Final Report

Factored Time-Lapsed Video

Bin Lu
Department of Computer Science
University of San Francisco
blu2@usfca.edu

Feng Wang
Department of Computer Science
University of San Francisco
fwang7@usfca.edu

May 15, 2014

1 Introduction

1.1 Goal

In this project, we focus on time-lapse sequences of outdoor scenes under clear-sky conditions. The camera viewpoint is fixed and the scene is mostly stationary, hence the predominant changes in the sequence are changes in illumination. Under these assumptions we have developed a method that provides a complete decomposition of the original dataset into shadow, illumination, and reflectance components. We call this representation Factored Time- Lapse Video (FTLV).

Our method begins by locating the onset of shadows using the time-varying intensity profiles at each pixel. We identify points in shadow and points in direct sunlight to separate skylight and sunlight components, respectively. We then analyze these spatiotemporal volumes using matrix factorization. The results are basis curves describing the changes of intensity over time, together with per-pixel offsets and scales of these basis curves, which capture spatial variation of reflectance and geometry. The resulting representation is compact, reducing a time-lapse sequence to three images, two basis curves, and a compressed representation for shadows.

FTLVs are an intrinsic image-like scene representation that allow a user to analyse, reconstruct and modify illumination, reflectance, or geometry. The shadows may be discarded or retained, depending on the ultimate application, while other "outliers" such as pedestrians or cars are implicitly ignored. FTLVs

can also be used for various computer vision tasks such as background modeling, image segmentation, and scene reconstruction. After several applications for FTLVs, including relighting, shadow removal, advanced image editing, and pictorial rendering.

FLTVs are a compact, intuitive, factored representation for time-lapse sequences that separate a scene into its reflection, illumination and geometry factors. We are going to take this advantage to implement editing image and compare it with the original one to verify the result.

1.2 Milestones

First, we are going to implement shadow estimation which is the foundation to finish other parts. Based on the proper shadow estimation, it would be easy to tell which pixel is affected by skylight and which one is affected by sunlight.

Result: A sequence of frames where the pixels are presented as 0 or 1 to show whether under shadow or not.

Second, we would like to implement skylight factorization and sunlight factorization using the result of shadow estimation. Basically we will separate a video(sequence of frames) into two images and two curves and use it to reconstruct the video.

Result: A image of skylight and image of sunlight. A foundation curve of skylight and a foundation curve of sunlight. A reconstructed video.

Third, we are going to implement it with editing which will use Photoshop or other tools to replace some part of skylight/sunlight image with USF logo. Then apply with the same shadow estimation, skylight curve and sunlight curve to reconstruct the video.

Result: in the reconstructed video, the variation of skylight and sunlight on editing part(USF logo) should be told properly. Comparing with original video, ideally, we would not tell too much differences between them especially for the editing part.

2 Approach

2.1 Representation

Our goal is to decompose the space-time volume F(t) of the time-lapse image sequence into factors that will enable us to analyze and edit the scene. As the

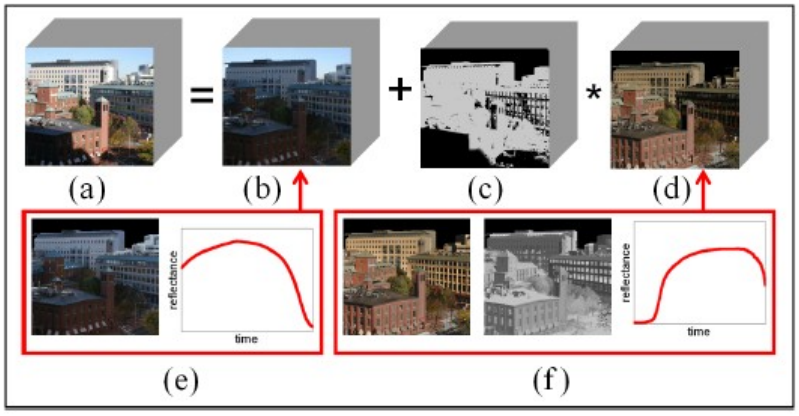

Figure 1: An overview of the FTLV factorization

sun moves, the observations at every pixel in the time-lapse sequence result in a continuous appearance profile.

Under the clear-sky assumption we can approximate the illumination as a sum of an ambient term corresponding to sky illumination and a single-directional light source corresponding to the radiance of the sun. Using the linearity of the rendering equation, the spatiotemporal volume F(t) can therefore be expressed as a sum of the sky light and sunlight components:

$$F(t) = I_{\text{sky}}(t) + S_{\text{sun}}(t) I_{\text{sun}}(t)$$

Here $S_{\text{sun}}(t)$ is the shadowing term that describes if a pixel is in shadow (and therefore has no sunlight contribution) or not.

In order to estimate $I_{\text{sky}}(t)$, we look to find a single illumination-vs.-time basis curve $H_{\text{sky}}(t)$ for the entire image sequence, such that the appearance of any shadowed pixel P_i may be reproduced as:

$$I_{\text{sky,i}}(t) = W_{\text{sky,i}} H_{\text{sky}}(t)$$

We call the matrix W_s of per-pixel weights the skylight image, and the basis curve $H_s(t)$ the skylight basis curve.

Based on our insight, we approximate $I_{\text{sun}}(t)$ with a single basis curve $H_{\text{sun}}(t)$ scaled by per-pixel weights W_{sun} for the normal dependence of the appearance profile:

$$I_{\text{sun,i}}(t) = W_{\text{sun,i}} H_{\text{sun}}(t)$$

We call weight matrix W_s the sunlight image, the basis curve $H_{\text{sun}}(t)$ the sunlight basis curve and the offset the shift map. Since the sun is a strong directional light source, $H_{\text{sun}}(t)$ is an estimate of the 1-D slice of surface reflectance corresponding to the camera viewpoint and the arc described by the sun's motion.

Finally, we can get:

$$F(t) = W_{\text{sky}} H_{\text{sky}}(t) + S_{\text{sun}}(t) * W_{\text{sun}} H_{\text{sun}}(t)$$

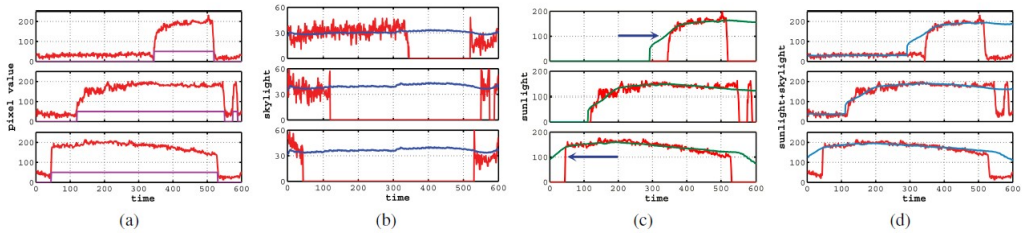

Figure 2: (a) Appearance profiles (red) and shadow functions (purple) of points A, B, and C (top to bottom). (b) Appearance profiles when points A, B, and C (top to bottom) are in shadow (red) and the estimated skylight curves (blue). Note the different scale on the intensity axis. (c) Appearance profiles when points A, B, and C are in the sun (red) and the estimated sunlight curves (green). The arrows indicate the direction of the shift of the basis curves. (d) Original appearance profiles (red) and the sum of sunlight and skylight curves (blue).

2.2 Shadow Estimation

Figure 2 shows one frame of a time-lapse sequence of the Santa Catalina Mountains in Arizona. We will use this frame and pixels A, B, and C throughout our discussion. All computations are performed in RGB color space and independently on the three color channels. Processing has provided the color() function to easily do that.

We use these discontinuities in the appearance profiles to estimate shadow images. We first compute the median value $m_{\min}$ of the n smallest intensities at each pixel. We typically assume that each point is in shadow at least 20frames in the sequence. We set the shadow function $S_i(t)$ (purple curve in Figure 2(a)) to one for each $F_i(t) > km_{\min}$ and to zero otherwise. We heuristically found that k = 2 worked well for most of the sequences in our experiments.

2.3 Skylight Factorization

Multiplying the appearance profiles $F_i(t)$ in Figure 2(a) by $(1 - S_i(t))$ yields the appearance profile of points in shadow as shown in Figure 2(b). We perform NMF using F(t) as the data matrix and use (1 - S(t)) as the matrix to tell which one is under shadow. This ensures that we consider only shadowed pixels.

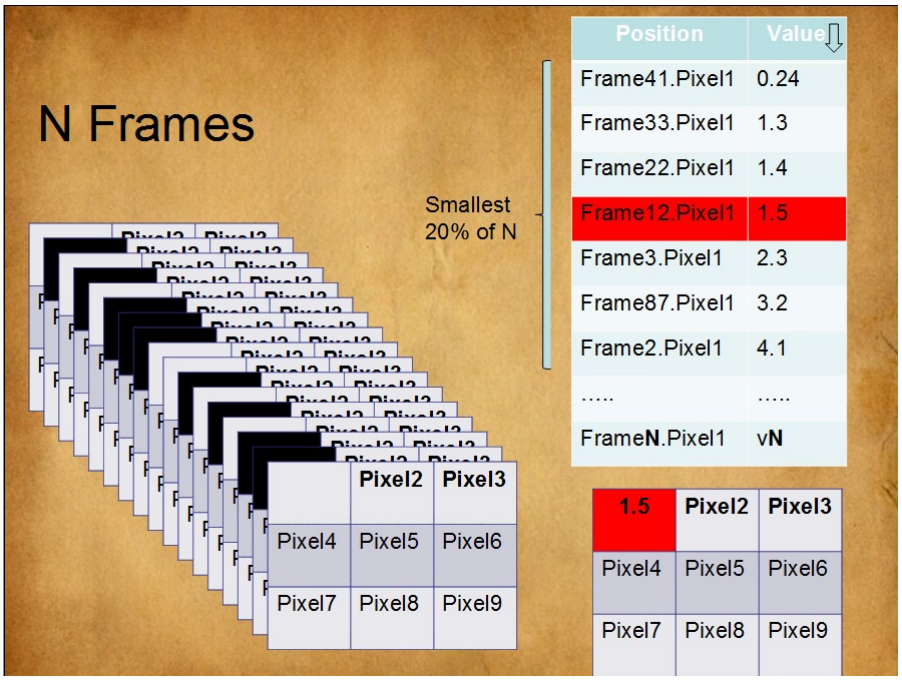

Figure 3: Example of shadow estimations

2.4 Sunlight Factorization

The reconstructed images $I_{\text{sky}}(t)$ are subtracted from the original data F(t) and the result is clamped to 0 to form the matrix $I_{\text{sun}}(t)$. Multiplying $I_{\text{sun,i}}(t)$ by $S_i(t)$ yields the appearance profile of frames when the points are illuminated only by sunlight as shown in Figure 2(c). Using $I_{\text{sun}}(t)$ as the data matrix and use S(t) to tell whether pixel is under shadow to ensures that only the sunlight component at every pixel is considered during the factorization.

3 Methodology

The computational framework the paper used to decompose appearance profiles into W and H factors is ACLS which is similarly well suited for our task of decomposing appearance profiles, since its ability to incorporate domain-dependent constraints—such as nonnegativity, sparseness, and smoothness—leads to stable and intuitive decompositions.

Formally, each application of ACLS decomposes an $m \times n$ data matrix F(t), where m is the number of pixels in the image and n is the number of frames in the time-lapse sequence, into the product of an $n \times k$ weight matrix W and a $k \times m$ basis matrix H(t). The algorithm takes k, the number of basis curves, as input. We set $k = 1$ for the decompositions shown in this paper. We apply ACLS in two separate steps to factor a matrix of measured spatiotemporal appearance

```
int M = 100; // rows in X
int T = 300; // columns in X
int R = 1; // number of basis columns in W

Matrix X = new Matrix(M, T);
MatrixUtilities.randomize(X);
Matrix W = new Matrix(M, R);
MatrixUtilities.randomize(W);
Matrix H = new Matrix(R, T);
MatrixUtilities.randomize(H);

NMFCostKL nmfSolver = new NMFCostKL(X, W, H, "solver1");

int iterations = 200;
for (int i = 0; i < iterations; i++) {
    nmfSolver.doLeftUpdate();
    nmfSolver.doRightUpdate();
    nmfSolver.printReconstructionError();
}
```

Figure 4: NMF adaptation

profiles F(t), first factoring $I_{\text{sky}}(t)$) and then solving for $I_{\text{sun}}(t)$).

However, in our implementation we used regular NMF which is similar to ACLS to solve this problem.

3.1 NMF

Non-negative matrix factorization (NMF), also non-negative matrix approximation is a group of algorithms in multivariate analysis and linear algebra where a matrix V is factorized into (usually) two matrices W and H, with the property that all three matrices have no negative elements. This non-negativity makes the resulting matrices easier to inspect. Since the problem is not exactly solvable in general, it is commonly approximated numerically[Wikipedia].

There are several ways in which the W and H may be found: Lee and Seung's multiplicative update rule has been a popular method due to the simplicity of implementation. In our implementation we adapted this algorithm to make matrix factorization.

3.2 Adaptation

Thanks Brian K. Vogel for sharing his code of NMF implementations. In Figure 4, we can see how easily we integrated NMF code into our project. In addition, since there is no confident matrix in regular NMF rather than ACLS, we still need to use the matrix of shadow estimation to separate them apart into pixels affected by sunlight and skylight.

Figure 5: Comparison between original video and reconstructed one

Figure 6: Editing example: Add USF Logo on the building

4 Results

As we can see in Figure 5, we have successfully reconstructed the video using the generated images and curves. A major benefit of FTLV is that it factors the scene into physically meaningful components, each of which can be edited to create interesting. In Figure 6, we put a USF logo into the building and it is affected by the sunlight and skylight and varied as well as the original part of the building.

5 Discussion

We optimized the algorithm of shadow estimation by dynamically adjusting the threshold. The shadow video looks better with less noise and became more smooth.

Regular NMF is the first time to apply to this project rather than ACLS which was used in the original paper implementation.