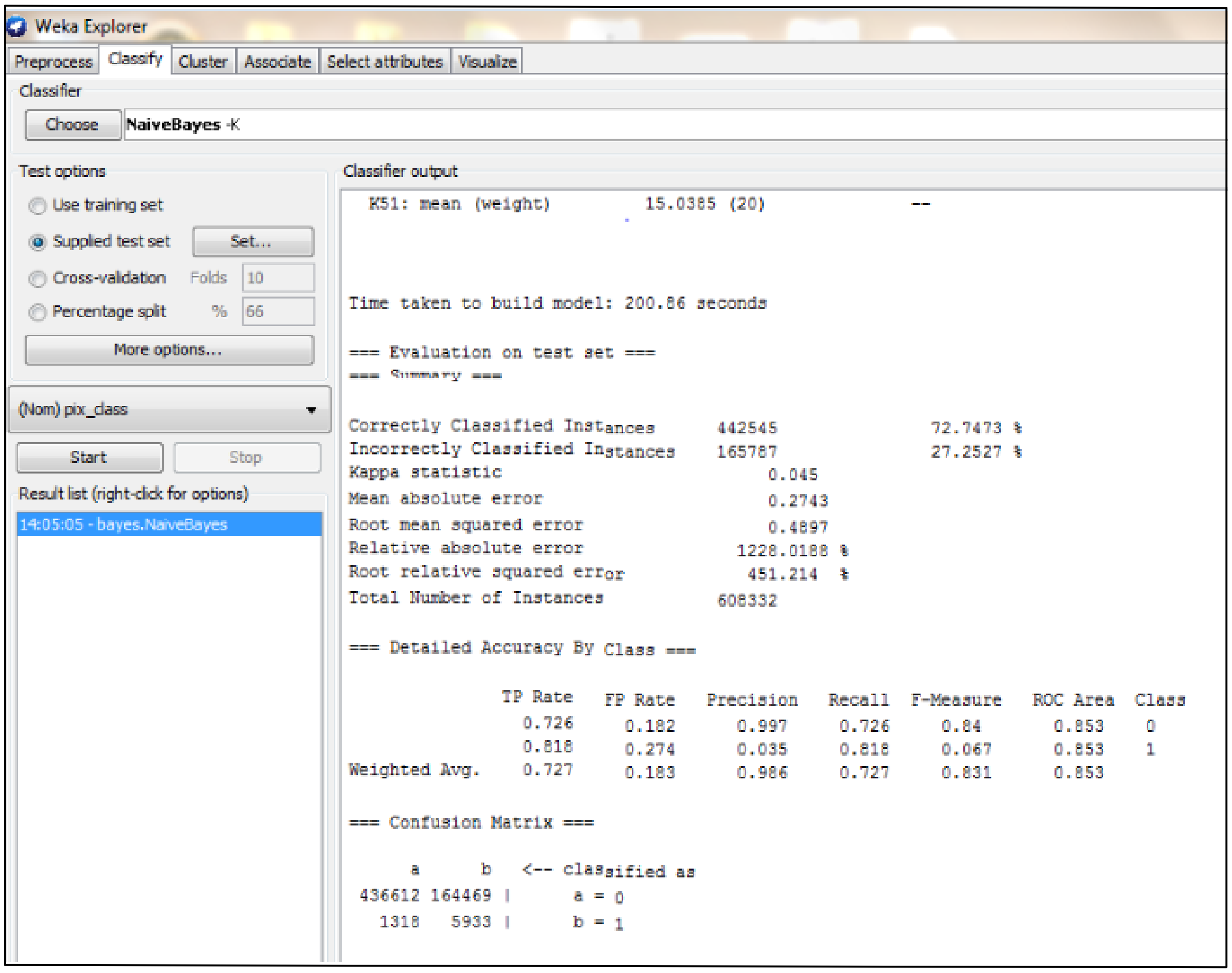
- **Class:** Since this is a classification task, we had 2 classes, either the pixel is a shadow (class 1) or it is non shadow (class 0).

The coding for arff file generation was done in JAVA.

Summary of code:

- We first extracted all the shadow coordinate pixels from the xml file that was provided with the dataset.
- Next, for each image, we compared the extracted shadow pixels with the pixels that we generated for that corresponding image using canny edge detector.
- All the pixel values that matched with the shadow pixel values were assigned class 1 and the remaining pixels were assigned class 0. (code that generates arff file is bundled in the material attached with the final submission).

Next, the training arff file was given as input to the classifier. We trained the model on the training dataset using the kernel estimator. Once the modeled was trained, we inserted the test arff file to classify the data based on the developed model. The classification process took considerable amount of time and produced the following result.



Classifier: NaiveBayes -K

Test options: [✓ Supplied test set] Folds 10, % 66

(Nom) pix_class

14:05:05 - bayes.NaiveBayes

```
  K51: mean (weight)         15.0385 (20)              --



 Time taken to build model: 200.86 seconds

 === Evaluation on test set ===
 === Summary ===

 Correctly Classified Instances       442545               72.7473 %
 Incorrectly Classified Instances     165787               27.2527 %
 Kappa statistic                           0.045
 Mean absolute error                       0.2743
 Root mean squared error                   0.4897
 Relative absolute error                1228.0188 %
 Root relative squared error             451.214  %
 Total Number of Instances            608332

 === Detailed Accuracy By Class ===

                TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                  0.726     0.182     0.997     0.726     0.84       0.853    0
                  0.818     0.274     0.035     0.818     0.067      0.853    1
 Weighted Avg.    0.727     0.183     0.986     0.727     0.831      0.853

 === Confusion Matrix ===

      a      b   <-- classified as
  436612 164469 |      a = 0
    1318   5933 |      b = 1
```

Snapshot of the WEKA classifier output

Result shows that accuracy that we obtained through this classification was 72.7%.

Confusion matrix:

a	b	←classified as
436612	164469	a = 0
1318	5933	b = 1

Here, class 1 corresponds to shadow pixels and class 0 corresponds to non-shadow pixels.
Total non shadow pixels under consideration were 601081.
Total shadow pixels under consideration were 7251.

Non shadow-pixels as it can be seen were very large as compared to shadow pixels because ideally in an image shadow edges are comparatively very less than non-shadow edges.

Confusion matrix depicts that 436612 out of 601081 non-shadow pixels (72.6%) were correctly classified and 5933 out of 7251 shadow pixels (81.8%) were correctly classified.

Issues Faced:

1) **Scaling issue:** The paper suggests calculating the set of 12 features on 3 different scales to get a total of 36 features. However, calculating the features in this way is a very complex and time consuming approach. Thus, instead of calculating the features on 3 different scales, we apply a different approach, wherein we change the size of the Gaussian window used to calculate the features. It was found that calculating features in this way expedited the process of feature extraction.

2) **Time required for training and testing the classifier:** Our dataset consist of total 135 images. Initially we planned to train the classifier on 110 images and test on 25. However, considering the amount of time required for training and testing the classifier, we reduced the training set to 42 images and we tested on 20 images. The main reason behind so much time consumption was the large numbers of edge pixels that were generated by applying canny edge detector. May be applying some different approach for edge detection would have generated redundant pixels and saved the training time.

3) **Accuracy:** It can be observed from the above confusion matrix that the accuracy of classifying pixel as a shadow pixel is 81.8 which is much better than the accuracy of classifying pixel as non shadow (72.6). The classification accuracy depends upon varied no of parameters, for example the size of the training/testing data set, accuracy of feature point extraction, ratio of shadow to non-shadow pixels in an image, etc. Thus tuning theses parameters can largely affect the classification accuracy. However, due to time constraint, we were not able to experiment with these parameters.

4) **Discrepancies in matching the shadow pixels:** As mentioned earlier, during the generation of arff files, we compared the shadow edge pixels from the xml dataset with the pixels that we generated using canny edge detector. However, not all the shadow pixels from the dataset matched with our pixels. ie. all the shadow edge pixels in the image were not detected. This might also have affected the accuracy classifier.

5) **Plotting the resulting shadow pixels on test image:** WEKA provided only visual classification result. We wanted to plot the pixels of the test image that were rightly classified as shadow pixels in MATLAB. However, we were unable to extract the classified pixels from WEKA, since the toolkit only generated the end result and not pixel by pixel outcome. We did not know how to extract these values from WEKA toolkit and due to time constraint we could not search for some other approach for pixel extraction.

Conclusion and Future Work:

The experiments performed and the results obtained are convincing enough to conclude that the set of features suggested by the authors to differentiate a shadow edge pixel from a non-shadow edge-pixel are indeed substantial. The shadow edge pixels identified as correct by the classifier can be extracted to form a set of true shadow pixels. These pixels can then be linked together with some edge-linking algorithm to form perfect shadow boundaries on an outdoor image illuminated by the Sun and the Sky.

The paper concentrated on the images lit by the Sun and the sky alone. Finding the features of shadows in indoor images still remains a challenge and forms a part of the future work to be researched.