# CS415 HW3

agupt69, kczmrzy2

Due: December 3, 2021

## 1 3D Optical Flow, Lead: Justyna

Derive the gradient constraint equation for optical flow between volumetric images (e.g. CT images) – the image sequence is now $I(x, y, z, t)$.

$I(x, y, z, t)$

$= I(x + dx, y + dy, z + dz, t + dt)$

$= I(x, y, z, t) + \frac{\partial I}{\partial x}(x + dx - x) + + \frac{\partial I}{\partial y}(y + dy - y) + \frac{\partial I}{\partial z}(z + dz - z) + \frac{\partial I}{\partial t}(t + dt - t)$

$0 = I_x dx + I_z dz + I_t dt$

$0 = I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_z \frac{dz}{dt} + I_t$

$-I_t = I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_z \frac{dz}{dt}$

$I_x V_x + I_y V_y + I_z V z = -I_t$

## 2 RGB Optical Flow, Lead: Justyna

Consider the RGB, that is $I(x, t) = [R(x, t), G(x, t), B(x, t)]$. Suggest an analogous gradient-constraint equation for optical flow between RGB images. Explain under what conditions the new equation will have:

1) infinitely-many solutions (recall aperture problem)
2) unique solutions
3) no solution

Usually, optical flow computation is based on grayscale images and the brightness conservation assumption. This is why when describing optical flow in the previous question, we expressed the image intensity $I$ as a function of space $(x, y, z)$ and time $t$.

In this case, we would think of the image intensity $I$ as a function of the 3 RGB color channels and time $t$. RGB contains three color channels: red, green, and blue. The calculation of optical flow for color images with these three channels would result in a system of three equations (where $V_x = \frac{dx}{dt}$):

$$I_{Rx}V_x + I_{Rt} = 0$$
$$I_{Gx}V_x + I_{Gt} = 0$$
$$I_{Bx}V_x + I_{Bt} = 0$$

where $I_R$, $I_G$, and $I_B$ are are the partial derivatives of the image brightness for each channel.

If we were to have a 5x5 window, we'd have 5×5×3=75 equations since we have 3 (where $R = 0, G = 1, B = 2$) color channels.

$$0 = I_t(p_i)[0, 1, 2] + \nabla I(p_i)[0, 1, 2] * [u, v]$$

This system of equations will have infinitely-many solutions if the number of unknowns exceeds the number of equations and it is a homogeneous system.

This system of equations will have a unique solution if they are an independent system. Graphically, this ordered triple defines a point that is the intersection of three channels.

This system of equations will have no solution if they are an inconsistent system. This will happen when there are three equations (due to the 3 color channels of RGB) and two unknowns. This will be like the aperture problem of regular optical flow.

# 3 Tracking using Optical Flow Estimation in Python, Lead: Justyna

## a Good Features to Track for Spare LK

## b Compute Optical Flow – both forward and backward

## c Update features to track using Optical flow estimation in sparse LK algorithm

## d Coarse-to-Fine Optical Flow

Suppose we are given with two square images I1 and I2, explain (not implement) in detail how to do coarse-to-fine (pyramid) optical flow estimation. You may

assume that the image sizes from level 1 to 3 are 100, 50, 25.

First you'd compute lower resolution images of I1 and I2 so that the optical flow constraint equation becomes valid again. This involves building pyramids with 3 levels, and the reduction ratio of 2. The upper or coarse resolution levels (level 3 here) are obtained by averaging the corresponding r x r pixels in the previous level. You'd apply the optical flow algorithm, Lucas-Kanade, to the lowest resolution image (to get a flow field representing the flow from the first frame to the second frame). Then, you'd apply this flow field to warp the first frame toward the second frame. Lastly, you rerun the Lucas-Kanade optical flow algorithm on the new warped image to get a flow field from it to the second frame and repeat until convergence.

### e Findings

State in your report which of the settings performed the best for each of the sequences given using quantitative comparison. You can compute the error map of your estimate using ground truth flow given in the video.

## 4 Background Subtraction in Python with Frame Differencing, Lead: Ayush

Figure 1: Ground Truth.

### a Findings

State in your report which of the algorithms performed the best for each of the sequences given using quantitative comparison. You can compute the error map of your estimate by using high flow pixels in the ground truth flow given in the video as the foreground.

Figure 2: Threshold Tracker.

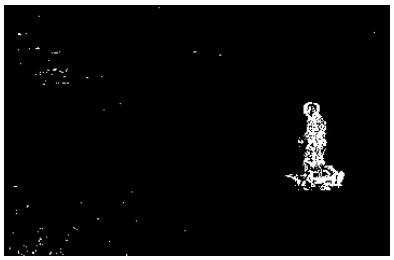

Figure 3: Frame Differencing - Nonadaptive.

Figure 4: Threshold Tracker.

## 5 Abstract/Parameter Gradients, Lead: Ayush

A trivial CNN takes a single, scalar input $x$, has a first layer with kernel $k_1$, bias $b_1$, with the activation function $h(a) = a^2$, and a second layer with kernel $k_2$, bias $b_2$, with a softmax activation function. All kernels and biases are scalar, so the output $\hat{y}$ is a scalar as well. The loss function is the cross entropy, i.e., $l(y, \hat{y}) = y\log\hat{y}$. Compute analytically each of the four components of the gradient of the error $E_1(w)$ for training sample $(x_1, y_1)$ relative to the vector $w = [k_1, b_1, k_2, b_2]^T$ of the neural network's parameters. Your answer should only involve variables mentioned in the question.

Since the input into the softmax function will be a scalar, we can use the sigmoid function. We will have to find four partial derivatives of the following equation with respect to $k_1, b_1, k_2, b_2$:

$$y\, log_e \left( \frac{1}{1 + e^{-\left(k_2(k_1 x + b_1)^2 + b_2\right)}} \right)$$

Therefore, these four partial derivatives (components of the gradient of error) will be:

$$\frac{\partial}{\partial\, k_1} = \frac{2k_2 e^{-b_2 - k_2(k_1 x + b_1)^2} yx\,(k_1 x + b_1)}{1 + e^{-k_2(k_1 x + b_1)^2 - b_2}}$$

$$\frac{\partial}{\partial\, b_1} = \frac{2k_2 e^{-b_2 - k_2(b_1 + k_1 x)^2} y\,(b_1 + k_1 x)}{1 + e^{-k_2(k_1 x + b_1)^2 - b_2}}$$

$$\frac{\partial}{\partial\, k_2} = -\frac{e^{-k_2(k_1 x + b_1)^2 - b_2} y\left(-k_1^2 x^2 - 2k_1 x b_1 - b_1^2\right)}{1 + e^{-k_2(k_1 x + b_1)^2 - b_2}}$$

$$\frac{\partial}{\partial\, b_2} = \frac{e^{-b_2 - k_2(k_1 x + b_1)^2} y}{1 + e^{-k_2(k_1 x + b_1)^2 - b_2}}$$

## 6 Helping Munchkins, Lead: Justyna

Munchkins wants to improve the accuracy of a virus detector using a CNN with $l$ layers and ReLU activation functions. However, Munchkins' boss Munchies wants to use linear activation functions, that is, $\sigma(x) = x$. Show that if the activation functions are linear, then the entire network can be replaced by single layer.

The linear activation function has an equation similar to that of a straight

line i.e. $\phi(v) =$ v'b + $a$. If there are multiple layers, all linear in nature, the final activation function of last layer is a linear function of the input of first layer. Therefore, all the layers can be replaced by a single layer.

The overall network is a combination of function composition. Mathematically, this is demonstrated below for the two layers:

$$g_w(x) = v_2(v_1 b + a_1) + a_2 = \tilde{v}b + \tilde{a}$$

By function composition, we will just get another linear function (i.e. Munchies should listen to Munchkins).