Temporality Transfer Between Media

Film Temporal Quality Appropriation Through Shot Analysis & Camera Reconstruction

Mengfei Wang, Xiaobi (Iris) Pan

SCI 6432: Quantitative Aesthetics: Design as Signal

Instructed by Panagiotis Michalatos

Abstract

Our project aims to understand the temporal quality of a film and transfer that into other media. For each film, its temporality is examined at different levels of resolution---from the entire film, to a selected group of shots, to a single shot. Then we try to reproduce the same temporality in a new context. In this case, architectural visualization is chosen as our application.

Part1: Shot-Cut Finding & Shot Classification

The first step is to identity shot cuts in a film to help us understand the overall film structure. Using openCVsharp in Visual Studio, our program was able to spot shot cuts when the total color difference between two successive frame screenshots exceeds a certain threshold (in our case, 30.00) and auto-save the frames of each continuous shot into separate folders. To improve computational efficiency, frame screenshot resolution was reduced to 320*240.

```
int smallX = 320;
int smallY = 240;
Mat opFlow = new Mat(new OpenCvSharp.Size(smallX, smallY), MatType.CV_32FC2);
```

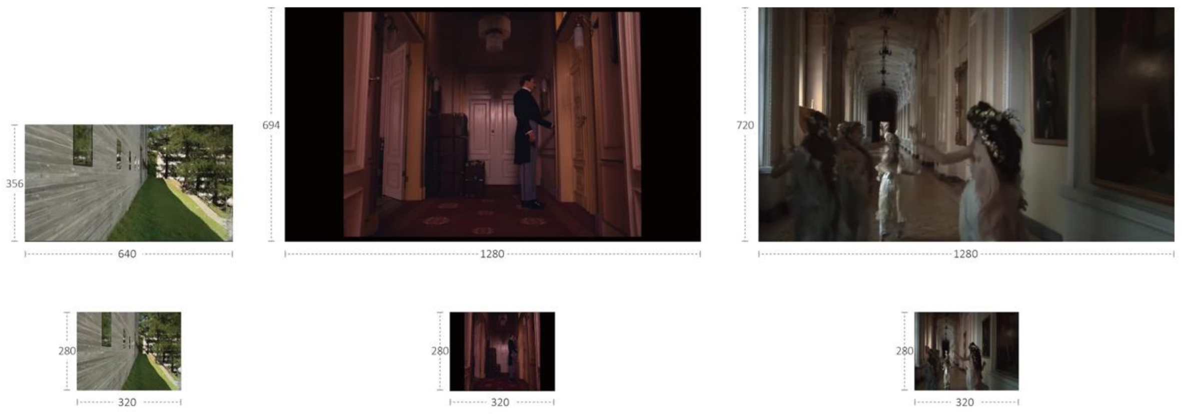
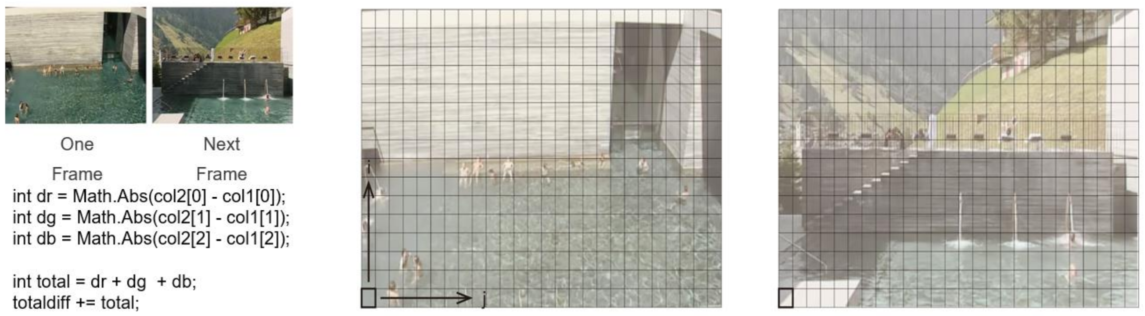


Fig.1 reduce frame resolution for computational efficiency



```
int dr = Math.Abs(col2[0] - col1[0]);
int dg = Math.Abs(col2[1] - col1[1]);
int db = Math.Abs(col2[2] - col1[2]);

int total = dr + dg  + db;
totaldiff += total;


totaldiff > 30.00
```

Fig.2 calculate frame total difference for shot cut finding

The movie we chose as subject for study belongs to the genre of Architecture Documentary, Peter Zumthor The Thermae of Stone. (25’32’’ Whole Film). It was divided into 97 distinct shots by our algorithm.

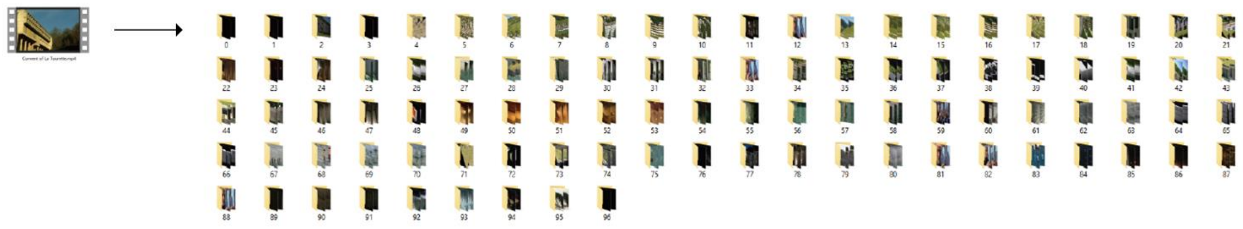


Fig.3 shot cut finding result: screenshots of each shot in separate folders

Then we classified the shots manually (though we hope to enable AI classification in the future) into 4 categories, shots about the architecture itself, shots about the architect, shots about the architecture model, and shots about others (interview, historic materials, etc.) We made an interactive timeline of shot categorization as below to illustrate the distribution of each category of shots in the film in a temporal sequence. Since we were only interested in the shots about the building itself, other categories were filtered out.

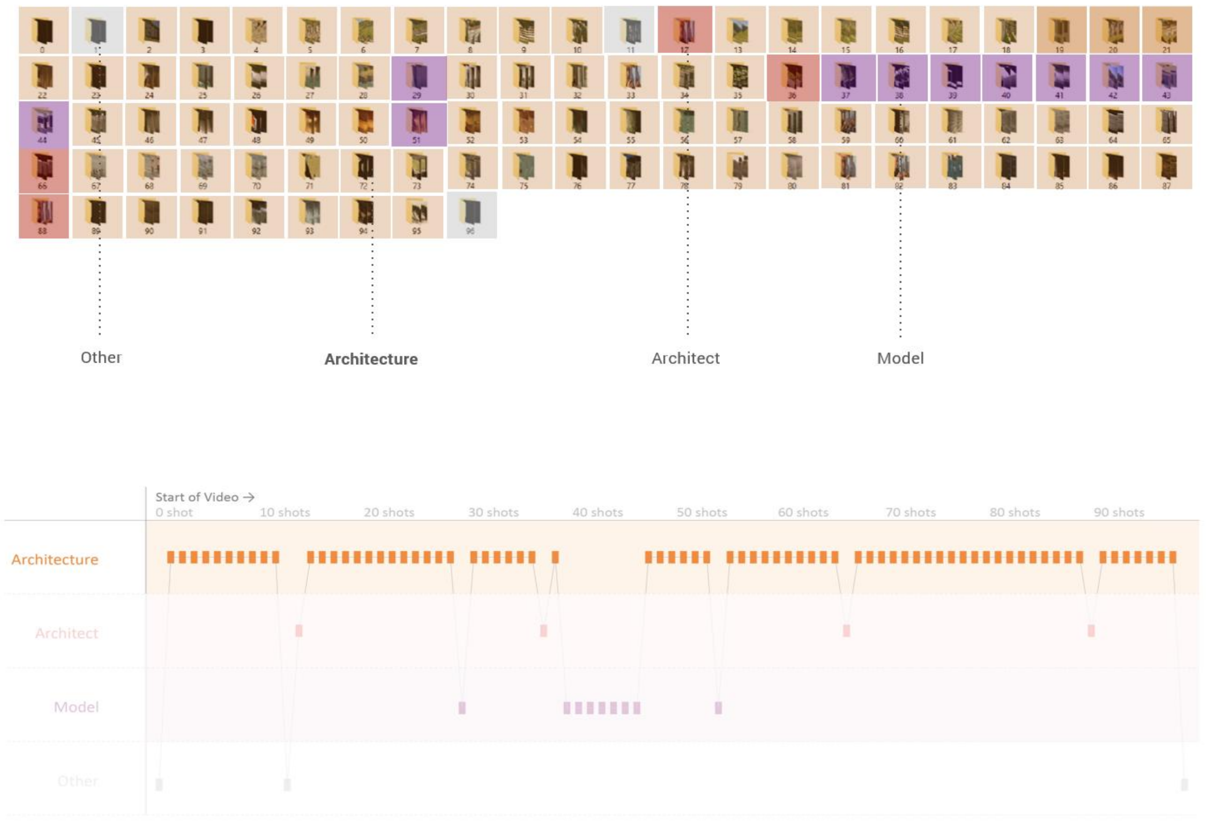


Fig.4 shot categorization by content

Part2: Static Diagram of Interested Shots

For the architecture-related shots, we produced a static diagram for each of them to understand the basic film properties (light/color/movement…) in each shot and their change with time. Rhinoceros, Grasshopper, and C# were used to visit each frame picture in a folder, cut a 1-pixel-wide section at its middle and place the sections chronologically next to one another. Thereby, we got a miniature of each shot that understored its time-related cinematographic qualities.

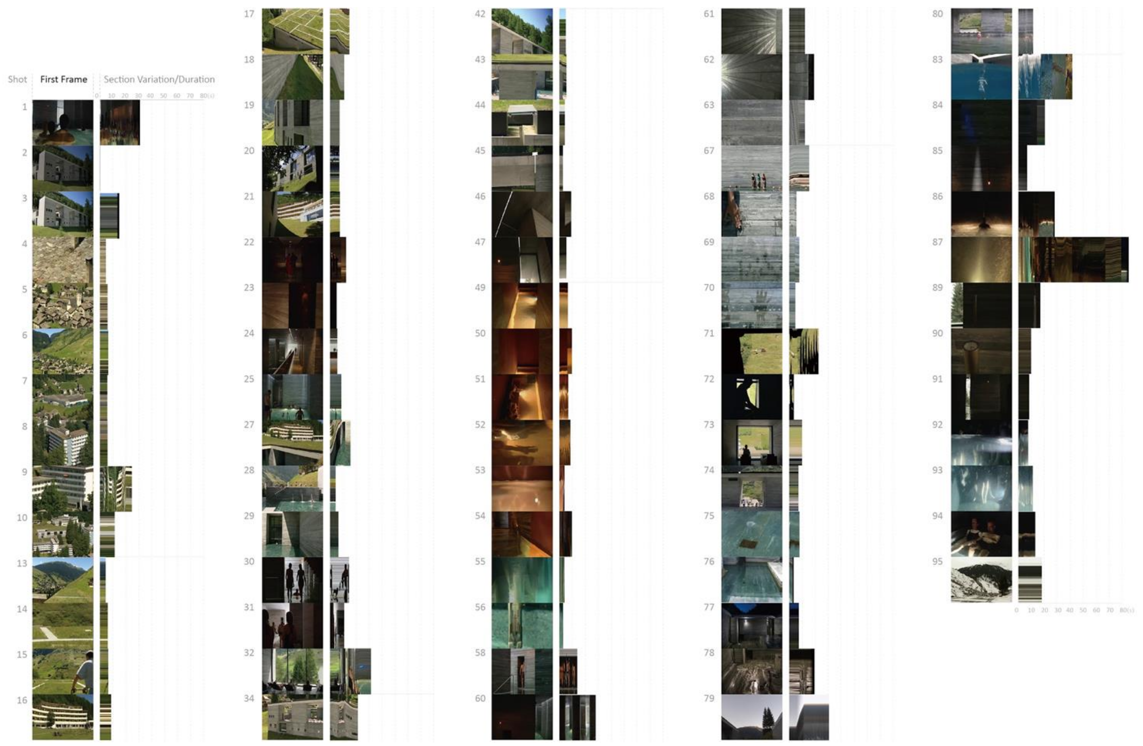
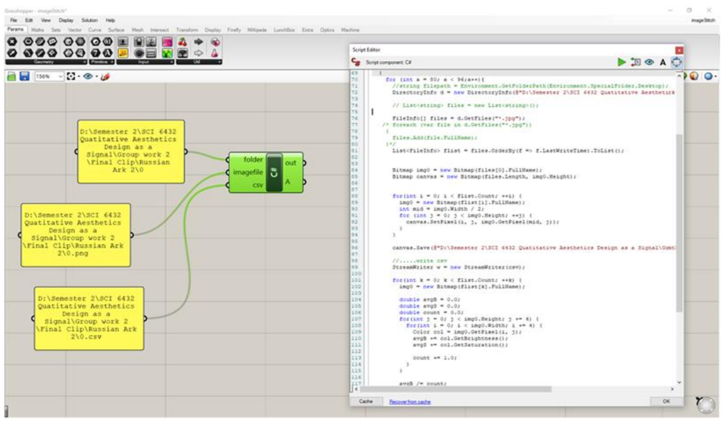


Fig.5 static diagram for all the architecture-related shots


File Edit View Display Solution Help
Params Maths Sets Vector Curve Surface Mesh Intersect Transform Display Firefly Millipede LunchBox Extra Optics Machine
folder
imagefile
csv
out
A
Script Editor
Script component: C#
Cache
Recover from cache
OK

```
D:\Semester 2\SCI 6432
Quatitative Aesthetics
Design as a
Signal\Group work 2
\Final Clip\Russian
Ark 2\0
```

```
D:\Semester 2\SCI 6432
Quatitative Aesthetics
Design as a
Signal\Group work 2
\Final Clip\Russian Ark
2\0.png
```

```
D:\Semester 2\SCI 6432
Quatitative Aesthetics
Design as a
Signal\Group work 2
\Final Clip\Russian Ark
2\0.csv
```

```
  {
    for (int a = 80; a < 96;a++){
      //string filepath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
      DirectoryInfo d = new DirectoryInfo(@"D:\Semester 2\SCI 6432 Quatitative Aesthetic

      // List<string> files = new List<string>();

      FileInfo[] files = d.GetFiles("*.jpg");
   /* foreach (var file in d.GetFiles("*.jpg"))
    {
      files.Add(file.FullName);
    }*/
      List<FileInfo> flist = files.OrderBy(f => f.LastWriteTime).ToList();


      Bitmap img0 = new Bitmap(files[0].FullName);
      Bitmap canvas = new Bitmap(files.Length, img0.Height);


      for(int i = 0; i < flist.Count; ++i) {
        img0 = new Bitmap(flist[i].FullName);
        int mid = img0.Width / 2;
        for (int j = 0; j < img0.Height; ++j) {
          canvas.SetPixel(i, j, img0.GetPixel(mid, j));
        }
      }

      canvas.Save(@"D:\Semester 2\SCI 6432 Quatitative Aesthetics Design as a Signal\Gmt

      //.....write csv
      StreamWriter w = new StreamWriter(csv);

      for(int k = 0; k < flist.Count; ++k) {
        img0 = new Bitmap(flist[k].FullName);

        double avgB = 0.0;
        double avgS = 0.0;
        double count = 0.0;
        for(int j = 0; j < img0.Height; j += 4) {
          for(int i = 0; i < img0.Width; i += 4) {
            Color col = img0.GetPixel(i, j);
            avgB += col.GetBrightness();
            avgS += col.GetSaturation();

            count += 1.0;
          }
        }

        avgB /= count;
```

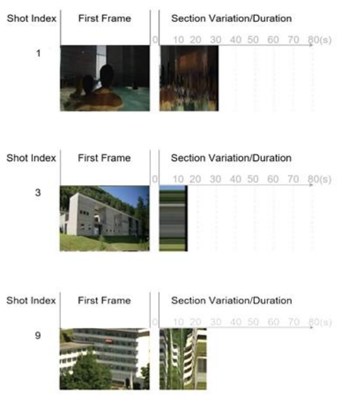


Fig.6 grasshopper screenshot & single shot diagram illustration

Looking at these static shot diagrams, we found that the temporal variations of film come from two aspects: the movement of camera and the movement of object/human, which spores 4 combinations.

	Object/human still	Object/human Moving
Camera Still	1	2
Camera Moving	3	4

Table 1: four combinations of camera/object movement

In our case, since it is hard for our software to analyze human motion, we decided to focus only on the camera motion. So we zoomed in on the shots of the 1st, and 3rd combination.

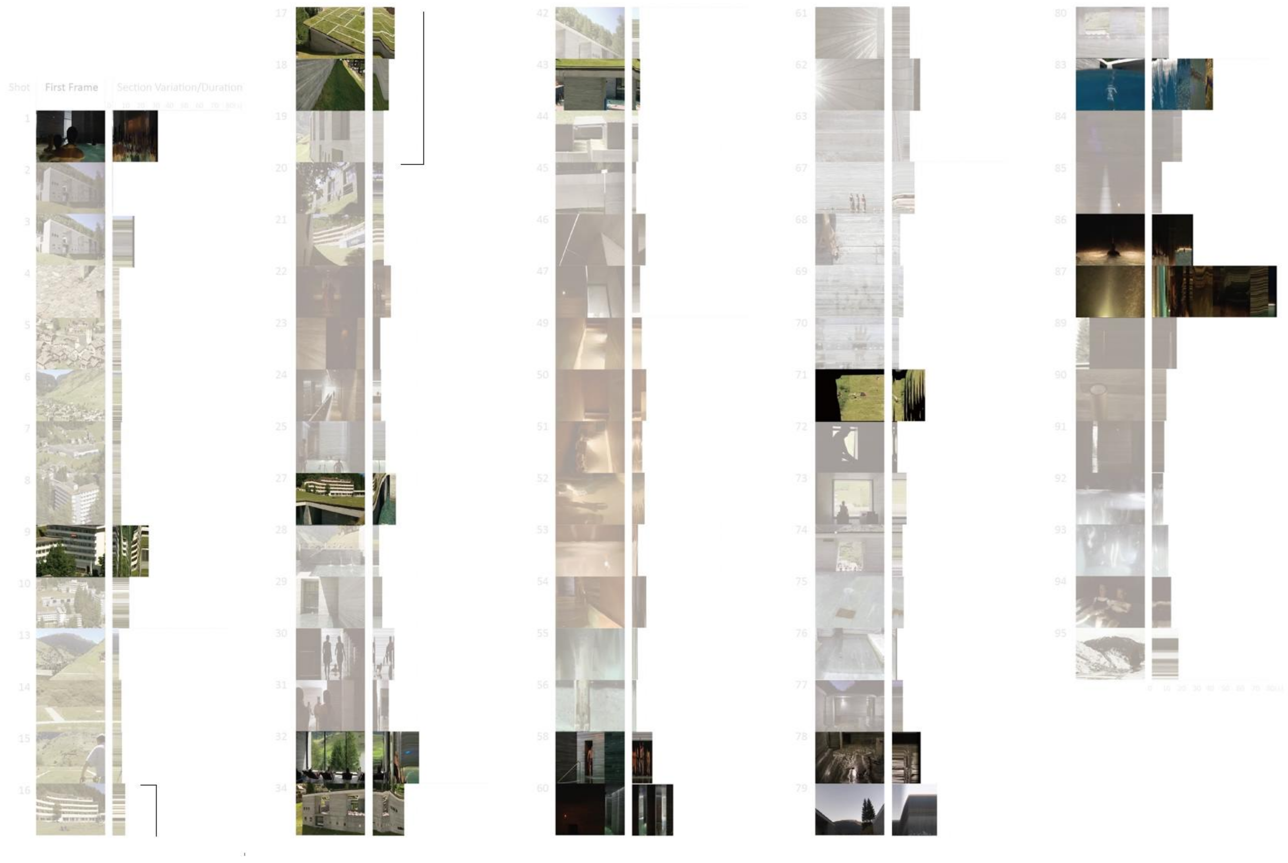


Fig.7 target group of shots to study its temporality: shot 16th -19th

Finally, we selected Shot 16th- 19th as our target group of shots to derive temporality from. The sequence has 2 static shots (1st combination, table 1) and 2 moving shots (3rd combination, table 1). The movement and staticity within each shot, as well as their chronological composition as an entity, became the focus of our study.

Part3: Camera Interpretation & Reconstruction

We used a tool called structure from motion (VisualSFM) to decipher the camera in each shot. By feeding the software a continuous group of frames, it was able to run feature matches between different viewpoints to interpret the camera position, angle, and focal length in each frame, then output the 3D camera point clouds and the csv files.

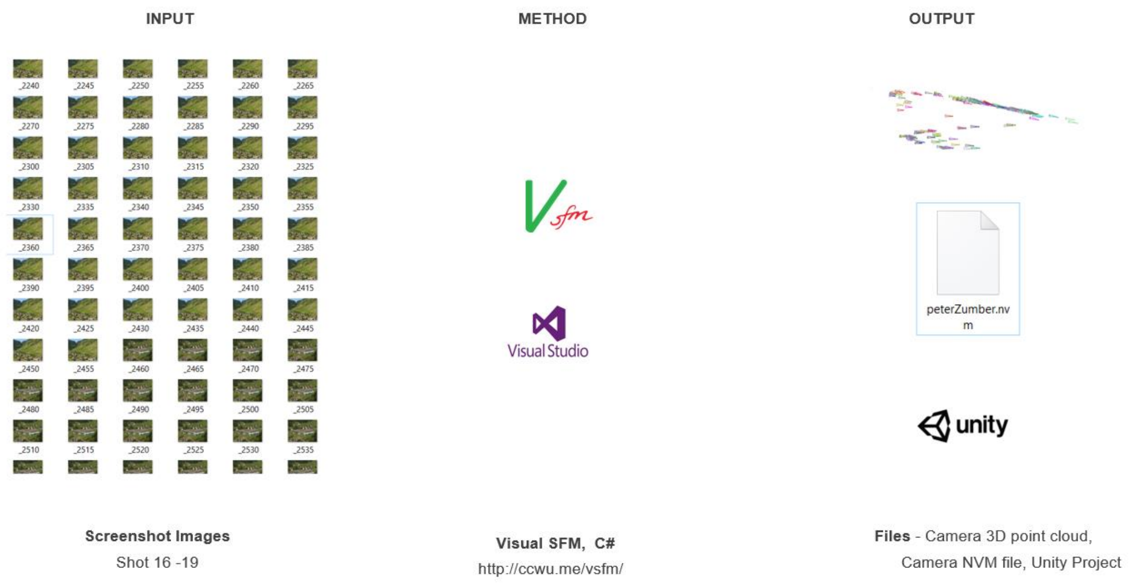
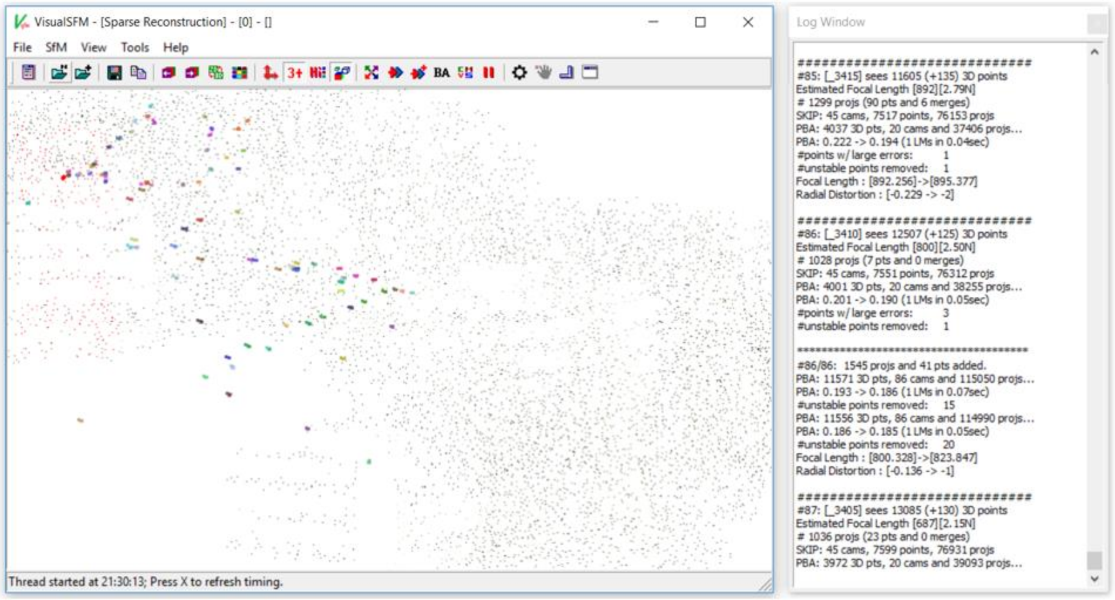


Fig.8 workflow in camera interpretation & reconstruction



VisualSFM - [Sparse Reconstruction] - [0] - []
File SfM View Tools Help
Thread started at 21:30:13; Press X to refresh timing.
Log Window

```
################################
#85: [_3415] sees 11605 (+135) 3D points
Estimated Focal Length [892][2.79N]
# 1299 projs (90 pts and 6 merges)
SKIP: 45 cams, 7517 points, 76153 projs
PBA: 4037 3D pts, 20 cams and 37406 projs...
PBA: 0.222 -> 0.194 (1 LMs in 0.04sec)
#points w/ large errors:        1
#unstable points removed:     1
Focal Length : [892.256]->[895.377]
Radial Distortion : [-0.229 -> -2]

################################
#86: [_3410] sees 12507 (+125) 3D points
Estimated Focal Length [800][2.50N]
# 1028 projs (7 pts and 0 merges)
SKIP: 45 cams, 7551 points, 76312 projs
PBA: 4001 3D pts, 20 cams and 38255 projs...
PBA: 0.201 -> 0.190 (1 LMs in 0.05sec)
#points w/ large errors:        3
#unstable points removed:     1

****************************************
#86/86:  1545 projs and 41 pts added.
PBA: 11571 3D pts, 86 cams and 115050 projs...
PBA: 0.193 -> 0.186 (1 LMs in 0.07sec)
#unstable points removed:     15
PBA: 11556 3D pts, 86 cams and 114990 projs...
PBA: 0.186 -> 0.185 (1 LMs in 0.05sec)
#unstable points removed:     20
Focal Length : [800.328]->[823.847]
Radial Distortion : [-0.136 -> -1]

################################
#87: [_3405] sees 13085 (+130) 3D points
Estimated Focal Length [687][2.15N]
# 1036 projs (23 pts and 0 merges)
SKIP: 45 cams, 7599 points, 76931 projs
PBA: 3972 3D pts, 20 cams and 39093 projs...
```

Fig.9 VisualSFM screenshot: sparse reconstruction

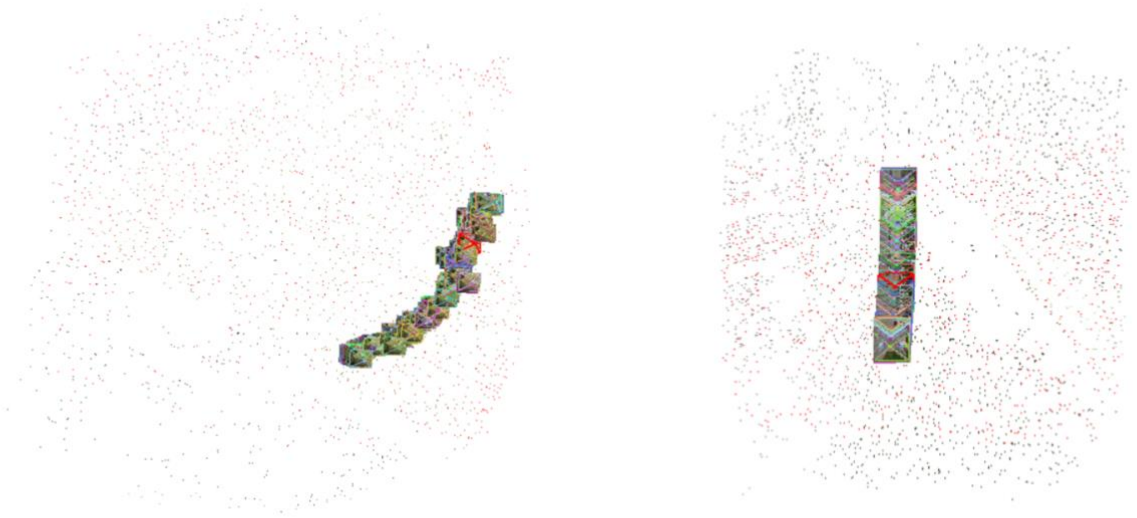
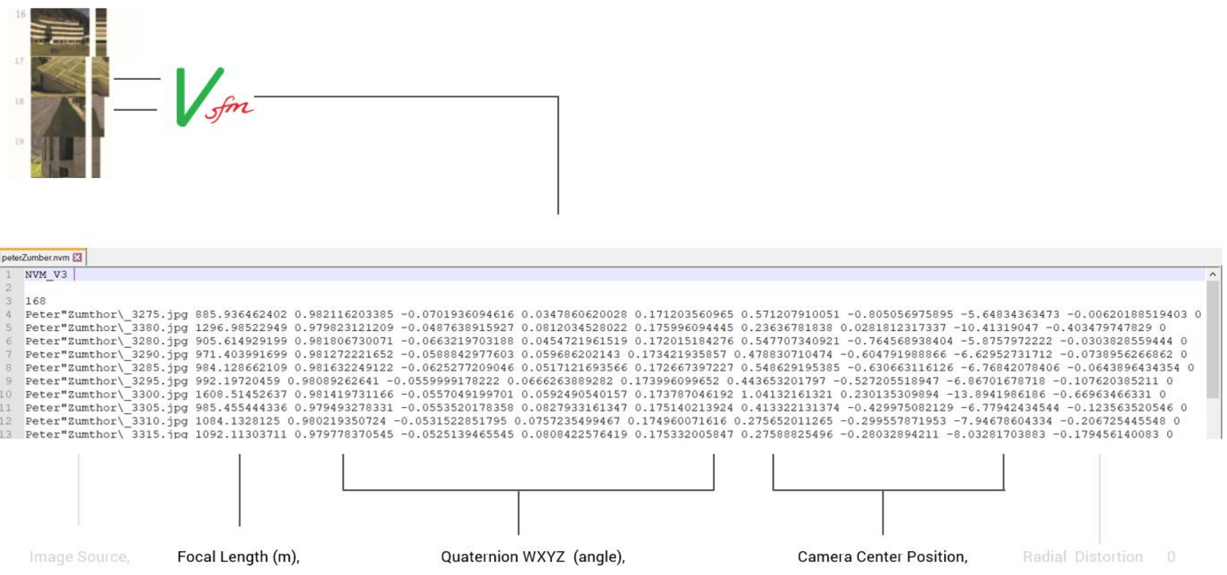


Fig.10 VisualSFM camera interpretation result: Shot 17th, 18th

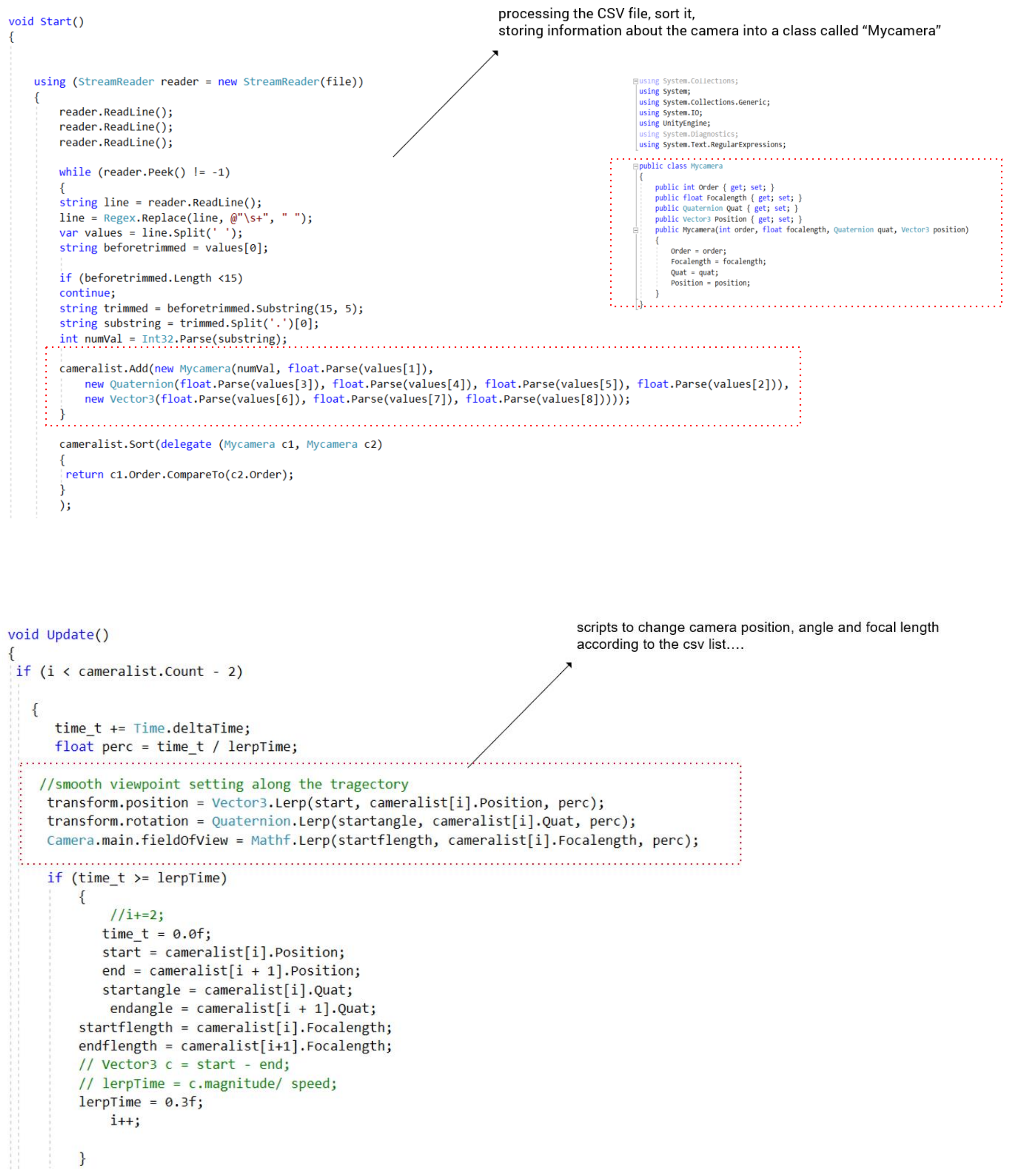
Once the camera data was retrieved for the moving shots, we wrote C# script in unity to process the CSV list, sort it in temporal order (in our case we used the frame index, the first column of the CSV list, to sort), and store the focal length, Quaternion angle, and position of a series of cameras into a list of objects in a class called "Mycamera". Then the update() function assigns the Main Camera in unity the corresponding position, angle and focal length from the list of cameras at different moments. In this manner, we managed to reconstruct the camera movement in shot 17th and 18th.



```
NVM_V3

168
Peter"Zumthor\_3275.jpg 885.936462402 0.982116203385 -0.0701936094616 0.0347860620028 0.171203560965 0.571207910051 -0.805056975895 -5.64834363473 -0.00620188519403 0
Peter"Zumthor\_3380.jpg 1296.98522949 0.979823121209 -0.0487638915927 0.0812034528022 0.175996094445 0.23636781838 0.0281812317337 -10.41319047 -0.403479747829 0
Peter"Zumthor\_3280.jpg 905.614929199 0.981806730071 -0.0663219703188 0.0454721961519 0.172015184276 0.547707340921 -0.764568938404 -5.8757972222 -0.0303828559444 0
Peter"Zumthor\_3290.jpg 971.403991699 0.98127221652 -0.058884297760 3 0.059686202143 0.173421935857 0.478830710474 -0.604791988866 -6.62952731712 -0.0738956266862 0
Peter"Zumthor\_3285.jpg 984.128662109 0.981632249122 -0.0625277209046 0.0517121693566 0.172667397227 0.548629195385 -0.630663116126 -6.76842078406 -0.0643896434354 0
Peter"Zumthor\_3295.jpg 992.19720459 0.98089262641 -0.0559999178222 0.0666263889282 0.173996099652 0.443653201797 -0.527205518947 -6.86701678718 -0.107620385211 0
Peter"Zumthor\_3300.jpg 1608.51452637 0.981419731166 -0.0557049199701 0.0592490540157 0.173787046192 1.04132161321 0.230135309894 -13.8941986186 -0.66963466331 0
Peter"Zumthor\_3305.jpg 985.455444336 0.979493278331 -0.0553520178358 0.0827933161347 0.175140213924 0.413322131374 -0.429975082129 -6.77942434544 -0.12356352054 6 0
Peter"Zumthor\_3310.jpg 1084.1328125 0.980219350724 -0.0531522851795 0.0757235499467 0.174960071616 0.275652011265 -0.299557871953 -7.94678604334 -0.206725445548 0
Peter"Zumthor\ 3315.jpg 1092.11303711 0.979778370545 -0.0525139465545 0.0808422576419 0.175332005847 0.27588825496 -0.28032894211 -8.03281703883 -0.179456140083 0
```

Fig.11 camera csv file exported from VisualSFM (in the format of NVM)



processing the CSV file, sort it,
storing information about the camera into a class called “Mycamera”

```
void Start()
{

    using (StreamReader reader = new StreamReader(file))
    {
        reader.ReadLine();
        reader.ReadLine();
        reader.ReadLine();

        while (reader.Peek() != -1)
        {
        string line = reader.ReadLine();
        line = Regex.Replace(line, @"\s+", " ");
        var values = line.Split(' ');
        string beforetrimmed = values[0];

        if (beforetrimmed.Length <15)
        continue;
        string trimmed = beforetrimmed.Substring(15, 5);
        string substring = trimmed.Split('.')[0];
        int numVal = Int32.Parse(substring);

        cameralist.Add(new Mycamera(numVal, float.Parse(values[1]),
            new Quaternion(float.Parse(values[3]), float.Parse(values[4]), float.Parse(values[5]), float.Parse(values[2])),
            new Vector3(float.Parse(values[6]), float.Parse(values[7]), float.Parse(values[8]))));
        }

        cameralist.Sort(delegate (Mycamera c1, Mycamera c2)
        {
         return c1.Order.CompareTo(c2.Order);
        }
        );
```

```
using System.Collections;
using System;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using System.Diagnostics;
using System.Text.RegularExpressions;

public class Mycamera
{
    public int Order { get; set; }
    public float Focalength { get; set; }
    public Quaternion Quat { get; set; }
    public Vector3 Position { get; set; }
    public Mycamera(int order, float focalength, Quaternion quat, Vector3 position)
    {
        Order = order;
        Focalength = focalength;
        Quat = quat;
        Position = position;
    }
}
```

scripts to change camera position, angle and focal length
according to the csv list….

```
void Update()
{
 if (i < cameralist.Count - 2)

   {
     time_t += Time.deltaTime;
     float perc = time_t / lerpTime;

    //smooth viewpoint setting along the tragectory
     transform.position = Vector3.Lerp(start, cameralist[i].Position, perc);
     transform.rotation = Quaternion.Lerp(startangle, cameralist[i].Quat, perc);
     Camera.main.fieldOfView = Mathf.Lerp(startflength, cameralist[i].Focalength, perc);

     if (time_t >= lerpTime)
         {
            //i+=2;
           time_t = 0.0f;
           start = cameralist[i].Position;
           end = cameralist[i + 1].Position;
           startangle = cameralist[i].Quat;
            endangle = cameralist[i + 1].Quat;
         startflength = cameralist[i].Focalength;
         endflength = cameralist[i+1].Focalength;
         // Vector3 c = start - end;
         // lerpTime = c.magnitude/ speed;
         lerpTime = 0.3f;
            i++;

         }
```

Fig.12   main parts of unity C# code to process the CSV data and control the Main Camera

In the meantime, for the static shots 16th and 19th, we place the camera in corresponding positions and angles manually in unity for the same period as in the movie. Eventually, we compiled the two parts of csv files and achieved a continuous mapping from the movie clip to our unity architectural visualization application.

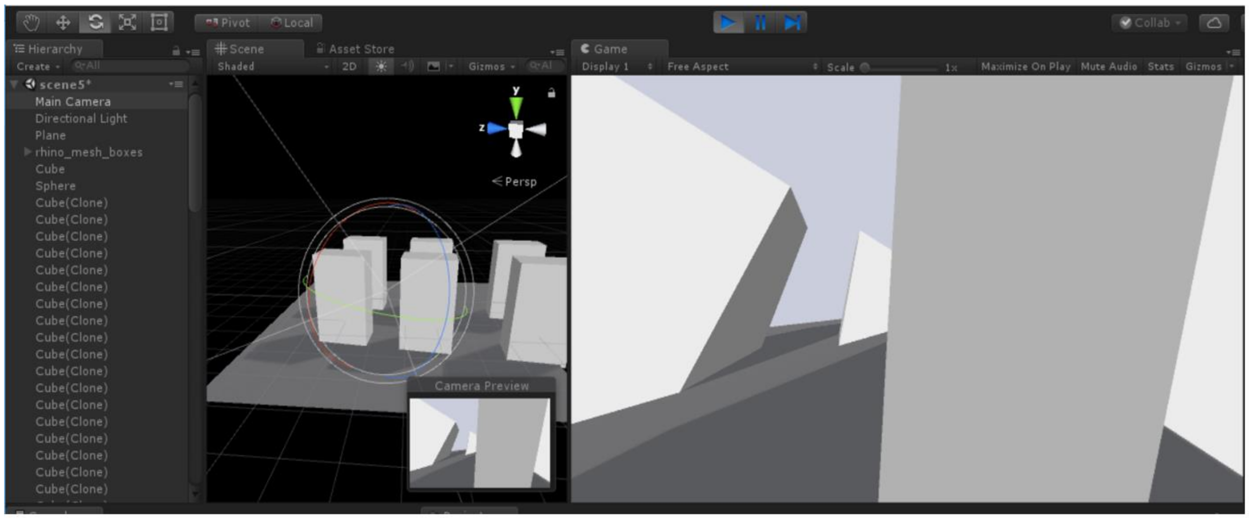


Fig.13 unity minimalist architectural scene setup as a demonstration

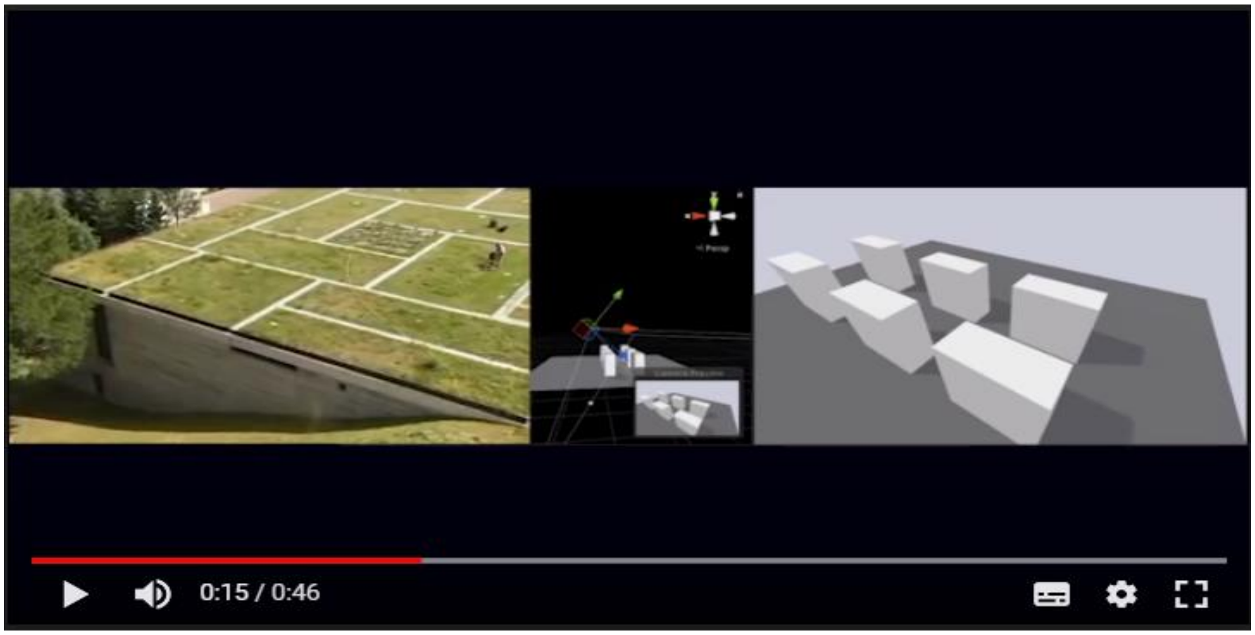


Fig.14 screenshot of the result video: transfer of film temporality into architectural visualization

In the future, we hope this technology can be applied in the VR realm for audiences to experience a building/space through the lens of a film. The camera motion sequence from an outstanding film can thus to translated into a new setting. Also, this can be utilized as a means of architectural design presentation for designers and architects.

Another potential would be to investigate other film properties such as light, color, or sound from a temporal angle, and then translate those aspects of film temporality into new areas.