Assignment 2
The Secrets of Optical Flow

Avneesh Mishra
avneesh.mishra@research.iiit.ac.in [*]

March 17, 2022

Contents

[*]M.S. by research - CSE, IIIT Hyderabad, Roll No: 2021701032

1 Basics of Optical Flow

1.1 Thought Experiments

1.1.1 Bullet Time - The Matrix (1999)

Question Explaining how optical flow is used for the bullet time scene in The Matrix (1999).

TL; DR Frame interpolation using optical flow.

Full Answer A series of cameras are arranged in the scene, starting from the point where the scene starts (in perspective view) and till the end. For this scene in particular, 120 cameras were used (watch reference video here [1]).

The actor is put in a green screen environment, at the center of those cameras. The action is performed at normal speed (human speed), with the cameras capturing the sequence with a slight time delay. Say the scene (the complete pose change of Neo) takes 1 second, but in the final movie, it is 10 seconds. Let's assume that the movie is 25 FPS and that the 120 camera captures are equally spaced in the entire band. That is, dividing $25 \times 10 = 250$ frames into 120 cameras giving approximately one real image (taken by the camera) followed by one interpolated image.

The problem now becomes finding the interpolated images. Theoretically, repeating the real image will create a jittery motion (it will seem to lag). The intermediate image can be estimated through *optical flow*. Since Neo (actor Keanu Charles Reeves) is on a green screen, tracking motion is relatively easier. For two real image captures given, the pixels of the intermediate frame is the half-displacement applied to each pixel. The total displacement is estimated through optical flow.

1.1.2 Painterly rendering - WDMC (1998)

Question Explaining the painterly effect scene in What Dreams May Come (1998).

TL; DR Transformation and tracking of particles are done using optical flow.

Full Answer Interestingly, the scene was probably captured using the work of Barbara Meier (Senior Lecturer at Brown University) in [Mei96] [6]. The painterly rendering pipeline uses the geometry of the object being painted (usually a triangle mesh). It gets particles (samples) in world space. Shaders are used to extract the **color** (color attributes and lighting), **size** (lighting, texture map and user-specified paintbrush attributes) and **orientation** (surface normal projection). A *brush image* showing the texture of the brush pattern is used.

Brush strokes are made on a canvas based on the attributes (emphasized above). These brush strokes are applied to the particles. In an animation, to maintain consistency, it is important to track these particles and use only them. Optical flow yields estimates of where the particles flow in successive frames. It can also be used to save computation by rearranging geometry using optical flow (the above-described process is computation heavy). An example application was developed as a project in DGP Toronto.

At the time the movie was developed, they must have chosen many properties by hand. Later developments in the field alleviate this load and make things much easier [HE04; Lin+10; OH11].

1.1.3 Lambertian ball

Assuming that the Lambertian ball doesn't have any features or texture on its surface (as is usually the case) and only reflects light (to create visual features).

Lighting fixed, ball rotating In the case where lighting is fixed, and the ball is rotating, *we will probably not get any optical flow* features out of it. This is because the ball has no texture, and since the lighting is fixed, the visual appearance (because of reflection) will remain the same. So the individual pixels will probably not change their values much, and hence there won't be any optical flow. This is nicely demonstrated in this clip where even the human would find it hard to recognize that the ball is rotating. This can be seen in Figure 1.

[1] `https://filmschoolrejects.com/the-matrix-bullet-time/`

[6] Source: news.brown.edu

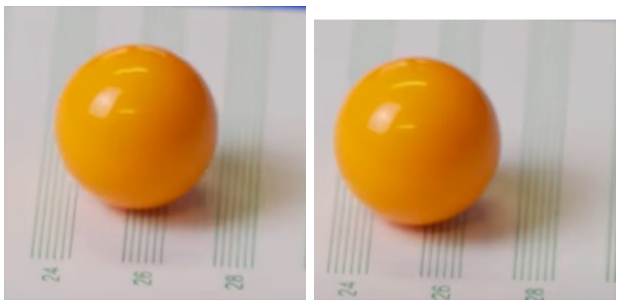

(a) 0:13 (b) 0:15

Figure 1: Rotating Lambertian ball
The two images are taken at different timestamps in the clip. The orange ball is rotating, but it can't clearly be said from looking at the two images.

Lighting moves, ball fixed In the case where the ball is fixed, and the lighting is moved, *we will see the optical flow on the ball's surface*. This is because the reflection of the lighting will move on the ball's surface, which means that the pixels on the ball will appear to move. This can be seen in

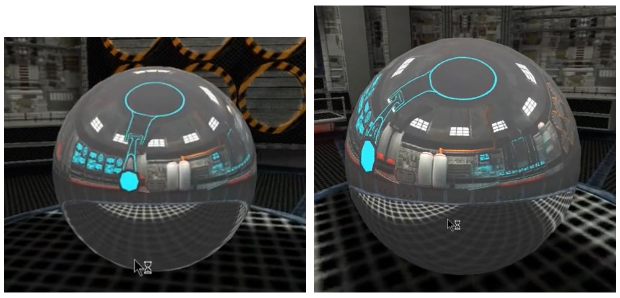

(a) Pose 1 (b) Pose 2

Figure 2: Illumination moving
The two poses of the blue light are different on the ball. This gives a virtue of motion to the texture. Images are from UserBenchmark tests on YouTube, figure a from `2:41` and b from `2:43`.

1.2 Concept Review

Optical flow aims to recover pixel-level motion between two images.
Some references for Optical Flow (not for reviewers)

- YouTube: Optical Flow - Computerphile
- YouTube: Optic Flow Solutions - Computerphile

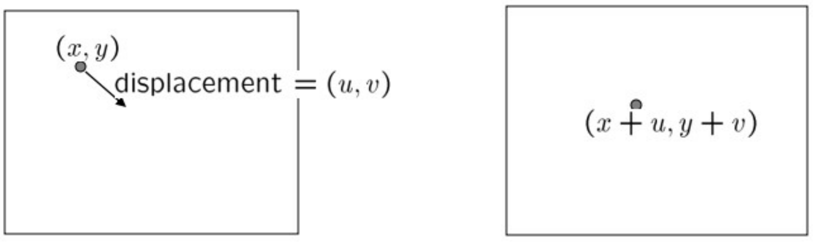
1.2.1 Assumptions of Optical Flow

Some key assumptions in optical flow are as follows

- **Brightness consistency** assumption: The brightness of a particular pixel (after being moved to a new pixel) is the same. That is, $I(x, y, t) = I(x + u, y + v, t + 1)$.
- **Spatial Coherence constraint**: Assume that the neighbors of the pixel being inspected have the same motion. That is, the blob has the same displacement. This allows us to solve the problem as a least-squares problem. This was first introduced in [LK+81].
- It is also assumed that there is no change in environment illumination and only a small motion has occurred between the frames that are being analyzed.

1.2.2 Objective function

The problem of optical flow is to track the motion of pixels in the neighborhood, that maintain color consistency (we assume that they are the same pixels if they have the same color).



(a) t (b) $t + \Delta t$

Figure 3: Pixel positions at different times
The pixel in the left image moves to the right image by ${}^{\Delta x}/_{\Delta t} = u$, ${}^{\Delta y}/_{\Delta t} = v$.

Brightness Consistency Applying the *Brightness consistency* constraint to figure 3, we get

$$I(x + \Delta x, y + \Delta y, t + \Delta t) \approx I(x, y, t)$$

The small motion assumption allows us to apply Taylor series expansion to the above equation

$$\begin{aligned} I(x + u, y + v, t + \Delta t) &= I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \text{higher order terms} \\ &\approx I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \end{aligned} \tag{1}$$

Merging the above two equations, we get

$$\begin{array}{ll} 0 \approx \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t & \frac{\partial I}{\partial x} = I_x \quad \frac{\partial I}{\partial y} = I_y \quad \frac{\partial I}{\partial t} = I_t \\ \Rightarrow 0 = I_x\,\Delta x + I_y\,\Delta y + I_t\Delta t = [I_x \;\; I_y] \cdot [\Delta x \;\; \Delta y] + I_t\Delta t & [I_x \;\; I_y] = \nabla I \\ \Rightarrow \nabla I \cdot [u \;\; v] + I_t = 0 & \end{array}$$

The data terms (obtained through the image) and the spatial terms (algorithm finds these) are marked below

$$\underbrace{[I_x \;\; I_y]}_{\text{data term}} \cdot \underbrace{[u \;\; v]}_{\text{spatial term}} + \underbrace{I_t}_{\text{data term}} = 0 \tag{2}$$

Generalized Implementation As implemented in [LK+81]. Consider F and G as images and $\mathbf{x}$ as pixel position and $\mathbf{h}$ as the shift in position (possibly of a region R). We wist to minimize the L_2 norm error

$$\mathrm{E}_{\mathrm{L2}} = \sum_{\mathbf{x} \in R} [F(\mathbf{x} + \mathbf{h}) - G(\mathbf{x})]^2$$

Using the taylor expansion

$$F(\mathbf{x} + \mathbf{h}) = F(\mathbf{x}) + h_x\frac{\partial F}{\partial x} + h_y\frac{\partial F}{\partial y} = F(\mathbf{x}) + \mathbf{h}\frac{\partial F}{\partial \mathbf{x}}$$

To obtain the $\mathbf{h}$ that minimizes the error E_{L2}, we do the following

$$
\begin{aligned}
0 &= \frac{\partial \mathrm{E}_{\mathrm{L2}}}{\partial \mathbf{h}} \\
&= \frac{\partial}{\partial \mathbf{h}} \sum_{\mathbf{x} \in R} \left[F(\mathbf{x}+\mathbf{h}) - G(\mathbf{x})\right]^2 \\
\frac{\partial E}{\partial h_x} &= \sum_{\mathbf{x} \in R} \left[F(\mathbf{x}) + h_x \left(\frac{\partial F}{\partial x}\right)^2 + h_y \left(\frac{\partial F}{\partial x}\right)\left(\frac{\partial F}{\partial y}\right) - G(\mathbf{x})\left(\frac{\partial F}{\partial x}\right)\right] \\
\frac{\partial E}{\partial h_y} &= \sum_{\mathbf{x} \in R} \left[F(\mathbf{x}) + h_x \left(\frac{\partial F}{\partial x}\right)\left(\frac{\partial F}{\partial y}\right) + h_y \left(\frac{\partial F}{\partial y}\right)^2 - G(\mathbf{x})\left(\frac{\partial F}{\partial y}\right)\right]
\end{aligned}
$$

This gives the final equation

$$
\mathbf{h} = \begin{bmatrix} h_x & h_y \end{bmatrix} = \left[\sum_{\mathbf{x} \in R} \begin{pmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} \end{pmatrix} (G(\mathbf{x}) - F(\mathbf{x}))\right] \begin{bmatrix} \sum_{\mathbf{x} \in R} \left(\frac{\partial F}{\partial x}\right)^2 & \sum_{\mathbf{x} \in R} \left(\frac{\partial F}{\partial x}\right)\left(\frac{\partial F}{\partial y}\right) \\ \sum_{\mathbf{x} \in R} \left(\frac{\partial F}{\partial x}\right)\left(\frac{\partial F}{\partial y}\right) & \sum_{\mathbf{x} \in R} \left(\frac{\partial F}{\partial x}\right)^2 \end{bmatrix}^{-1} \tag{3}
$$

The above equation has the data terms on the right hand side.

1.2.3 Taylor series approximation

It is assumed that the motion is small. This means that the second derivative will almost have no value throughout the image. Therefore, the displacement is assumed to be linearly approximated to the first derivative, in the Taylor expansion.

The first order approximation also makes the differentiation task easier, as seen when minimizing E_{L2} with respect to $\mathbf{h}$.

Second order derivatives will involve calculating Laplacians and Hessians (which are more expensive).

1.2.4 Ill-posed constraint

The constraint of brightness consistency (borrowed from [HS81]) is given by

$$
I_x u + I_y v + I_t = 0 \Rightarrow (I_x, I_y) \cdot (u, v) = -I_t \tag{4}
$$

In such a case, optical flow perpendicular to the image gradient I_x, I_y cannot be uniquely distinguished. This is demonstrated in the figure below.

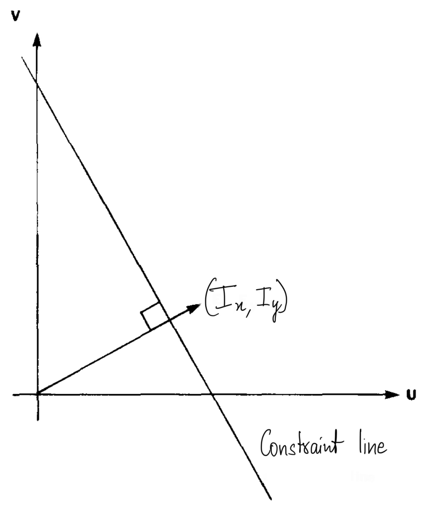


Figure 4: Ill-posed optical flow constraint

References

[HS81] Berthold KP Horn and Brian G Schunck. “Determining optical flow”. In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203.

[LK+81] Bruce D Lucas, Takeo Kanade, et al. “An iterative image registration technique with an application to stereo vision”. In: Vancouver. 1981.

[Mei96] Barbara J Meier. “Painterly rendering for animation”. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques.* 1996, pp. 477–484.

[HE04] James Hays and Irfan Essa. “Image and video based painterly animation”. In: *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering.* 2004, pp. 113–120.

[Lin+10] Liang Lin et al. “Painterly animation using video semantics and feature correspondence”. In: *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering.* 2010, pp. 73–80.

[OH11] Peter O’Donovan and Aaron Hertzmann. “Anipaint: Interactive painterly animation from video”. In: *IEEE transactions on visualization and computer graphics* 18.3 (2011), pp. 475–487.