

9 commits

1 branch

0 releases

1 contributor

Branch: master


New pull request

Create new file

Upload files

Find file

Clone or download

pjreddie Merge branch 'master' of <https://github.com/pjreddie/vision-hw3>

Latest commit d354730 on 4 May

data	initial commit	8 months ago
figs	new line image	8 months ago
src	add the actual file	8 months ago
.gitignore	initial commit	8 months ago
Makefile	Update Makefile	8 months ago
README.md	Update README.md	8 months ago
tryflow.py	initial commit	8 months ago
uwimg.py	initial commit	8 months ago

README.md

# CSE 455 Homework 3

Welcome friends,

It's time for optical flow!

To start out this homework, copy over your `process_image.c`, `filter_image.c`, `resize_image.c`, `harris_image.c`, and `panorama_image.c` files from hw2 to the `src` directory in this homework. We will be continuing to build out your image library.

## 1.1 Integral images

Optical flow has to run on video so it needs to be fast! We'll be calculating structure matrices again so we need to do aggregated sums over regions of the image. However, instead of smoothing with a Gaussian filter, we'll use [integral images](#) to simulate smoothing with a box filter.

Fill in `make_integral_image` as described in the wikipedia article. You may not want to use the `get_pixel` methods unless you do bounds checking so you don't run into trouble.

## 1.2 Smoothing using integral images

We can use our integral image to quickly calculate the sum of regions of an image. Fill in `box_filter_image` so that every pixel in the output is the average of pixels in a given window size.

## 2. Lucas-Kanade optical flow

We'll be implementing [Lucas-Kanade](#) optical flow. We'll use a structure matrix but this time with temporal information as well. The equation we'll use is:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix}$$

## 2.1 Time-structure matrix

---

We'll need spatial and temporal gradient information for the flow equations. Calculate a time-structure matrix. Spatial gradients can be calculated as normal. The time gradient can be calculated as the difference between the previous image and the next image in a sequence. Fill in `time_structure_matrix`.

## 2.2 Calculating velocity from the structure matrix

---

Fill in `velocity_image` to use the equation to calculate the velocity of each pixel in the x and y direction. For each pixel, fill in the `matrix M`, invert it, and use it to calculate the velocity.

Try calculating the optical flow between two dog images:

```
a = load_image("data/dog_a.jpg")
b = load_image("data/dog_b.jpg")
flow = optical_flow_images(b, a, 15, 8)
draw_flow(a, flow, 8)
save_image(a, "lines")
```

It may look something like:

## 3. Optical flow demo using OpenCV

---

Using OpenCV we can get images from the webcam and display the results in real-time. Try installing OpenCV and enabling OpenCV compilation in the Makefile (set `OPENCV=1`). Then run:

```
optical_flow_webcam(15,4,8)
```

## 4. Turn it in

---

Turn in your `flow_image.c` on canvas under Assignment 3.