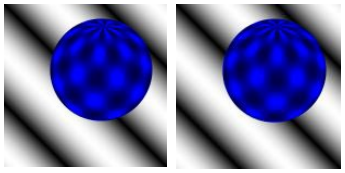
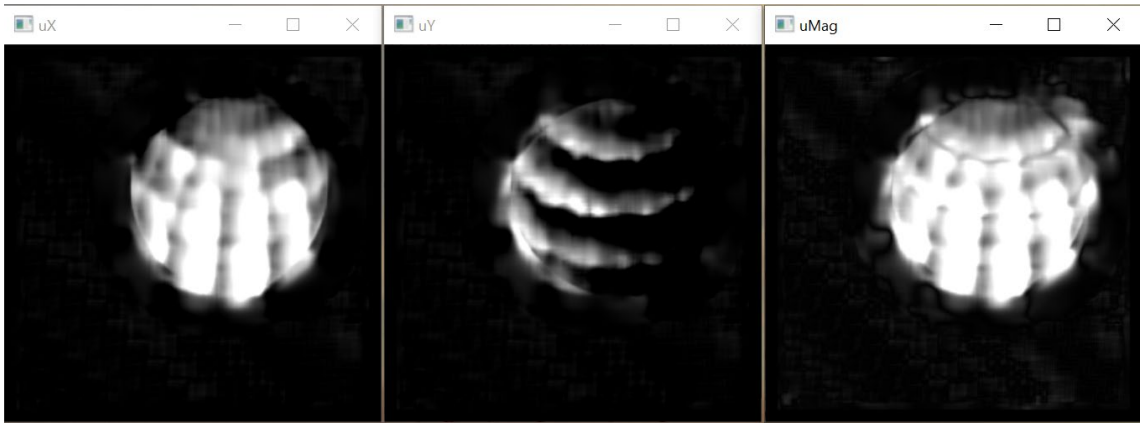
Programming Assignment - Optical Flow

The goal of this assignment is to visualize the motion between two images.

Part 1 – Optical Flow Calculation (60 points):
Write a program that loads a pair of images with a small amount of motion in between them. Implement the Lucas-Kanade algorithm covered in class to compute the optical flow relating the image pair. You're encouraged to build on your Harris Corner assignment. Start with one of the sphere image pair.

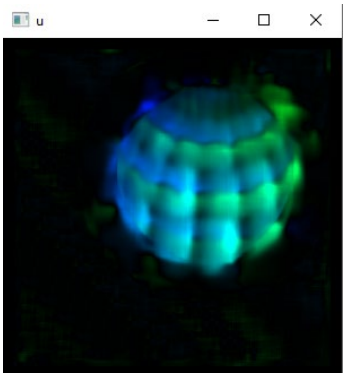


You must implement all calculations related to Lucas-Kanade yourself (i.e. getting image gradients, solving for u and v, getting the matrix determinant, vector magnitude) yourself. You may not use any OpenCV or Numpy built-in functions to complete these steps.



The results shown above were computed with a 21x21 pixel neighborhood.

Part 2 – Visualization (20 points):
Color-code the flow direction using two color channels. Use atan2() to compute the angle of the flow at a particular pixel based on the flow values in the x and y directions. atan2 returns an angle between -pi and pi. Map the angle to values in two color channels using interpolation. For instance, assign an angle of -pi to 0 in the blue channel and 255 in the green channel. Then assign an angle of pi to 255 in the blue channel and 0 in the green channel. Interpolate all values in between. Multiply the computed color value by the magnitude of the flow vector so that areas with little motion will be closer to black and areas with a lot of motion will be brighter.



Part 3 – Data (5 points):
Run your code on the four pairs of sample data included with the assignment (2dShapes, sphere, tree, house). Additionally, take a pair of photos moving the camera a small amount in between the two. Run your code on this image pair.

Part 4 – Report and Submission (15 points):
Submit your source code and a PDF of a report. The report should include a description of the assignment and of the Lucas-Kanade optical flow algorithm, equations used, images of your results (on 5+ pairs), and any issues encountered. Final result should be the color-coded flow direction image and the "arrowedLined" direction image. For at least one image pair, include images of the following intermediate stages:

- Gradient for the x-direction
- Gradient for the y-direction
- Temporal gradient
- (Temporal gradient)*(Gradient x-direction)
- (Temporal gradient)*(Gradient y-direction)
- Flow in the x-direction (u value)
- Flow in the y-direction (v value)
- Magnitude of flow: sqrt(flowX*flowX + flowY*flowY)

Include observations on how the algorithm performs on different types of data. Embed the images in your file. Each section should be labeled and images should be referenced with figure numbers. Do not upload separate image files.

The neatly and clearly written portion of your report with a title, complete sentences, organized paragraphs, and references to figures should be at least 3 paragraphs.

Allowable Libraries and Existing Functions
You may only use the functions listed below from the libraries listed below. All other code in your submission must be original code you write yourself.

- resize (OpenCV) – resize an image (shrink to speed up testing, enlarge to see details)
- cvtColor (OpenCV) – convert color image to grayscale (immediately after loading in image)
- imshow (OpenCV) – display an image
- imwrite (OpenCV) – save an image
- zeros (numpy) – make empty image
- range (Python built-in)
- float (Python built-in)
- sum (Python built-in)
- atan2 (Python – math)
- pi (Python – math)