YouCook Dataset

Chenliang Xu, Pradipto Das, Richard F. Doell, Philip Rosebrough and Jason J. Corso
Department of Computer Science and Engineering
SUNY at Buffalo
Buffalo, NY 14260-2500
{chenlian,pdas3,rfdoell,psr2,jcorso}@buffalo.edu

Document Dated: 4 June 2013 version 1

Abstract

The YouCook dataset contains videos of people cooking various recipes. The videos were downloaded from YouTube and are all in the third-person viewpoint; they represent a significantly more challenging visual problem than existing cooking and kitchen datasets (the background kitchen/scene is different for many and most videos have dynamic camera changes). In addition, frame-by-frame object and action annotations are provided for training data (as well as a number of precomputed low-level features). Finally, each video has a number of human provided natural language descriptions (on average, there are eight different descriptions per video). This dataset has been created to serve as a benchmark in describing complex real-world videos with natural language descriptions.

1 Description of the YouCook dataset

The YouCook dataset consists of 88 videos downloaded from YouTube and roughly uniformly split into six different cooking styles: baking, grilling, making breakfast, making sandwich, preparing salad and cooking in general. The videos all have a third-person viewpoint, take place in different kitchen environments, and frequently display dynamic camera changes. The training set consists of 49 videos with object and action annotations. The test set consists of 39 videos. All videos are annotated with multiple human descriptions obtained from Amazon Mechanical Turk.

When you use the YouCook dataset, please use the following citation:

P. Das, C. Xu, R. F. Doell, J. J. Corso. "A Thousand Frames in Just a Few Words: Lingual Description of Videos through Latent Topics and Sparse Object Stitching." Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2013.

```
@inproceedings{DaXuDoCVPR2013,
  author = {Das, P. and Xu, C. and Doell, R. F. and Corso J. J.},
  booktitle = {{Proceedings of IEEE Conference on Computer Vision
      and Pattern Recognition}},
  title = {A Thousand Frames in Just a Few Words: Lingual Description
      of Videos through Latent Topics and Sparse Object Stitching},
  year = {2013}
}
```

2 Organization of the YouCook dataset

The YouCook dataset is available for download at http://www.cse.buffalo.edu/~jcorso/r/youcook/. The primary download contains the original videos, annotations, and some features (details described below). This primary download is about 1.6GB. Due to the size of data, ***YouCook/VideoFrames*** and ***YouCook/Features/HOG3D/feature_vectors*** are optional for download (6GB and 2.8GB respectively).

The dataset is organized as follows:

YouCook/Videos
It contains 88 videos in MPEG-4 format. See Table 3 for training and test splits.

YouCook/VideoFrames
It contains the extracted frames for the 49 training videos. The IDs of frames are used in annotations of object tracks and actions. Download the package from http://www.cse.buffalo.edu/~jcorso/r/youcook/youcook_videoframes.tbz and extract the contents (from the root directory). Frame extracted can be reproduced with *mmreader* or *ffmpeg* (more details follow).

YouCook/Annotations
It contains following three kinds of annotations.

YouCook/Annotations/Object_Tracks Object tracks for the training set. See Section 3.1 for details.

YouCook/Annotations/Actions Actions for the training set. See Section 3.2 for details.

YouCook/Annotations/Human_Descriptions Human descriptions for all 88 videos. See Section 3.3 for details.

YouCook/Features
It contains two kinds of precomputed features: HOG3D and color histogram.

YouCook/Features/HOG3D

YouCook/Features/HOG3D/feature_vectors contains precomputed HOG3D features. Download the package from http://www.cse.buffalo.edu/~jcorso/r/youcook/youcook_hog3dfeatures.tbz and extract the contents (from the root directory).

YouCook/Features/HOG3D/codebook_1000.txt HOG3D codebook with 1K means obtained from the training videos.

YouCook/Features/HOG3D/hist contains 1000 dimensional histograms for each video extracted using the codebook.

YouCook/Features/ColorHist contains precomputed 512 dimensional color histogram features. See Section 4.2 for detail.

YouCook/Features/Code contains code to compute HOG3D (Matlab) and ColorHist (Java).

YouCook/Evaluation
It contains the ROUGE [Lin and Hovy, 2003] evaluation. See Section 5 for details.

YouCook/Evaluation/ROUGE-1.5.5 contains ROUGE evaluation tool.

YouCook/Evaluation/config.txt configuration file of ROUGE.

YouCook/Evaluation/summaries
It consists of files that are used for evaluating lingual descriptions generated for test videos by comparing them against the human descriptions.

YouCook/Evaluation/summaries/human_descriptions contains human descriptions of the text set.

YouCook/Evaluation/summaries/cvpr2013 contains the system output of our CVPR2013 paper [Das et al., 2013].

Dairy	Count
butter	6
milk	3
yogurt	2
Total	**11**

Utensils	Count
knife	13
pan	9
plate	9
spreader	9
cup	8
spatula	8
spoon	7
whisk	7
tongs	6
blender	2
brush	1
fork	1
pitcher	1
pot	1
Total	**82**

Meats	Count
chicken	7
fish	4
hen	2
meat	2
salmon	2
turkey	2
steak	1
Total	**20**

Misc Ing.	Count
oil	6
bread	5
egg	5
tomato	5
lemon	3
chocolate	1
croutons	1
doughnut	1
flour	1
sugar	1
Total	**29**

Bowls	Count
bowl	100
Total	**100**

Veg./Fruits	Count
lettuce	3
apple	2
carrot	2
cucumber	2
brocolli	1
corn	1
onion	1
Total	**12**

Condiments	Count
pepper	4
salt	2
creamcheese	1
jelly	1
mustard	1
peanutbutter	1
Total	**10**

Table 1: Annotated objects in seven categories.

3 Annotations

3.1 Object Tracks

The objects for YouCook are in the categories of bowls (38%), utensils (31%), misc ingredients (11%), meats (8%), dairy (4%), vegetables/fruits (4%) and condiments (4%) as shown in Figure 1. The specific objects in each category are listed in Table 1.

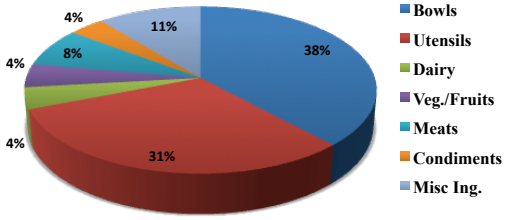

Figure 1: Pie chart of object categories.

Files containing object tracks are stored in Matlab format and placed in *YouCook/Annotations/Objects*. Filenames are in the format of *"VideoID_ObjectName_Num.mat"*, in which *"Num"* is used to identify the tracks of same object names in a video. In each annotation file, there is a matrix that records the corresponding object track in a video. The columns map to the extracted frames of the video (see ***VideoFrames*** in Section 2). The first four rows are used to capture the object bounding boxes and are in the order of *"x_min, y_min, width, height"*, and the fifth binary row marks the object presented in current frame (*1*) or not (*0*).

3.2 Actions

The annotated action types are in seven categories: stir, pick up, put down, season, flip, pour and empty (no desired actions are happening at this time), and each has an ID as shown in Table 2. Action annotations are in *YouCook/Annotations/Actions*, and organized by video names. For each annotation file, it has a matrix that records the time intervals of actions in a video. Columns

Actions	ID
empty	0
stir	1
pick up	2
put down	3
season	4
flip	5
pour	6

Table 2: List of actions.

store actions in a sorted order by time. Rows are *"Action ID, Start Frame, End Frame"*. All frames have an action annotation (and it may be the empty(0) action).

3.3 Human Descriptions

We use Amazon Mechanical Turk to obtain multiple human descriptions for each video. The annotators are shown an example video with a sample description focusing on the actions and objects therein. Participants in MTurk are instructed to watch a cooking video as many times as required to lingually describe the video in *at least* three sentences totaling a *minimum* of 15 words. We set our minimum due to the complex nature of the videos in this dataset.

Annotation files of lingual descriptions are in *YouCook/Annotations/Human_ Descriptions*, and of the format "VideoID_ DescriptionID.txt". The descriptions are provided with the format obtained from the human annotator, without any processing. Table 3 shows the statistics of the human description annotations.

Train			
VideoID	Num of Descrip.	Ave Word	Ave Sent.
0002	1	136	10
0004	7	63	4.6
0005	8	49.3	3.9
0006	9	62.8	4.9
0007	9	51.4	3.7
0014	10	48.2	5.8
0015	10	39.3	3.7
0018	10	39.4	3.4
0019	10	39.3	3.5
0020	10	44.6	3.5
0025	10	48.6	3.8
0026	8	34.9	3.9
0028	9	45.3	4.6
0029	9	63.6	4.4
0030	9	63.6	5.8
0032	10	54.4	4.6
0052	10	57.2	4.6
0054	10	41.3	3.6
0065	7	35.1	4.1
0066	9	46.1	4.1
0067	7	53.1	4.4
0068	6	48.2	4.3
0071	5	47.8	3.6
0072	10	42.9	4.6
0073	8	50.8	3.4
0074	9	43.7	3.8
0075	9	56.6	5
0076	10	49.2	4.3
0077	10	84.8	5.8
0078	8	50.3	4.5
0079	10	42.8	3.7
0080	9	36.3	3.4
0081	10	58.2	4.4
0085	8	46.9	3.9
0086	10	70	5.4
0087	10	48.7	4.2
0088	10	59	4.6
0090	9	38.8	3.4
0091	9	39.4	3.7
0092	10	50.9	3.9
0093	10	46.6	3.8
0094	10	43.4	3.8
0095	10	51.1	4.1
0099	7	50.3	4.7
0100	10	36.5	3.8
0101	3	35.7	3.7
0102	10	33.2	3.2
0103	6	35.7	4
0104	8	44.9	4.5

Test			
VideoID	Num of Descrip.	Ave Word	Ave Sent.
0003	8	75.6	5.8
0008	10	75.2	5.5
0009	7	129.1	7.9
0010	7	87.3	9
0011	10	43.8	3.3
0012	8	54.9	5
0013	8	48.9	4.3
0016	10	76.5	5.9
0017	9	75	5.8
0021	8	66	5
0022	4	63.5	5.8
0023	7	45.9	4.4
0024	9	56.1	4.6
0027	5	62.8	5.2
0033	9	69.3	4.9
0034	8	89	6.6
0035	7	80.9	8.4
0036	10	65.4	4.7
0037	8	51.4	4.1
0048	10	47.4	4.9
0050	5	78.6	7.8
0051	9	73.3	6.8
0053	7	61.9	5.3
0055	10	31.6	3.6
0056	10	60	5.4
0057	10	62	4.6
0058	6	93.7	6.2
0059	5	103.2	7.4
0060	10	56.6	4.3
0061	7	110.9	8
0062	6	56.7	4.3
0063	9	80.9	7
0064	9	28.9	3.2
0069	8	43	4
0082	10	50.8	4.9
0083	10	68.2	4.6
0084	10	52.1	4.3
0097	10	63.7	6.5
0098	9	39.4	3.6

Table 3: Statistics of human descriptions. The left part is the training set and the right part is the test set. We record the number of lingual descriptions per-video (Num of Descrip.), average word count per-description per-video (Ave Word) and average sentences per-description per-video (Ave Sent.).

4 Precomputed Features

4.1 HOG3D Features

The low level HOG3D features for our YouCook videos are obtained using the open source software provided by Kläser et al. [Kläser et al., 2008][1]. The codebook (1000 Words) is computed in a standard way using the K-Means algorithm[2]. We outline the main steps in Algo. 1:

Algorithm 1 Codebook computation: a general outline

1: Compute HOG3D features from frames sampled from the videos in the training set. Listings 1 and 2 show the Matlab code for extracting the feature vectors corresponding to a sample training video using the method described in [Kläser et al., 2008].
2: Compute K-Means over the HOG3D features of the training dataset (see Listing 3). A maximum of 10,000 feature vectors (each a 300-dimensional descriptor) for each video is obtained and concatenated for all videos in the training set to form the initial dataset for K-Means clustering. After this, clustering is performed for a maximum of 100 iterations and the codebook vectors are saved.
3: A K-dimensional histogram of HOG3D features are computed for each video (training or test) using the codebook obtained in Step 2 and its HOG3D feature descriptor (see Listing 4).

After the first step in Algo. 1, we obtain large feature files for each video where each line in output of the HOG3D feature extractor corresponds to a frame of the video and consists of a 308-dimensional vector in the following format:

```
<x> <y> <frame> <x-norm.> <y-norm.> <t-norm.> <xy-scale> <t-scale>
<descriptor>
```

The "descriptor" is the actual 300-dimensional action descriptor that is ultimately used in codebook computation. The stride for the spatial scale is set to nine and that for the temporal scale is set to five. We spatially rescale the video such that the larger of its width and height is 160 pixels.

There is also a [xy/t]-max-scale option that controls at which spatial/temporal scale the sampling is stopped. We set this option's value to one to reduce computational time.

4.2 Color Histogram Features

The low level color histogram features are computed in an efficient way from the frames obtained by using the ffmpeg command:

```
ffmpeg -i <movie-file> -y <output-directory>/frame%05d.jpg
```

Listing 5 in appendix shows the Java code for extracting color histogram for an image using an user specified number of bins for each of the red, green and blue components. We set the number of such bins for each component to be eight for a total of 512-dimensional color histogram for each frame of the video. Histograms from all such frames are concatenated to form the color histogram for the entire video. The text file containing concatenated features for a video are stored on disk in g-zipped format. For use in topic models, we use the average of the values of the color combinations (i.e. the bins) from all of these histograms as the color histogram descriptor of a single video.

When we use the color histogram features in the topic model, we only use the bins that contain values within the 15^{th} to 85^{th} percentile of the values in the bins. The higher values need to be removed since they often increase the magnitude of the slopes around the fixed points in the fixed point iterations of the local (video document specific) E-step in variational Bayes optimization framework to be more than one and thus leading to non-convergence [Conte and Boor, 1980] and possible degeneration of the proportions of a few topic components. The lower values are removed to offset the bias in removing the higher band. The rest of the bin values are normalized to lie in $[0, 200]$.

[1] http://lear.inrialpes.fr/people/klaeser/software_3d_video_descriptor
[2] http://crcv.ucf.edu/source/K-Means

5 Evaluating the Natural Language Descriptions

The primary purpose of the YouCook dataset is to serve as a challenging benchmark for the natural language video description problem.

We use the ROUGE [Lin and Hovy, 2003] tool to evaluate the generated lingual descriptions for all the test videos to the corresponding ground truth human descriptions that we obtained through M-Turk. The tool is included for convenience or can be downloaded from http://www.berouge.com/.

ROUGE is an n-gram recall oriented tool which measures the coverage of information need between summaries manually written by annotators and that automatically generated by systems. ROUGE-1 (unigram) recall is the best option to use for comparing descriptions based only on predicted keywords. ROUGE-2 (bigram) and ROUGE-SU4 (skip-4 bigram) is best suited to evaluate summaries w.r.t. coherence and fluency.

We now show an example of how to run the ROUGE scorer on a sample of lingual descriptions obtained from the low level multi-modal topic model only.

The ROUGE command is run from the *YouCook/Evaluation* directory as:

```
perl ./ROUGE-1.5.5/ROUGE-1.5.5.pl -e ./ROUGE-1.5.5/data
-n 2 -x -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -a -d  -l 67
./config.txt > ./rouge.out
```

The command line options specified here ("*-n 2 -x -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -a -d*") consist of the standard set of parameters used in text summarization conferences[3]. We add the extra parameter "-l 67" to indicate a cut-off length of 67 words for system summaries prior to evaluation. The summaries generated from the low level of our system are stored in the directory *YouCook/Evaluation/summaries/cvpr2013*. Each file, for e.g. *YouCook/Evaluation/summaries/cvpr2013/0003-summary.txt*, consists of a single lingual description of a test video in a sentence-per-line (SPL) format (see file "*config.txt*" for details). Further, the lingual descriptions are generated using predicted keywords from the MMLDA model using an asymmetric topic proportion prior [Das et al., 2013] with twenty topics and using discrete visual features—HOG3D and color histogram (chist).

The human description files are similarly stored in the directory *YouCook/Evaluation/summaries/human_descriptions*, where there can be multiple descriptions corresponding to a single test video.

6 Final Remarks

We thank the anonymous annotators for their contribution to the YouCook dataset. If you run into any problems, have any suggestions, or make any improvements, please contact us via email.

6.1 Reproducibility

Note that the extracted frames depend on the extraction tool and version. We use *mmreader* in Matlab to extract ***VideoFrames*** and the following *ffmpeg* to extract frames for precomputed low level features.

FFmpeg version 0.6.6-4:0.6.6-0ubuntu0.11.04.1, Copyright (c) 2000-2010 the Libav developers built on Jun 12 2012 16:35:16 with gcc 4.5.2

[3] http://www-nlpir.nist.gov/projects/duc/duc2005/

References

[Conte and Boor, 1980] Conte, S. D. and Boor, C. W. D. (1980). *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw-Hill Higher Education, 3rd edition. 5

[Das et al., 2013] Das, P., Xu, C., Doell, R., and Corso, J. J. (2013). A thousand frames in just a few words: lingual description of videos through latent topics and sparse object stitching. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2, 6

[Kläser et al., 2008] Kläser, A., Marszalek, M., and Schmid, C. (2008). A spatio-temporal descriptor based on 3d-gradients. In *BMVC*. 5, 8

[Lin and Hovy, 2003] Lin, C.-Y. and Hovy, E. (2003). Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACL HLT*. 2, 6

A Code Snippets

Listing 1: Driver for obtaining HOG3D histogram

```
cookingVidPaths = ['Videos/0002.mp4']; % add more here
tmpPath='ffprobe_tmp.txt';
featureDir='Features/HOG3D/feature_vectors';

c = cellstr(cookingVidPaths);
nVids = size(c);
mkdir(featureDir);
for i = 1: nVids
   videoPath = c{i};
   HOG3DExtractor(videoPath, tmpPath, featureDir);
end
```

Listing 2: Extracting HOG3D features for a video using the static executable in [Kläser et al., 2008]

```
function xyscale = HOG3DExtractor(videoPath, tmpPath, featureDir)
% videoPath: path to the input video
% tmpPath: output of "ffprob -show_streams videoPath > tmpPath".
% featureDir: Directory that you want to save feature.

% extract width and height
[~, name, ~] = fileparts(videoPath);
fprintf('Processing video %s.... \n', videoPath);
system(sprintf('ffprobe -show_streams %s > %s', videoPath, tmpPath));
[~, grep_width] = system(sprintf('grep "width=*" %s', tmpPath));
[~, grep_height] = system(sprintf('grep "height=*" %s', tmpPath));
width = sscanf(grep_width, 'width=%d');
height = sscanf(grep_height, 'height=%d');
max_w_h = max(width, height);
xyscale = -1;
if(any (size(max_w_h) == 0))
   fprintf('!!! error in ffprobe of %s. !!!\n', name);
   return;
end
xy_srtride = 9;
t_stride = 5;

% compute rescaling ratio
xyscale = 160 / max_w_h;
featurePath = sprintf('%s/%s.txt', featureDir, name);
% need to specify the path to HOG3D code
system(sprintf('path/to/HOG3D_code/extractFeatures_static --simg %f --xy-
    stride %d --xy-max-scale 1 --t-stride %d --t-max-scale 1 %s > %s',
    xyscale, xy_srtride, t_stride, videoPath, featurePath));
featureInfo = dir(featurePath);
if( ~exist(featurePath, 'file') || featureInfo.bytes == 0)
   fprintf('!!! error in feature extraction of file %s. !!!\n', name);
   return;
end
fprintf('video %s was processed successfully.\n\n', videoPath);
ffprobe_file = tmpPath
delete(ffprobe_file);
```

Listing 3: Computing HOG3D codebook vectors using K-Means

```
cookingHOG3DPaths = ['Features/HOG3D/feature_vectors/0002.txt']; % add
    more here

c = cellstr(cookingHOG3DPaths);
nVids = size(c);
```

```
D = [];
nFrames = 10000;
for i = 1: nVids
   HOG3DPath = c{i};
   fprintf('loading file %s\n',HOG3DPath);
   A = load(HOG3DPath);
   P = A(:,9:308)';
   [row, col] = size(P);
   if ( col > nFrames )
      interval = floor(col/nFrames);
   else
      interval = 1;
   end
   index = [0:interval:col-1]+1;
   D = [D P(:,index)];
end

nClusters = 1000;
[CX, sse] = vgg_kmeans(D, nClusters, 'maxiters', 100);

dlmwrite('Features/HOG3D/codebook_1000.txt', CX');
```

Listing 4: Quantizing a video using a pre-computed HOG3D K-means codebook

```
function xyscale = quantizeVideo(videoPath, tmpPath, featureDir,
    codebookFile, wordDir, histDir)
% videoPath: path to the input video
% tmpPath: output of "ffprob -show_streams videoPath > tmpPath".
% featureDir: Directory that you want to save feature.
% codebookPath: address of the coreword
% wordDir: directory to save the quantized feature
% histDir: directort to save the global histogram of HOG3D

addpath('path/to/VGG_KMeans'); % path to K-means code

scanFormat = '%d %d %d %f %f %f %d %d ';
for i = 1 : 300
  if i ~= 300
    scanFormat = [scanFormat, '%f '];
  else
    scanFormat = [scanFormat, '%f'];
  end
end

% read codeword dictionary
clusterCen = load(codebookFile);

% extract width and height
[pathstr, name, ext] = fileparts(videoPath);

fprintf('Processing video %s.... \n', videoPath);
system(sprintf('ffprobe -show_streams %s > %s', videoPath, tmpPath));

[status grep_width] = system(sprintf('grep "width=*" %s', tmpPath));
[status grep_height] = system(sprintf('grep "height=*" %s', tmpPath));

width = sscanf(grep_width, 'width=%d');
height = sscanf(grep_height, 'height=%d');

max_w_h = max(width, height);

xyscale = -1;
if(any (size(max_w_h) == 0))
```

```
   fprintf('!!! error in ffprobe of %s. !!!\n', name);
   return;
end

xy_srtride = 9;
t_stride = 5;
% compute rescaling ratio
xyscale = 160 / max_w_h;


featureFile = sprintf('%s/%s.txt', featureDir, name);
if exist(featureFile, 'file')
   system(sprintf('rm -f %s', featureFile));
end
% need to specify path to HOG3D code
system(sprintf('path/to/HOG3D_code/extractFeatures_static --simg %f --xy-
    stride %d --xy-max-scale 1 --t-stride %d --t-max-scale 1 %s > %s',
    xyscale, xy_srtride, t_stride, videoPath, featureFile));

ftrInfo = dir(featureFile);
if( ~exist(featureFile, 'file') || ftrInfo.bytes == 0)
   fprintf('!!! error in feature extraction of file %s. !!!\n', name);
   return;
end

% read features, save their indeces i.e. Quantize the extracted features
    using the pre-computed words
fid = fopen(featureFile);
A = fscanf(fid, scanFormat, [308 100000]);
word2 = cell(100,1);
i = 1;
while ~any(size(A)==0)
   [ind d2] = vgg_nearest_neighbour(A(9:end,:), clusterCen');
   word2{i} = [A(1:3, :)' ind];
   i = i+1;
   A = fscanf(fid, scanFormat, [308 100000]);
end
words = cat(1, word2{:});
fclose(fid);

system(sprintf('rm -f %s', featureFile));

fprintf('saving the words.\n');
% compute histogram from the quantized features
hist = zeros(size(clusterCen, 1), 1);
for i=1:size(words,1)
   hist(words(i,4)) = hist(words(i,4)) + 1;
end
save_word_name = sprintf('%s/%s.txt', wordDir, name);
dlmwrite(save_word_name, words);

% This is what we are interested in ultimately
save_hist_name = sprintf('%s/%s.txt', histDir, name);
dlmwrite(save_hist_name, hist);

fprintf('video %s was processed successfully.\n\n', videoPath);
```

Listing 5: Java code for computing color histogram from an image

```
public static int[] binnedImageHistogram(BufferedImage input, int nBins)
{
  int[] hist = new int[nBins*nBins*nBins];

  int interval = 256/nBins;
```

```
  int nBinsSquared = nBins*nBins;

  int W = input.getWidth(); int H = input.getHeight();
  for(int i = 0; i < W; ++i) {
    for(int j = 0; j < H; ++j) {
      int red = new Color(input.getRGB (i, j)).getRed();  // 0 to 255
      int green = new Color(input.getRGB (i, j)).getGreen(); // 0 to 255
      int blue = new Color(input.getRGB (i, j)).getBlue(); // 0 to 255

      int rBin = red/interval;
      int bBin = blue/interval;
      int gBin = green/interval;
      int coord = rBin*nBinsSquared + gBin*nBins + bBin;

      hist[coord]++;
    }
  }
  return hist;
}
```

The color histogram code is run from the "*code*" directory as follows:

```
java -cp *.jar edu.buffalo.cse.VideoProcessing.ColorHistogramFromVideo
../Videos ../ColorHist
```