# LIBSVX Tutorial

Chenliang Xu and Jason J. Corso
SUNY at Buffalo
{chenlian,jcorso}@buffalo.edu

27 July 2012 (corresponds to libsvx v 2.0)

This tutorial will guide you through the use of the various elements of libsvx: both the supervoxel segmentation methods and the benchmark evaluation. To make this easy, we have included a short subset of a video sequence on which we'll work, which is taken from our labeled benchmark subset of `xiph.org`. Assume the frames of the input video *bus* are extracted into two formats: ppm and png, which are placed at *path/to/libsvx/example/frames_png/* and *path/to/libsvx/example/frames_ppm/*. The video *bus* has 30 frames with the resolution 240x160.

We hope this tutorial is able to make it easy for you to learn how to use the library we've provided. Suggestions and code improvements are quite welcome; email us!

The library project website is `http://www.cse.buffalo.edu/~jcorso/r/supervoxels/`, which includes a link to this tutorial in www format that includes video results to help guide you through the tutorial.

# 1 Installation

Please follow the README file in the directories of each methods and benchmark code.

Put simply, there is code in *path/to/libsvx/gbh_stream/*, *path/to/libsvx/gbh/* and *path/to/libsvx/swa/* that needs to be compiled (via provided makefiles). The code in *path/to/libsvx/nystrom_ncut/* and *path/to/libsvx/svxbench/* are in Matlab with no mex'd code.

# 2 Example Usage

We include the sample usages of the library methods here. Assume the frames of the input video *bus* are extracted into two formats: ppm and png, which are placed at *path/to/libsvx/example/frames_png/* and *path/to/libsvx/example/frames_ppm/*. The video *bus* has 30 frames with the resolution 240x160.

## 2.1 Streaming Algorithm

The streaming algorithm requires only constant memory (depends on the streaming window range) to execute the algorithm which makes it feasible for surveillance or to run over a long video on a less powerful machine. It approximates the solution of an offline algorithm.

### 2.1.1 Graph-based Streaming Hierarchical Video Segmentation (StreamGBH)

Assume you are in *path/to/libsvx/gbh_stream* and that you have successfully compiled the gbh_stream executable.

To perform the segmentation, run the following command, which will output segmented frames into *path/to/libsvx/example/output_gbh_stream/*:

```
./gbh_stream 5 200 100 0.5 10 20 ../example/frames_ppm ../example/output_gbh_stream
```

This command should take about a few minutes to run.

The parameters of the gbh_stream executable are fully explained in the README file in *path/to/libsvx/gbh_stream*. In short, the 5 200 and 100 are the merging thresholds (roughly put, larger thresholds means larger supervoxels). The 0.5 is the Gaussian smoothing parameter.

The 10 is the number of frames to include in one video subsequence/streaming window rang, and the 20 is the desired levels in the hierarchy. The algorithm implements a superset of StreamGBH, StreamGB, GBH and GB, corresponding to different combinations of these two parameters.

The **output** of the folder is organized with each hierarchy level having its own newly created subdirectory. In our case, there will be new directories `00` through `20`. In each of these directories is the supervoxel output as images with frames number similar to the input frames. Separate supervoxels are colored with unique RGB values.

## 2.2 Offline Algorithm

Offline algorithms require the video to be available in advance and short enough to fit in memory. It loads the whole video at once and processes afterwards. Under most circumstances, it gives better segmentation results than the corresponding streaming algorithm since it is aware of the whole video.

### 2.2.1 Graph-based Hierarchical Segmentation (GBH)

Although you can run GBH with a parameter setting in StreamGBH, here we still keep a separate version of GBH, which inherits from libsvx 1.0. This implementation features with a balanced searching tree that makes it slightly faster when the video is short enough to load into memory at once.

Assume you are in *path/to/libsvx/gbh/* and that you have successfully compiled the gbh executable..

To perform the segmentation, run the following command, which will output segmented frames into *path/to/libsvx/example/output_gbh/*:

```
./gbh 5 200 100 0.5 20 ../example/frames_ppm ../example/output_gbh
```

This command should take about a few minutes to run.

The parameters of the gbh executable are fully explained in the README file in *path/to/libsvx/gbh*. They are the same parameters as in StreamGBH except that it lacks a streaming window range.

### 2.2.2 Graph-based Segmentation (GB)

Here, we treat the GB as one special case of GBH, where the hierarchy level is equal to zero.

Again, assume you are in *path/to/libsvx/gbh*. Run the following command, which will output the segmented frames into *path/to/libsvx/example/output_gb/*:

```
./gbh 100 0 100 0.5 0 ../example/frames_ppm ../example/output_gb
```

This code will run much faster.

The second parameter is disregarded, as it is only for the hierarchical version.

#### 2.2.3 Nyström Normalized Cuts (Nyström)

Open MATLAB (we use R2011b). Assume you are in *path/to/libsvx/nystrom_ncut*.

Run the following command in MATLAB command shell, which will output results into *path/to/libsvx/example/output_nys*. Note, this command, requires the Optimization Toolbox and will automatically open a matlabpool.

```
Nystrom_video('../example/frames_ppm','../example/output_nys',50,200,50,20,20,0)
```

Note: The algorithm requires large memory and takes long time to compute. Please watch your system monitor.

The parameters to this function are explained in the script itself. Briefly, they specify, the input and output paths, the desired number of supervoxels (the higher this number, the longer it will take to compute), number of Nyström sample points (the higher the number the more accurate the approximation is and yet the more memory that is required), the number of eigenvectors to compute for the embedding, the next two are the importance of the Euclidean distance and the color space, and the last one specifies whether or not to use KNN for the output (0 is only use kmeans and 1 is use 10% kmeans and then do KNN, which will generate the output faster but is an approximation to the full kmeans).

#### 2.2.4 Segmentation by Weighted Aggregation (SWA)

Assume you are in *path/to/libsvx/swa* and you have compiled the swa binary.

First of all, please note the config file *path/to/libsvx/example/swa_config/config_example.txt* that has been created for this tutorial. Its contents are

```
InputSequence=../example/frames_png/%05d.png
Frames=30-30
NumOfLayers=12
MaxStCubeSize=100
VizLayer=5-12
VizFileName=../example/output_swa
```

Full description of these parameters are provided in the README and the swa.cpp source file. In short, they indicate what to run only, no parameters.

Then run the following command, which will create the supervoxel output in *path/to/libsvx/example/output_swa*.

```
./swa ../example/swa_config/config_example.txt
```

The output is stored in a similar manner as the other hierarchical methods above.

## 3 Supervoxel Benchmark

The supervoxel benchmark is a separate part the supervoxel library that is able to compute quantitative scoring metrics against human-drown segmentations. Our CVPR 2012 paper thoroughly describes the metrics that are included in the benchmark. Here, we show you how to run the previously computed results from the tutorial (above) through the benchmark and generate scores.

To run the benchmark, you must compute the results for a complete data set, say the Chen xiph.org data set, which is included with the benchmark download. And, these results must be computed with a varying set of parameters, so that we can generate the output curves (corresponding to different supervoxel numbers per video).

The next paragraph describes how to use a provided shell script to generate a full set of results. Alternatively, you can download them from our website (if you just want to test the benchmark code) at `http://www.cse.buffalo.edu/~jcorso/extdelivery/libsvx_example_full.tar.bz`, and skip the next paragraph. Be sure to place them in *path/to/libsvx/example/chen_swa*.

Here we show an example of the SWA method. We have provided a script in *path/to/libsvx/example/compute_chen_swa.bash*, which you should execute (it is a bash-shell executable script) from inside of the *path/to/libsvx/example* directory. This will take about 90 minutes and 10GB memory to run—it computes the SWA segmentation over all the videos. The program will create a directory in *path/to/libsvx/example/chen_swa*, which contains all 8 videos in Chen's data set with the video name as the name of the directory. In each video directory, you will see subdirectories 07 through 12, which contain the segmentation results of level 07 trough level 12 respectively.

Once you have the segmentation results ready, you can run the benchmark code to generate the comparative figures. Following the above example, you need to set

```
path_input_method = '../example/chen_swa';
path_ppm = 'dataset/Chen_ppm';
dataset = 1;
output_path = '../example/chen_swa_benchmark';
```

in *path/to/libsvx/svxbench/EVALUATION.m*. Then run the script inside the Matlab command shell. It will take an hour or so to run (depends on the number of supervoxels); there are a lot of methods in the evaluation. It will save all of the computed figures and metrics into the output_path that is specified in the script.

We use the GBH method as another example. The script is in *path/to/libsvx/example/compute_chen_gbh.bash*. This will take about 60 minutes and 4GB memory to run—it computes the GBH segmentation over all the videos. Note different parameter settings require different time to compute, and will generate different results. To make it run faster, we set the following

```
./gbh 5 500 200 0.5 20 /path/to/input /path/to/output
```

in our script for all videos. The program will create a directory in *path/to/libsvx/example/chen_gbh*, which contains all 8 videos in Chen's data set with the video name as the name of the directory. In each video directory, you will see subdirectories 00 through 20, which contain the segmentation results of level 00 (oversegmentation) through level 20 respectively.

Once you have the segmentation results ready, you can again run the benchmark code to generate the comparative figures. You need to set

```
path_input_method = '../example/chen_gbh';
path_ppm = 'dataset/Chen_ppm';
dataset = 1;
output_path = '../example/chen_gbh_benchmark';
```

in *path/to/libsvx/svxbench/EVALUATION.m*. Then run the script.

# 4 Final Remarks

We hope you have found this tutorial to be a gentle introduction to the library and benchmark. We, of course, also hope you are able to make good use of the code. If you run into any problems, have any suggestions, or make any improvements, please contact us via email. We will periodically release updates.

The segmentation methods provided herein are implemented to the best of our abilities to match the original works. Many of the methods are parameter dependent and require large amounts of memory and compute time to run. We are working on a novel streaming method that will be able to get around these hurdles, or at least some of them, and will release it when complete.