Report 6

Jiachen Tian

Objectives achieved this week

- Explore the optical flow algorithm, precisely Lucas Kanade optical flow.

- Implement Lucas Kanade optical flow from scratch and got it work on sample cases.

Objectives for next week

- Keep adding more functions to Lucas Kanade optical flow.

- Adding a pyramid to the optical flow to ensure precision.

- Parallelize optical flow as much as possible because its a slow algorithm.

Results Demo

Lucas Kanade optical flow is exactly what I am looking for.

pros

-Could be fast after proper parallelization.

-Could be Robust by implementing extra algorithm including Gaussian and Laplacian Pyramids.

-Completely automated.

cons

-Hard to implement.

Theory explanation

-Every image has gradient. After converting it to single color, Gradient could be represented by change of numbers on the image.

-Optical flow assunes image gradient remains constant after it moves to a new location.(as shown below, x -> location x, y -> location y, t -> time, u and v -> moving factor, I() -> gradient factor).

-After step 4, since we have a whole image of vectors but only two variables, we could use overconstrained linear system to solve for the two unknowns as shown at step 5.

-Therefore, U and V(direction vector for each pixel displacement) could be solved.

In [31]:

```
%%latex
\begin{align}
(1)I(x, y, t) = I(x + u, y + v, t + 1)//Consistency \\
(2)I(x+n, y+v,t+1) - I(x,y,t) = t_x*u+I_y*v+I_t \\
(3)I_x*u+I_y*v+I_t = 0 \\
(4)\nabla I[u v]^T + I_t = 0 \\
(5)\begin{bmatrix}
    \sum{I_xI_x} & \sum{I_xI_y}\\
    \sum{I_xI_y} & \sum{I_yI_y}
    \end{bmatrix}
    \begin{bmatrix}
    u \\
    v
    \end{bmatrix}
    =-\begin{bmatrix}
    \sum{I_xI_t}\\
    \sum{I_yI_t}
    \end{bmatrix}
\end{align}
```

$$(1)I(x, y, t) = I(x + u, y + v, t + 1)//Consistency$$

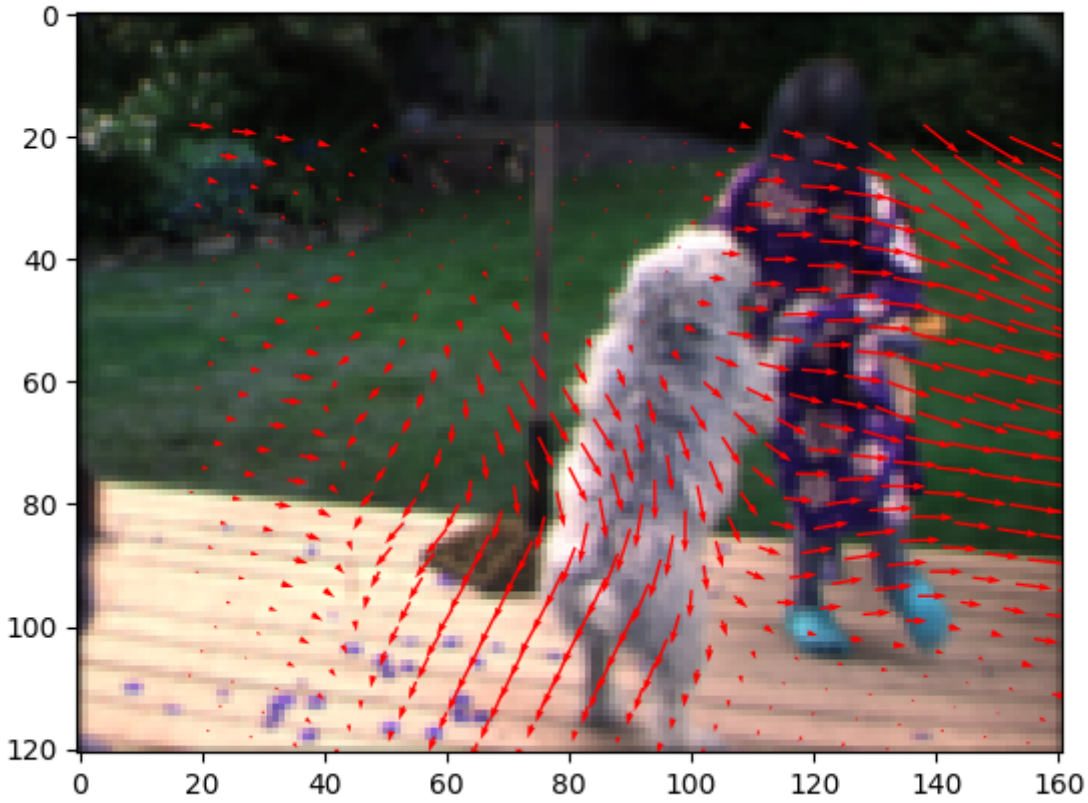
$$(2)I(x + n, y + v, t + 1) - I(x, y, t) = t_x * u + I_y * v + I_t$$

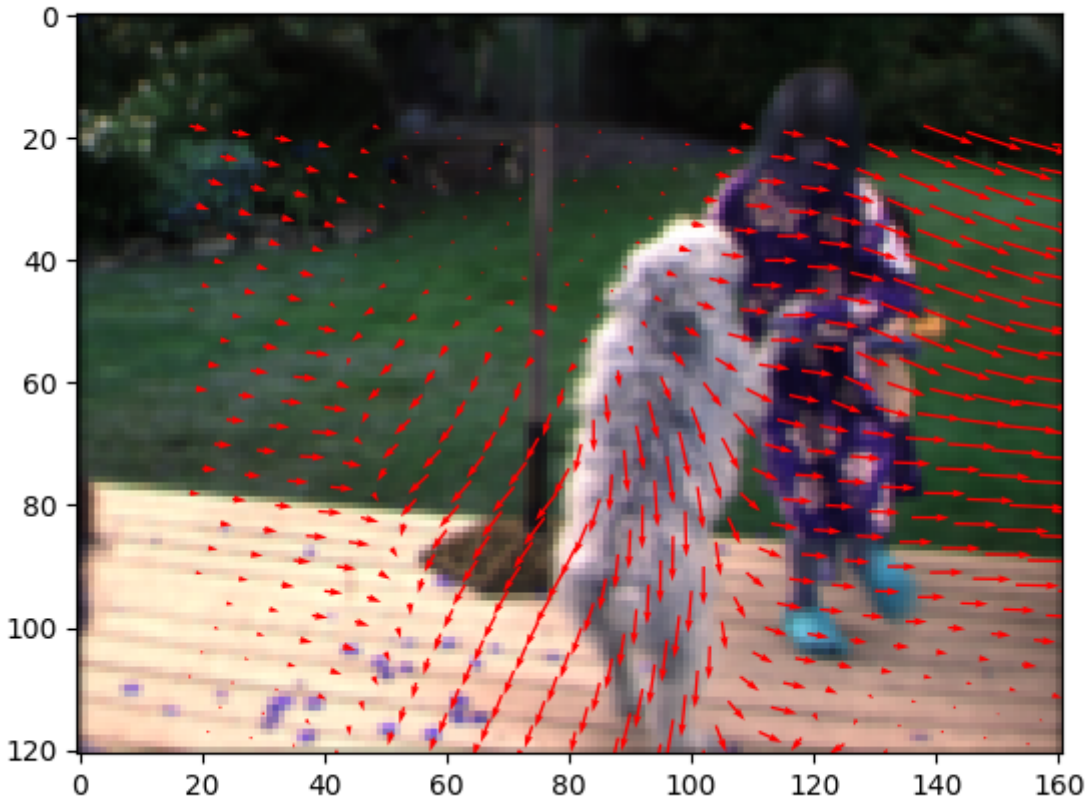
$$(3)I_x * u + I_y * v + I_t = 0$$

$$(4)\nabla I[uv]^T + I_t = 0$$

$$(5)\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Picture Demo(Pay attention to the shift between upper picture and lower picture)





Conclusion

-Now we are officially getting into the fun stuffs. Successful implementation would result in best outcomes possible among all the other algoritums. Failure, on the other hand would be devastating due to its hard-to-debug nature. One last thing to look into if successfully implemented would be hidden markov models, which could further improve precision.