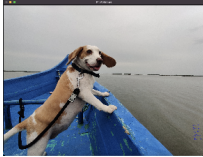
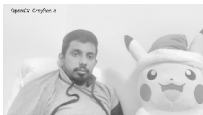
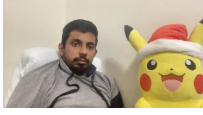
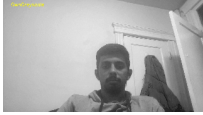

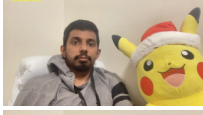
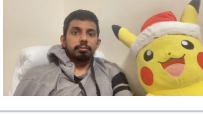
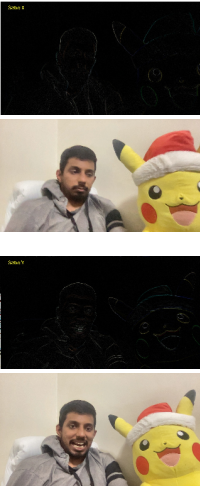
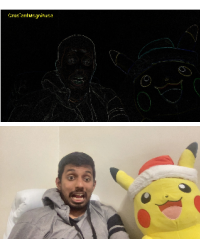





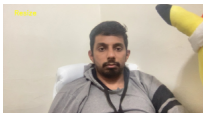
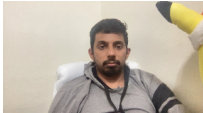
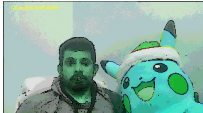



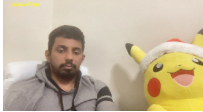
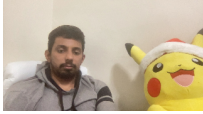
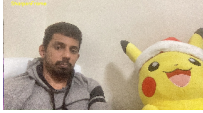
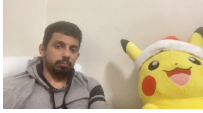

# Project 1 : Real-time filtering

This project begins with setting up OpenCV packages and IDE for Mac and then works mostly with manipulating Images and Frames from the webcam and storing these at the user's request.

The main aim of the project was to understand how pixels are being accessed, how to manipulate them and to design real-time filters from scratch. The driver application uses the web camera to fetch frames and applies filters on them based on the user's input. The below table displays the task, description of it, and thumbnail of the output( Click on the output to enlarge the size ).

Tasks and Extensions	Explanation	Link
<b>Task 1:</b> Read an image from a file and display it	Reading an image from disk and showing it till the user presses 'q'	
<b>Task 2:</b> Display live video	<ul style="list-style-type: none"><li>Opens a video channel, creates a window, and keeps updating the window with the current frame(live stream).</li><li>On 's' key, saves the current frame as image and on pressing 'q' the program quits.</li></ul>	<a href="#">Sample Live Video</a>
<b>Task 3:</b> Display greyscale live video	<ul style="list-style-type: none"><li>OpenCV cvtColor function is used to convert color frames to greyscale frames in the live feed.</li><li>Conversion from RGB to Greyscale, the G(0.587) is weighed more because Green is more to the human eye, followed by Red(0.299) and the least being Blue(0.114).</li><li><math>Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B</math></li></ul>	 
<b>Task 4:</b> Display alternative greyscale live video	<ul style="list-style-type: none"><li>Implemented my own greyscale filter by assigning the Blue channel value to the rest of the channel.</li><li>Learned how to manipulate pixels in a frame.</li></ul>	 
<b>Task 5:</b> Implement a 5x5 Gaussian filter as separable 1x5 filters	<ul style="list-style-type: none"><li>Implemented Gaussian Blur Filter using {1,2,4,2,1}. Implemented the filter using the 2-pass mechanism first, convolved horizontally and then vertically.</li></ul>	 

<p><b>Task 6:</b> Implement a 3x3 Sobel X and 3x3 Sobel Y filter as separable 1x3 filters</p>	<ul style="list-style-type: none"> <li>Sobel X and Sobel Y filters are used to get the horizontal and vertical edges respectively.</li> <li>Converted the video stream to CV_16SC3 to cover range [-255, 255].</li> <li>Split the filter matrix 3*3 to 3*1 and 1*3 and convolved the matrix separately to get the results.</li> <li>Converted the result to CV_8UC3 using convertScaleAbs as imshow() doesn't support CV_16SC3.</li> </ul>	 <p>The image shows a sequence of four frames. The top frame is a dark image with white edges detected by the Sobel X and Y filters. The bottom three frames show a man and a Pikachu plush toy, with the top frame being the original image and the bottom two being the edge detection results.</p>
<p><b>Task 7:</b> Implement function to generate gradient magnitude image from the X and Y Sobel images</p>	<ul style="list-style-type: none"> <li>Generates gradient magnitude using Euclidean Distance.</li> <li>This combines the horizontal and vertical edges and shows all the edges and can be used for edge detection.</li> </ul>	 <p>The image shows a sequence of four frames. The top frame is a dark image with white edges detected by the gradient magnitude function. The bottom three frames show a man and a Pikachu plush toy, with the top frame being the original image and the bottom two being the gradient magnitude results.</p>
<p><b>Task 8:</b> Implement a function that blurs and quantizes a color image</p>	<ul style="list-style-type: none"> <li>Blur filter applied, based on the levels bucket is created using <math>b=255/levels</math></li> <li>Quantized the value per pixel per color by <math>xt = x / b</math>, then execute <math>xf = xt * b</math>, <math>xf</math> is stored in the destination frame.</li> </ul>	 <p>The image shows a sequence of four frames. The top frame is a blurred and quantized version of the original image. The bottom three frames show a man and a Pikachu plush toy, with the top frame being the original image and the bottom two being the blurred and quantized results.</p>
<p><b>Task 9:</b> Implement a live video cartoonization function with gradient magnitude &amp; blur/quantize</p>	<ul style="list-style-type: none"> <li>Gradient magnitude is followed by blur and quantizing the source image (Temp).</li> <li>In Temp image the pixels with gradient magnitude greater than a threshold are set to black.</li> </ul>	 <p>The image shows a sequence of four frames. The top frame is a cartoonized version of the original image, where the background is blurred and the foreground is quantized. The bottom three frames show a man and a Pikachu plush toy, with the top frame being the original image and the bottom two being the cartoonized results.</p>
<p><b>Task 10.1:</b> Allow the user to adjust brightness or contrast.</p>	<ul style="list-style-type: none"> <li>The user is allowed to increase or decrease the brightness of the live frames or the loaded image.</li> <li>We are just adding/subtracting one from each pixel and each color.</li> </ul>	 <p>The image shows a sequence of four frames. The top frame is a version of the original image with adjusted brightness or contrast. The bottom three frames show a man and a Pikachu plush toy, with the top frame being the original image and the bottom two being the adjusted results.</p>

<b>Task 10.2:</b> Stretch or warp the image	<ul style="list-style-type: none"> <li>Based on the preset values the frames are resized.</li> <li>This can be used when we need the display area to do other parallel works, or to make it more cartoony.</li> </ul>	 									
<b>Task 10.3:</b> Change around the color palette	<ul style="list-style-type: none"> <li>Swapped the colors at each pixel and applied the cartoony effect developed earlier.</li> <li>This can be used to create HULK effect later on.</li> </ul>	 									
<b>Task 10.4:</b> Make the image a negative of itself	<ul style="list-style-type: none"> <li>Inverting dark and white pixels and setting it to extreme maximum values.</li> </ul>	 									
<b>Task 10.5:</b> Implement other types of blur filters	<ul style="list-style-type: none"> <li>Implemented median filter.</li> <li>Took all the neighbouring 9 pixels and sorted them and set the pixel to the median value.</li> </ul>	 									
<b>Extension 1:</b> Implement Additional Filters  <table border="1" data-bbox="602 1245 722 1379"> <tbody> <tr> <td>0</td><td>-1</td><td>0</td></tr> <tr> <td>-1</td><td>5</td><td>-1</td></tr> <tr> <td>0</td><td>-1</td><td>0</td></tr> </tbody> </table>	0	-1	0	-1	5	-1	0	-1	0	<ul style="list-style-type: none"> <li>Implemented sharpness filter.</li> <li>Below filter is used for the implementation</li> </ul>	 
0	-1	0									
-1	5	-1									
0	-1	0									
<b>Extension 2:</b> Implement Additional Effects	<ul style="list-style-type: none"> <li>Implemented flash action hero filter, that shows trailing movement.</li> </ul>	Flash Effect									
<b>Extension 3:</b> Implement your effects for still images and enable the user to save it	<ul style="list-style-type: none"> <li>Implemented all the effects in the Image, except videoStoring and Flash as these both are related to video. Below table contains the LEGEND on how to operate.</li> </ul>										
<b>Extension 4:</b> Let the user save short video sequences with special effects	<ul style="list-style-type: none"> <li>Users can start recording the video by clicking 'v' and performing all the button clicks for various effects and stopping the video with another 'v' click.</li> </ul>	All Effects									
<b>Extension 5:</b> Let the user add captions to images or video sequences when they are saved	<ul style="list-style-type: none"> <li>When saving the image or video, the program asks for the meme text and attaches it to the output.</li> </ul>	Already All the images above have the captions									

KeyPress	vidDisplay	imgDisplay		KeyPress	vidDisplay	imgDisplay
<b>q</b> : Quits Program	Yes	Yes		<b>s</b> : Save Current Frame as Image	Yes	No
<b>s</b> : Save Frame	Yes	Yes		<b>i</b> : Blur and Quantize	Yes	Yes
<b>c</b> : Cartoonization	Yes	Yes		<b>w</b> : Negative Inverted	Yes	Yes
<b>x</b> : Sobel X	Yes	Yes		<b>n</b> : Set back to Normal	Yes	Yes
<b>y</b> : Sobel Y	Yes	Yes		<b>f</b> : Flash Effect	Yes	No
<b>m</b> : Gradient Magnitude	Yes	Yes		<b>a</b> : Sharped Image	Yes	Yes
<b>+</b> : Increase Brightness	Yes	Yes		<b>r</b> : Resize	Yes	Yes
<b>-</b> : Decrease Brightness	Yes	Yes		<b>p</b> : Change Color Palette	Yes	Yes
<b>b</b> : Gaussian Blur	Yes	Yes		<b>m</b> : Median Filter	Yes	Yes
<b>v</b> : Start and Stop Recording	Yes	No				

## Learning Outcome

At the completion of Project 1, the following are my learning outcomes:

- Learned basic OpenCV API's to read and display images and videos.
- Learned about the cv::cvtColor() to convert video color.
- Learned about manipulating pixels in a video stream.
- Learned how to apply separable filters in a video stream.
- Learned about converting image matrix from one type to another to support a greater range of values for a channel, manipulating pixels to detect edges by using separable filters in a video stream.

## Acknowledgment

- [splitMerge](#)
- [Video Read Display](#)
- [Video Capture](#)
- [PutText](#)