Convolutional Neural Network Implementation of Superresolution Video

David Zeng
Stanford University
Stanford, CA
dyzeng@stanford.edu

Abstract

This project implements "Enhancing and Experiencing Spacetime Resolution with Videos and Stills" by Gupta et al. [4] with convolutional neural networks. The input is a series of video frames. The first and last frame are high resolution while the middle frames are low resolution. Optical flow is computed between both high resolution frames to each low resolution frame by FlowNet [5]. The high resolution images are warped with this optical flow. A superresolution image is created by a Graphcut composition from the two warped high resolution images and the upsampled low resolution image. Multiple units of this computation are concatenated to form video.

1. Introduction

This project implements "Enhancing and Experiencing Spacetime Resolution with Videos and Stills" [6] with a convolutional neural network (CNN) framework. The goal of the original paper is to produce superresolution video frames by inferring high resolution information from periodically acquired high resolution still frames. The authors originally envisioned this technology implemented in hybrid camcorders which would have two image sensors—one for acquiring low resolution video and one for acquiring high resolution stills.

While such technologies did not gain traction in the consumer market, the mobile computing platform has developed to be quite capable. With the continuing advance of consumer video resolutions from 2K to 4K, demand for computational power, bandwidth, and storage requirements will continue to increase as well. This could allow for power-efficient real-time processing of high resolution video.

From an even more constrained embedded system perspective, this allows for relaxed hardware requirements. Rather than having a hardware pipeline to process full resolution video in real-time, fewer pixels have to be processed in real-time and less data has to be stored. There is then a tradeoff that more software processing will have to be performed later to playback the video. This processing could be performed on an accompanying mobile device or offloaded to the cloud.

This project aims to implement [6] by using a CNN framework. The goal is to create believable superresolution video from low resolution frames while striving for real-time computation.

2. Background

There are two primary technical areas in this project: optical flow and image composition.

2.1. Optical Flow

Optical flow takes in two images and assigns a flow vector for each pixel from one image to the other. With the flow image, we can warp one image to the other to approximate the other image.

There are several methods to compute optical flow. One of the simplest methods is to search the neighborhood around the current pixel and find the most similar pixel for a given distance metric. There exist heuristics to determine neighborhood size and traversal method.

Expanding on this idea are block-based methods since pixels usually move similarly to their neighbors and are likely to be part of the same object. The Lucas-Kanade method is based upon this idea and solves for motion in a least squares sense [9]. Extending this idea even further, there are optical flow methods that first segment the two images and then calculate the geometric distortion to arrive at the final optical flow vector [14].

There are also models based upon other heuristics which can be generalized as assuming some property of the image is slowly varying while the flow field is smooth. A seminal paper in this trend is the Horn-Schunck method [7].

With these methods, there have also been techniques that enhance these methods [12]. Multi-scale coarse-to-fine image pyramids are applicable to optical flow. High-order filter constancy provides robustness to lighting changes. Median filters remove outliers and smooths the flow field.

Traditionally, these objectives were formulated as a matrix problem and solved by least-squares. With the rise in popularity of non-convex objectives and increasingly powerful convex solvers, many modern methods now solve an objective to simultaneously determine the flow field.

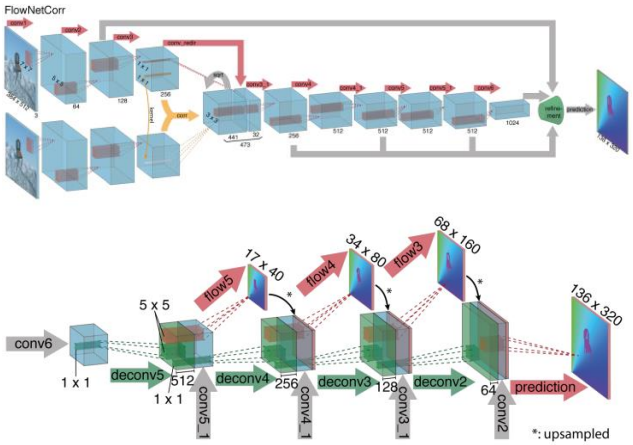


Figure 1. FlowNet network architecture

Convolutional neural networks are resurging in the image processing field and this course is based upon it. FlowNet (Figure 1) is currently the only CNN method for optical flow [5]. Rather than make assumptions and develop quantifiable metrics for these assumptions, CNNs are data-driven models. With enough training data, we can allow our CNN to learn how to compute optical flow.

Briefly, FlowNet first processes each image individually in parallel feature extraction networks and then combines the results in a correlation network before computing insertion into a series of convolutional layers. Coarser layers from earlier in the neural network are input and the convolutional layer output is the input to a series of fractionally strided convolution layers to generate larger images.

2.2. Image Compositing

Image composition is the blending of several images with each other. The simplest method is alpha-blending. Alpha blending is a linear transition between two images with slope alpha [11]. The blending regions can vary in geometry, including planes, rectangles, and ellipses.

Extending this idea, we can perform multi-scale Laplacian pyramid blending [13]. We compute Laplacian and Gaussian pyramids of each image and then linearly combine them at each scale. We collapse the pyramid and arrive at our final image. Again, the blending region can be adjusted.

Another method is two-band blending [2]. Low frequencies are smoothly blended with alpha blending while high frequencies are concatenated. Extending this idea is gradient blending where we adjust the image we want to blend until the gradient is smooth enough [10].

In this project, we are using Graphcut [8]. Graphcut segments out a piece of the intended image by minimizing a cost metric. The cost metric is composed of a smoothness cost so that the seams are not noticeable and a data cost so that the pixels being replaced are similar to the rest of the image.

3. Methods

The input for this project is a set of low resolution video frames and two high resolution images which bookend the video frames. For initial development, the Middlebury dataset is used. After tuning, the system is validated on actual video footage.

The Middlebury dataset is all full-resolution images and provides eight frames for each video. Thus, we define a unit of computation as a set of eight images. To synthesize an appropriate dataset, the middle six images are downsampled by a given factor with bilinear interpolation and then upsampled by the same factor with bilinear interpolation. The upsampling is necessary to create images of identical size for input to the optical flow CNN. Various downsampling factors are tested to explore the capabilities of the algorithm.

4K and 2K video are also used. We also downsample and upsample the middle frames to simulate low resolution video acquisition.

There are two primary steps for implementing this project. The first is computing optical flow from both high resolution images to each low resolution frame and synthesizing the warped images. The second step is compositing the warped images to form the final high resolution image.

3.1. Optical Flow

We use a pretrained FlowNet to compute optical flow [5]. Optical flow is a pairwise operation and we can exploit the temporal of the relations to improve optical flow results. Consider a sequence of images A-B-C-…-H where A and H are high resolution and B-I are low resolution. Denote the upsampled low resolution images as B_u-I_u. Let us initially consider optical flow for only one direction. We first compute optical flow for $A \rightarrow B_u$. Let the warped image of A be denoted as B_t. We then compute optical flow for $B_t \rightarrow C_u$. Let the warped imaged of B_t be denoted as C_t. We continue this until we have completed all of the pairwise optical flows. By using the warped image to compute optical flow, we are effectively accumulating the flow vectors from A to each image.

We now use the intermediate results as a seed to refine the long range optical flow from A to each target image. We compute $B_t \rightarrow B_u$, $C_t \rightarrow C_u$, and so forth and arrive at final forward warped images of B_f, C_f, and so forth.

We repeat this process for warped images of H to each low resolution image. Let us call these backward warped images B_b, C_b, and so forth. This results in a set of three images: the upsampled image, the forward warped image, and the backward warped image.

Figure 2. The left is composition by traditional Graphcut. The right is using the data cost only.

3.2. Image Compositing

With the three candidate images, we now use Graphcut [8] to composite the images. By optimizing the smoothness cost and data cost, we can create a composite image.

Traditional Graphcut methods use an iterative approach, minimizing data cost and smoothness cost alternately to minimize the ultimate cost function. The upsampled image and two warped images are all trying to approximate the same image. Thus, we can assume smoothness *a priori*. For verification, Figure 2 displays the results of using Graphcut and only the data cost. The output compositions are nearly indistinguishable. We can thus optimize the cost on a pixel-by-pixel basis in a single pass. The data cost at a particular 3-channel RGB pixel follows.

$$c_{upsample} = k_u$$
$$c_{forward} = \left\| I_f(x,y) - I_u(x,y) \right\|_2$$
$$c_{backward} = \left\| I_b(x,y) - I_u(x,y) \right\|_2$$

In this implementation, the upsampled image is viewed as the base image and has self-cost k_u. The self-cost is a tunable parameter. A small k_u results in a lower resolution image but with fewer artifacts. A larger k_u results in higher detail but can introduce greater artifacts. For this project, we choose k_u=0.15.

4. Results and Discussion

For developing and tuning the system, the Middlebury and Sintel test datasets are used [1,3]. The low resolution images are downsampled by a factor of 2 and 4 in each dimension. The results are displayed in Figure 3.

Figure 3a) is from the image set "Backyard" in the Middlebury dataset and downsampled by a factor of 2. We can see slight blurring in the image on the left. After superresolution, we see significant resolution improvement in the background plants, the facial features, and on the piece of furniture that the girl with the red skirt is standing on.

Figure 3b) is downsampled by a factor of 4. In this case, the blurring is significant. The superresolution image recovers many of the details but much of the information has been lost as well.

Comparing the result in b) with the result in a), we notice much more detail in a). Even comparing the result in b) with the downsampled image in a), we notice that the result in b) is worse.

This shows that downsampling by a factor of 4 negatively affects the image more significantly than downsampling by a factor of 2. Analysis from a Fourier approach suggests that the filter when downsampling by a factor of 2 does not remove much information since energy is sparse in high frequencies. When downsampling by a factor of 4, denser energy frequencies are lost.

Figure 3c) is from the image set "temple_1" in the Sintel dataset and downsampled by a factor of 2. We can see the improvements most obviously at the textures around the joints in the bamboo.

Figure 3d) is downsampled by a factor of 4. Similarly, the blurring is very noticeable but we do recover lots of sharpness in the image, again noticeable on the bamboo joints.

While the Sintel dataset is a popular dataset for optical flow development, it is not useful in this application based upon its results. The Sintel dataset is entirely computer generated and thus does not have the same level of detail that a natural image would have. During downsampling, little information is actually being discarded and thus there is little information to recover. This is apparent even in the images below. This is disappointing because the Sintel dataset is commonly used in other papers and has video sequences.

Figure 4 contains the L1 and L2 errors of each channel of the superresolution frames compared to the original full resolution frame. Frames 1 and 8 have zero error because they are drawn directly from the source. The error for the frames in the middle plateaus. This plateau is interesting and unexpected. Intuitively, the error should have a peak in the middle. Frames 2 and 7 would be expected to have low error because they are very similar to the high resolution frames. Frames 4 and 5 are temporally distant from the high resolution frames and it is expected that the optical flow has accumulated error as well as lost information.

Computing these errors on other image sets, using both L1 and L2 error do not give additional insight so only L1 error is used for faster computation.

Based upon these results, downsampling by a factor of 2 still has an advantage of almost one fourth the necessary pixels while maintaining enough recoverable detail. This technique is now applied to 4K and 2K video. The results are shown in Figure 5.

Viewing the video associated with Figure 5a), it is notable that the method can handle the fast action of the kick. Analyzing the error graph for the video does not reveal any correlations between the motion and the error.

Qualitatively viewing the video associated with Figure 5b), the video still looks high resolution. Paying closer attention to the video, there are some artifacts from this

method. Some out of focus areas of the video with fine details are temporally oscillating in their quality such as the trees in the background. This comes from the high detail in the high resolution frames and less detail in the superresolution frames.

Analyzing the error graph for the video, there is again no correlation between error and the noticeable artifacts in the video. Furthermore, there is no correlation between objects entering and exiting the scene.

It is difficult to create a metric for superresolution applications because almost all published image quality metrics are correlated with image energy. In this case, we have preserved most of the image's energy as it is dense in low frequencies. We are trying to recover the high frequencies of each image where energy is sparse and small.

This raises the idea of only comparing high frequencies with each other. However, this is also problematic because high frequencies are dominated by noise and signal-to-noise ratio is small. Comparing high frequencies would ultimately be comparing noise floors between images which is not the desired result.

The Middlebury dataset has RGB images with 640x480 resolution. Calculating the superresolution for one unit of computation requires about 32 seconds. For eight frames, this is an average 4 seconds per frame which is not fast enough for real-time.

The Sintel dataset has RGB images with 1024x436 resolution. The superresolution for one unit of computation requires about 34 seconds.

4K video has a resolution of 3840x2160 and requires about 90 seconds for computation. 2K video has a resolution of 1920x1080 and requires about 50 seconds of computation.

We have not achieved even close to real-time computational speeds with this method. There is still hope though. The computational time does not scale linearly with pixels as would be expected if computation alone were the bottleneck. Almost the entirety of the computation is optical flow since we reduced the complexity of Graphcut. This suggests that most of the computation is spent transferring the data to and from the GPU.

More efficient hardware stacks can be implemented to reduce this cost in mobile and embedded systems. For embedded systems especially, there exist dedicated convolution chips which could increase performance.

All of the videos in the results are included in the supplemental material. For space considerations, only the 2K version of the videos have been included.

4.1. Super-resolution CNN

As another point of comparison, there exists a super-resolution CNN from Dong et al [4]. Unfortunately, the CNN was only created for black-and-white images. Trying to add back color is not quite successful but does help for comparison. Since the image is black-and-white, the above metrics cannot be used and it is for qualitative purposes only. The videos are also included in the supplementary materials.

The superresolution images retain sharpness but are obviously lacking details that the above method recovers. When comparing computation time with the above method, the super-resolution CNN is significantly faster. The architecture is CONV-RELU-CONV-RELU-CONV. Both Middlebury and Sintel datasets takes 12 seconds. 2K video requires 20 seconds and 4K video requires 30 seconds of computation. This data again suggests that most of the time is spent on data transit.

With better hardware optimization, this CNN could lead to a camera that captures a slightly smaller amount of pixels that then calculates a superresolution image at the promised quality. The benefits of this would be less initial processing while capturing in real-time and larger pixel sensors for better low-light performance.

5. Future Work

Almost the entirety of the computation is calculating optical flows. Optical flow is an expensive task and may limit the efficiency of this method.

Future work could involve training a shallower network with results that are still acceptable. Graphcut adds some robustness to the method by filtering out significant artifacts.

Another idea is to replace Graphcut with a CNN as well. The current implementation performs a single computation on each pixel so it is unlikely that it can be faster. However, there may be higher quality image composition methods that could compensate for lower quality optical flow warped images.

A better metric to evaluate high frequency details also needs to be developed for better quantitative comparison of high frequency and detailed differences images.

References

[1] S. Baker et al., A Database and Evaluation Methodology for Optical Flow, International Journal of Computer Vision, 2011.

[2] M. Brown and D.G. Lowe, Recognising Panoramas. ICCV, 2003.

[3] D.J. Butler et al., A Naturalistic Open Source Movie for Optical Flow Evaluation. ECCV, 2012.

[4] C. Dong et al., Image Super-Resolution Using Deep Convolutional Networks. ECCV, 2014.

[5] A. Dosovitskiy et al., FlowNet: Learning Optical Flow with Convolutional Networks. IEEE International Conference on Computer Vision (ICCV), Dec 2015.

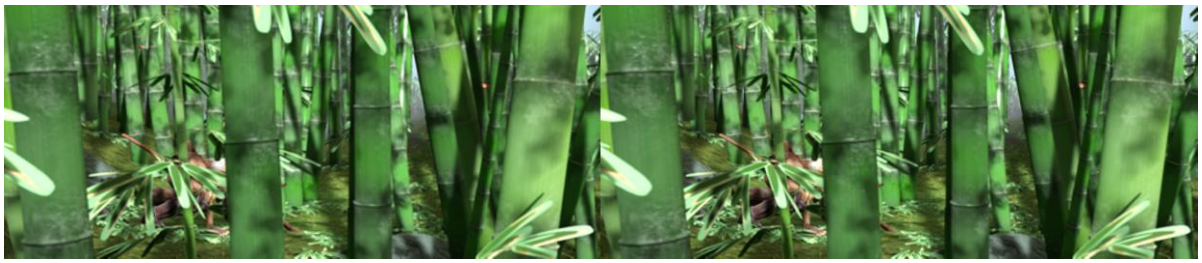
[6] A. Gupta et al., Enhancing and Experiencing Spacetime Resolution with Videos and Stills. International Conference on Computational Photography (ICCP), 2009.

[7] B.K. Horn, B.G. Schunck, Determining Optical Flow. Proc. SPIE, 1981.

[8] V. Kwatra et al., Graphcut Textures: Image and Video Synthesis Using Graph Cuts. SIGGRAPH, 2003.
[9] B. Lucas and T. Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision. Proceedings of Imaging Understanding Workshop, 1981.
[10] P. Perez, M. Gangnet, A. Blake, Poisson Image Editing. SIGGRAPH, 2003.
[11] T. Porter, T. Duff, Compositing Digital Images. SIGGRAPH, 1984.
[12] D. Sun, S. Roth, M.J. Black, Secrets of Optical Flow Estimation and Their Principles. CVPR, 2010.
[13] R. Szeliski and H-Y Shum, Creating Full View Panoramic Image Mosaics and Environment Maps. SIGGRAPH, 1997.
[14] C. Zitnick, N. Jojic, S.B. Kang, Consistent Segmentation for Optical Flow Estimation. ICCV, 2005.

(a)

(b)



(c)

(d)

Figure 3. The left image is the downsampled and upsampled image. The right image is the synthesized superresolution image. a) and b) are from “Backyard in the Middlebury dataset. c) and d) are from “bamboo_3” in the Sintel dataset. a) and c) are downsampled by a factor of 2. b) and d) are downsampled by a factor of 4.

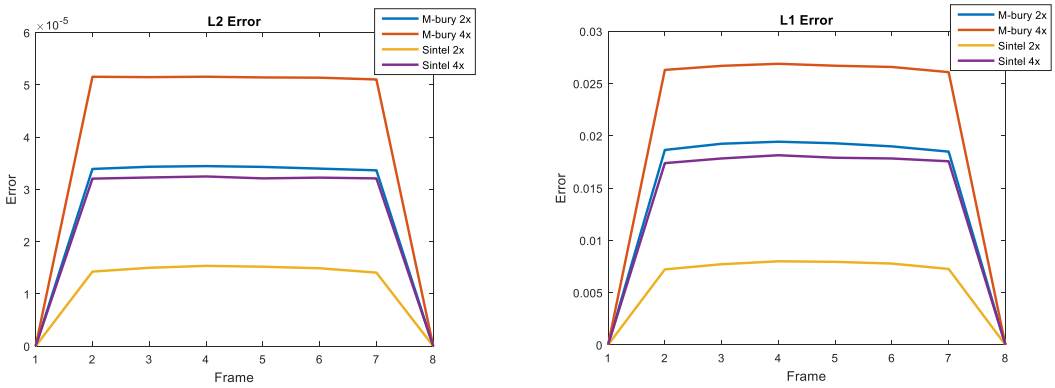
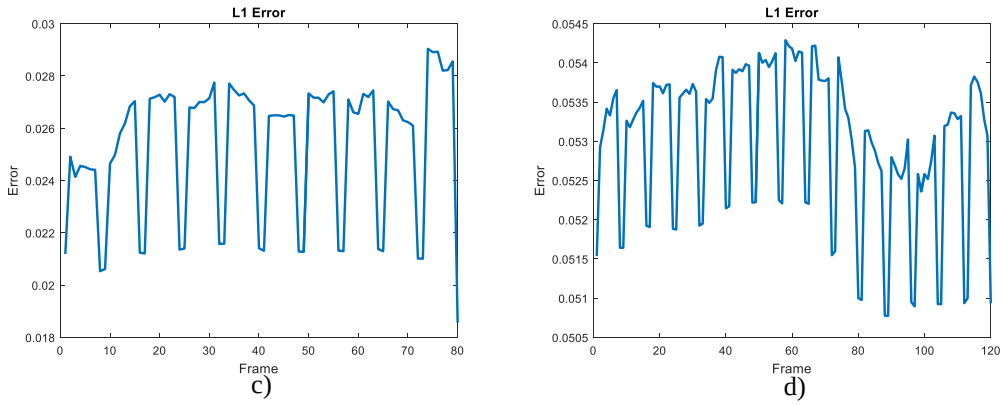


Figure 4. The left graph is L2 error, the right graph is L1 error of the Middlebury and Sintel dataset images.

a)

b)



c) d)

Figure 5. a) and b) show the original frame on the left and the supperresolution frame on the right after downsampling by a factor of 2. c) and d) show the L1 error for a) and b) respectively.