




Topic	INVISIBILITY CLOAK	
Class Description	The student uses the camera vision library to create an invisibility cloak.	
Class	PRO C121	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> <li>Learn about image processing techniques such as saturation and segmentation.</li> <li>Write a program to create an invisibility cloak</li> </ul>	
Resources Required	<ul style="list-style-type: none"> <li>Teacher Resources:               <ul style="list-style-type: none"> <li>Laptop with internet connectivity</li> <li>Earphones with mic</li> <li>Notebook and pen</li> <li>Smartphone</li> </ul> </li> <li>Student Resources:               <ul style="list-style-type: none"> <li>Laptop with internet connectivity</li> <li>Earphones with mic</li> <li>Notebook and pen</li> </ul> </li> </ul>	
Class structure	<b>Warm-Up</b> <b>Teacher-Led Activity 1</b> <b>Student-Led Activity 1</b> <b>Wrap-Up</b>	<b>05 mins</b> <b>15 mins</b> <b>20 mins</b> <b>05 mins</b>
Credit & Permissions:	OpenCV by intel	
WARM-UP SESSION - 10 mins		
<div>  </div> <p><b>Teacher Starts Slideshow</b>  <b>Slide 1 to 4</b>          Refer to speaker notes and follow the instructions on each slide.</p>		

Teacher Action	Student Action
<p>Hey &lt;student's name&gt;. How are you? It's great to see you! Are you excited to learn something new today?</p> <p><b>Following are the WARM-UP session deliverables:</b></p> <ul style="list-style-type: none"> <li>• Greet the student.</li> <li>• Revision of previous class activities.</li> <li>• Quizzes.</li> </ul>	<p><b>ESR:</b> Hi, thanks! Yes I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
<p><b>WARM-UP QUIZ</b> Click on In-Class Quiz</p>	
<p><b>Continue WARM-UP Session</b> Slide 5 to 11</p> 	
<p><b>Activity Details</b></p> <p><b>Following are the session deliverables:</b></p> <ul style="list-style-type: none"> <li>• Appreciate the student.</li> <li>• Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.</li> </ul>	
Teacher Action	Student Action
<p>Can you quickly tell me what we did in the previous class?</p> <p>Yes, correct!</p> <p>Today we'll be creating a capstone project. Do you like magic? Which magical power would you like to have?</p> <p>Today we are going to do something magical and that is we are going to make an invisibility cloak that makes us</p>	<p><b>ESR:</b> We created a chatbot by creating the Machine learning model.</p> <p><b>ESR:</b></p>

invisible. Are you excited about it?	The student talks about the magic he/she likes.
<b>ESR:</b> Yes!	
<div style="text-align: center;">  </div>	
<b>Teacher Ends Slideshow</b>	
<b>TEACHER-LED ACTIVITY - 15 mins</b>	
<b>Teacher Initiates Screen Share</b>	
<b><u>ACTIVITY</u></b>	
<ul style="list-style-type: none"> <li>• Use the different methods of the CV2 library to process images</li> <li>• Capture images to create a video with a mask filter</li> </ul>	
Teacher Action	Student Action
So you remember a few classes ago we used the camera of our system?	
Can you tell me what we did using the camera?	<b>ESR:</b> Using the camera we did face detection, object detection and gesture detection to control input devices(mouse & keyboard).
Yes. And can you tell me which library we used for that?	<b>ESR:</b> We used the cv2 library to access the camera.
Great!	

<p>Today we'll also use the same library to make an invisibility cloak. To create this magical experience using an image processing technique called <b>Color Detection</b> and <b>Segmentation</b>.</p>	
<p>In today's class, we will create an invisibility cloak.</p> <p>We have to capture the background first.</p> <p>Then we'll be coding to detect a red-colored cloth by the camera.</p> <p>When a person stands in front of the camera with a red-colored cloth in front of them, it is masked and the background behind the person will be visible.</p> <p>So, it gives us the illusion that the person standing behind the cloth is invisible. We can choose any color, but here we are using red.</p> <p>Our algorithm or the steps will be as follows:</p> <ol style="list-style-type: none"> <li>1. Capture and store the background frame. This will be done for an initial few seconds.</li> <li>2. Detect the <b>red-colored cloth</b> using <b>color Detection</b> and <b>Segmentation</b> algorithms.</li> <li>3. Segment out the red-colored cloth by generating a mask. [ used in code ]</li> <li>4. Generate the final augmented output to create a magical effect. [ video.mp4 ]</li> </ol> <p>Let's start with our code then.</p> <p><b>Note:</b> The teacher downloads the boilerplate code from <a href="#">Teacher Activity 1</a>.</p> <p><i>The teacher explains the boilerplate code to the student.</i></p>	

Let's go through the boilerplate code once. So we'll be using the **OpenCV (cv2)**, **time** and **numpy** libraries in our algorithm. We can see that these are already imported in the boilerplate code.

```
invisibility_cloak.py > ...  
1 import cv2  
2 import time  
3 import numpy as np
```

To save the video as the output, we have used the **VideoWriter\_fourcc**.

**FourCC** is a code used to specify the video codec. Codecs are used to compress data.

**FourCC** code is passed as:

**cv.VideoWriter\_fourcc('M','J','P','G')** or **cv.VideoWriter\_fourcc(\*'MJPG')** for MJPEG.

We want to save our video with the **.avi** extension. So, we have passed the codec as **XVID**.

[Teacher Activity2: FourCC](#)

```
invisibility_cloak.py > ...  
1 import cv2  
2 import time  
3 import numpy as np  
4  
5 #To save the output in a file output.avi  
6 fourcc = cv2.VideoWriter_fourcc(*'XVID')  
7 output_file = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))  
8
```

Then, we started reading the video from the webcam of our system. We have used the **VideoCapture()** function to capture the video.

Then, we have used the **time.sleep(2)** method to add a 2 seconds delay in the execution of the program. This will allow the camera to start.

```
invisibility_cloak.py > ...
1  import cv2
2  import time
3  import numpy as np
4
5  #To save the output in a file output.avi
6  fourcc = cv2.VideoWriter_fourcc(*'XVID')
7  output_file = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))
8
9  #Starting the webcam
10 cap = cv2.VideoCapture(0)
11
12 #Allowing the webcam to start by making the code sleep for 2 seconds
13 time.sleep(2)
14 bg = 0
15
```

We need to have a video that has some seconds dedicated to the background frame so that it could easily save the background image.

So we have to capture the background in the range of 60. We can vary the range but 60 seconds would be enough. We are flipping the background because the camera captures the image inverted.

```
16 #Capturing background for 60 frames
17 for i in range(60):
18     ret, bg = cap.read()
19     #Flipping the background
20     bg = np.flip(bg, axis=1)
21
```

After this, we have a **while** loop which runs until the camera is open.

```
#Reading the captured frame until the camera is open
while (cap.isOpened()):
    ret, img = cap.read()
    if not ret:
        break
    #Flipping the image for consistency
    img = np.flip(img, axis=1)

    #Generating the final output
    final_output = img
    output_file.write(img)
    #Displaying the output to the user
    cv2.imshow("magic", final_output)
    cv2.waitKey(1)
```

Can you tell me what the next portion of code is doing?

```
while (cap.isOpened()):
    ret, img = cap.read()
    if not ret:
        break
    #Flipping the image for consistency
    img = np.flip(img, axis=1)
```

Great! Now, we stop the loop when **ret** is **false**.

**ESR: cap.read()** method gets the next frame from the camera. It returns 2 values:

1. **ret** - returns true if the frame is available. Otherwise, it will be false.
2. **img** - holds an image array.

```
while (cap.isOpened()):  
    ret, img = cap.read()  
    if not ret:  
        break  
    #Flipping the image for consistency  
    img = np.flip(img, axis=1)
```

What do you think the next line of code is doing?

```
while (cap.isOpened()):  
    ret, img = cap.read()  
    if not ret:  
        break  
    #Flipping the image for consistency  
    img = np.flip(img, axis=1)
```

**ESR:** It is flipping the image.

Exactly! Our front camera returns us a mirror image of our face or surroundings. The **flip()** method accepts two arguments - an array and an axis. The **flip()** method reverses the array along the given axis. Here, it flips the image so that we get the proper image instead of the mirror image of it.

The last portion of code stores the fetched img in the **final\_output** variable and writes it to the video file we wanted to create.

Then , we display the output on screen using the **cv2.imshow()** method.



```
#Reading the captured frame until the camera is open
while (cap.isOpened()):
    ret, img = cap.read()
    if not ret:
        break
    #Flipping the image for consistency
    img = np.flip(img, axis=1)

    #Generating the final output
    final_output = img
    output_file.write(img)
    #Displaying the output to the user
    cv2.imshow("magic", final_output)
    cv2.waitKey(1)

cap.release()
out.release()
cv2.destroyAllWindows()
```

At the very end, we release the camera. We use **cv2.destroyAllWindows()** to close all windows opened by cv2.

That sums up our boilerplate code.

*The teacher starts coding after the boilerplate code is explained.*

Now, let's start writing our code,

As we capture frames we are also capturing the colors in those frames.

And we need to convert the images from BGR (Blue Green Red) to HSV (Hue, Saturation, Value). We need to do this so that we can detect the red color more efficiently.

**HSV** color model can be imagined as a cylinder where -

1. **Hue:** This channel encodes color information. **Hue** is measured in degrees from 0 to 360 along the circumference of the base of the cylinder. The color red lies between 0-10 degrees and 170-180 degrees.
2. **Saturation:** This encodes the intensity of color. More saturated means more brightness in your color. The **Radius** of the cylinder represents **saturation**.
3. **Value:** This encodes the brightness of color, Shading and gloss components of an image. The height represents **value**.

```

22 #Reading the captured frame until the camera is open
23 while (cap.isOpened()):
24     ret, img = cap.read()
25     if not ret:
26         break
27     #Flipping the image for consistency
28     img = np.flip(img, axis=1)
29
30     #Converting the color from BGR to HSV
31     hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

```

Now that we have converted the color from **BGR** to **HSV** it will be easier for us now to identify colors.

We have to create masks that will check for the colors in the specified range and then mask it with the background image.

We will create a mask which will detect the color red. We know that the color red lies between 0-10 degrees and 170-180 degrees (hue) in the HSV color model.

We'll be creating 2 different masks which will help us detect the red color in that given range.

We are creating masks to detect the red colour. You can change the colour value depending on the colour you want.

```
23 while (cap.isOpened()):
24     ret, img = cap.read()
25     if not ret:
26         break
27     #Flipping the image for consistency
28     img = np.flip(img, axis=1)
29
30     #Converting the color from BGR to HSV
31     hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
32
33     #Generating mask to detect red colour(values can be changed)
34     lower_red = np.array([0, 120, 50])
35     upper_red = np.array([10, 255, 255])
36     mask_1 = cv2.inRange(hsv, lower_red, upper_red)
37
38     lower_red = np.array([170, 120, 70])
39     upper_red = np.array([180, 255, 255])
40     mask_2 = cv2.inRange(hsv, lower_red, upper_red)
```

Here, we define a variable named **lower\_red** and create an array using the **np.array()** method which holds [0 , 120, 50]. The 3 values in the array represent hue, saturation and value respectively.

Then we define another variable named **upper\_red** and create an array using the **np.array()** method which holds [10 , 255, 255]. Again, the 3 values in the array represent hue, saturation and value respectively.

After that, we use the **cv2.inRange()** method which has three parameters.

**inRange(source\_array, upper\_bound\_array, lower\_bound\_array)**

These parameters are-

- **source\_array**- is the array which we have to compare to check if it is in a certain range.

- **upper\_bound\_array**- is the array consisting of the upper bound.
- **lower\_bound\_array**- is the array consisting of the lower bound.

So, we write -

```
mask_1 = cv2.inRange(hsv, lower_red, upper_red)
```

The **mask1** detects red color in the **hue** range 0-10.

Similarly, we write the code for **mask2** which detects color in the **hue** range 170-180.

We merge these masks into one by writing-

```
mask_1 = mask_1 + mask_2
```

Now, the mask is created.

To check the output we'll use the **cv2.imshow()** method.

```
cv2.imshow("mask_1", mask_1)
cv2.waitKey(1)
```

Let's run this and check the output.

Go to **the command prompt** and traverse the working folder.

Create a **virtual environment** and run the file.



As you can see, the red-colored part of the cloth is masked and displayed as white pixels.

Great!!!

So we have created the mask now, the next step would be to perform the segmentation operation using this mask.

### Teacher Stops Screen Share

So now it's your turn.

Please share your screen with me.

### Teacher Starts Slideshow




#### Slide 12 to 14

Refer to speaker notes and follow the instructions on each slide.

We have one more class challenge for you.  
Can you solve it?

Let's try. I will guide you through it.

<div>  </div> <p>Teacher Ends Slideshow</p>	
STUDENT-LED ACTIVITY - 20 mins	
<ul style="list-style-type: none"> <li>Ask the student to press the ESC key to come back to the panel.</li> <li>Guide the student to start Screen Share.</li> <li>The teacher gets into Full Screen.</li> </ul>	
Student Initiates Screen Share	
<p><b>ACTIVITY</b></p> <ul style="list-style-type: none"> <li>To perform segmentation on the captured image</li> <li>Displaying final result</li> </ul>	
Teacher Action	Student Action
<p><b>Note:</b> Guide the student as per the following steps:</p> <ol style="list-style-type: none"> <li>Open <a href="#">Student Activity1</a>, download the code in a folder.</li> <li>Go to the command prompt and traverse the folder.</li> <li>Create a virtual environment and run the file.</li> <li>Now you can see the output of <b>mask_1</b>.</li> </ol> <p>Let's start writing the code for segmentation and final output.</p> <ol style="list-style-type: none"> <li>We have the <b>mask1</b> now which holds the red area of the image.            We will do some morphological transformation on the image now. Morphological means relating to the form or shape.            Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images.</li> </ol> <p>You can read more about morphological transformations <a href="#">here</a>.</p>	

2. We will use the **morphologyEx()** method for our morphological transformation.

This method expects 3 arguments-

- The **first argument** is the image we want to apply the morphological operation to. In this case, the image is stored in variable **mask\_1**.
- The **second argument** is the actual type of morphological operation. We will use **cv2.MORPH\_OPEN**. It is useful for removing noise (random variation of brightness) from the image.



- The last required argument is the kernel/structuring element that we are using. We are passing it as-

```
np.ones((3, 3), np.uint8)
```

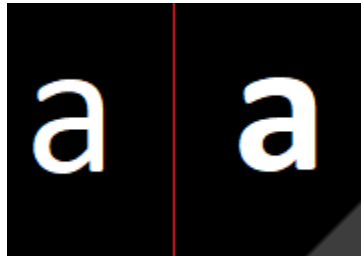
Here, **np.ones((3,3), np.uint8)** returns an array of given shape and type, with ones. In this case, it will return an array of shape 3x3 which will consist of 1's.

```
48 mask_1 = cv2.morphologyEx(mask_1, cv2.MORPH_OPEN, np.ones((3, 3), np.uint8))
```

This line of code will remove noise from our image.

3. Similarly, we can also dilate the image to make it more prominent. We will use the same **cv2.morphologyEx()** method. We will change the second parameter to **cv2.MORPH\_DILATE**.

Example of dilation:



```
mask_1 = cv2.morphologyEx(mask_1, cv2.MORPH_DILATE, np.ones((3, 3), np.uint8))
```

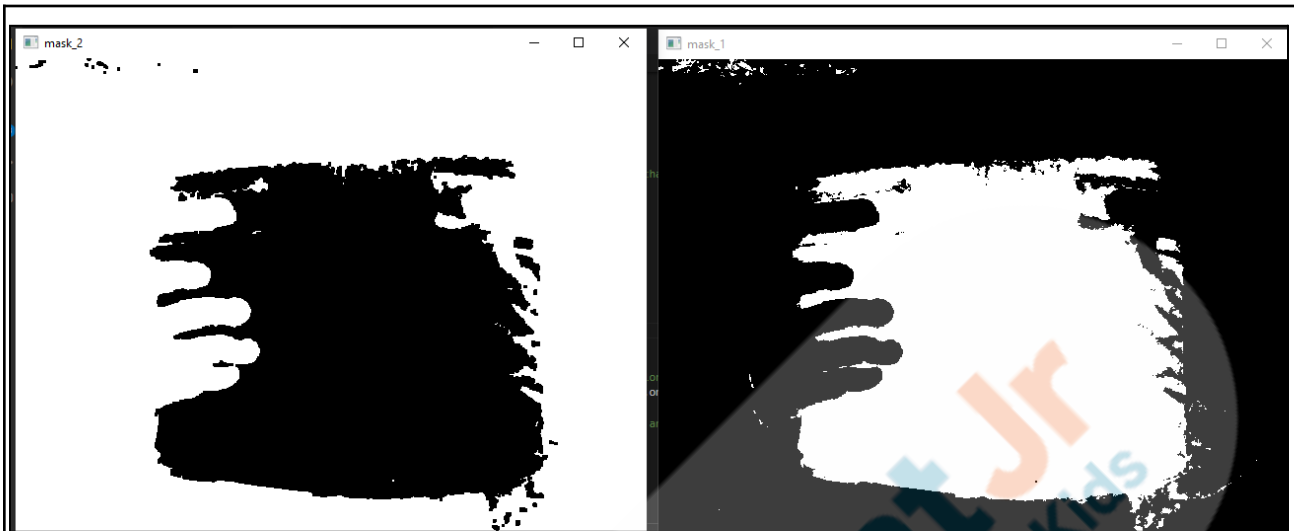
4. Next step is to create an inverted mask to segment out the red color from the frame.

Use the **bitwise\_not** function of OpenCV. The **bitwise\_not** function flips pixel values. All pixels that are greater than zero are set to zero, and all pixels that are equal to zero are set to 255.

```
50
51     #Selecting only the part that does not have mask one and saving in mask 2
52     mask_2 = cv2.bitwise_not(mask_1)
```

If we display both **mask\_1** and **mask\_2** too using **cv2.imshow()** method, we will get the following output.





5. Now, **mask\_2** holds the part of the image which is not red. This portion of the image which is not red should stay the same as the image that we have captured in the **img** variable.

Segment the red color part out of the frame using **bitwise\_and** with the inverted mask.

Here, we will pass three arguments through the **cv2.bitwise\_and()** method.

### Syntax:

**cv2.bitwise\_and(src1, src2, mask)**

Let's understand each argument-

- **src1**: the first image
- **src2**: the second image
- **mask**: This holds the rules to merge. If a region of image (which is gray-scaled, and then masked) has black color (valued as 0), then it is not combined (merging region of the first image with that of the second one). If a region of image has white color (valued as 255), then it is combined (merging region of the first image with that of the second one).

```

53
54     #Keeping only the part of the images without the red color
55     #(or any other color you may choose)
56     res_1 = cv2.bitwise_and(img, img, mask=mask_2)

```

Here, **img** is passed as both **src1** and **src2**. So, this line of code basically replaces the white region of the **mask\_2** with the **img**.

If we display **res\_1** using **cv2.imshow()** method, we will get an output where everything will be visible except the red area in the image.

- Similarly, we will write another line of code which will create an image showing static background frame pixels only for the masked region.

```

58     #Keeping only the part of the images with the red color
59     res_2 = cv2.bitwise_and(bg, bg, mask=mask_1)

```

If we display **res\_2** using **cv2.imshow()** method, we will get an output where the red region of the **img** will be replaced by the **bg** image that we have captured initially.

- Next step is to generate the final image by adding the outputs **res\_1** and **res\_2**.

We can use the **cv2.addWeighted()** method to blend both **res\_1** and **res\_2**.

#### Syntax:

**cv2.addWeighted(source1, alpha, source2, beta, gamma)**

Here, the 5 arguments are-

- **source1**- first image
- **alpha**- visibility of the first image. The value can range between 0-1.
- **source2**- second image
- **beta**- visibility of the second image. The value can range between 0-1.
- **gamma**-The gamma value will be added to the final value and help in processing and

blending the image. In this case, we will keep it as 0.

```
#Generating the final output by merging res_1 and res_2
final_output = cv2.addWeighted(res_1, 1, res_2, 1, 0)
```

8. Create a file **output\_file**. Use the **write()** method to keep on adding the video frames into it.
9. Use the **imshow()** method to display the output. Now, use the **waitKey()** method.

Can you tell me why the **waitKey()** method is used?

Great!!

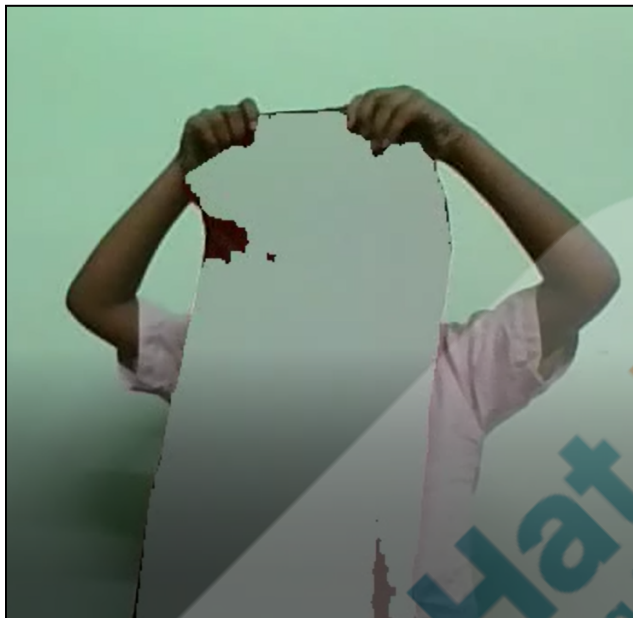
So the compiler will execute the next instruction after the specified time in the **waitKey()** method; meanwhile any key is pressed and it returns the ASCII value of that key.

**ESR:** Yes. it is used to make the compiler wait for the time specified in it.

```
61 #Generating the final output by merging res 1 and res 2
62 final_output = cv2.addWeighted(res_1, 1, res_2, 1, 0)
63 output_file.write(final_output)
64 #Displaying the output to the user
65 cv2.imshow("magic", final_output)
66 cv2.waitKey(1)
67
```

10. Save this and run the file.

11. First, let it capture the background for **1 min or 60 seconds**. Next, you can use a **red-colored cloth** as your invisibility cloak. When you use red cloth in front of the camera, you can see the background is displayed.



Great work!!

So you can see you have created an invisibility cloak today. You can try this with other colors such as green or blue and see the result.

**Teacher Guides Student to Stop Screen Share**

**WRAP-UP SESSION - 05 mins**

**Teacher Starts Slideshow**  
**Slide 15 to 20**



### Activity details

**Following are the WRAP-UP session deliverables:**

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

**WRAP-UP QUIZ**

Click on In-Class Quiz



## Teacher Starts Slideshow Slide 21 to 26

### Activity Details

#### Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

#### FEEDBACK

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

#### Teacher Action

You can also show this to your friends and amaze them.  
We'll be exploring more such things in our upcoming classes.

Are you excited for that?

In the next class, we'll be integrating our chatbot into the Digi-Diary we have created before.

#### Student Action

**ESR: Yes**

*Make sure you have given at least 2 hats-off during the class for:*

Creatively Solved Activities  +10

Great Question  +10

Strong Concentration  +10

### PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks

✕ End Class

### ACTIVITY LINKS

Activity Name	Description	Links
Teacher Activity 1	Teacher Boilerplate Code	<a href="https://github.com/procodingclass/-PRO-C121-Teacher-Boilerplate">https://github.com/procodingclass/-PRO-C121-Teacher-Boilerplate</a>
Teacher Activity 2	FourCC	<a href="https://www.fourcc.org/">https://www.fourcc.org/</a>
Teacher Reference 1	Reference Code	<a href="https://github.com/procodingclass/PRO-C121-Reference-Code">https://github.com/procodingclass/PRO-C121-Reference-Code</a>
Teacher Reference 2	Project	<a href="https://s3-whjr-curriculum-uploads.whjr.online/c6634fd6-4089-4ae8-b94e-c1deeb98a59d.pdf">https://s3-whjr-curriculum-uploads.whjr.online/c6634fd6-4089-4ae8-b94e-c1deeb98a59d.pdf</a>
Teacher Reference 3	Project Solution	<a href="https://github.com/procodingclass/PRO-C121-Project-Solution.git">https://github.com/procodingclass/PRO-C121-Project-Solution.git</a>
Teacher Reference 4	Visual-Aid	<a href="https://s3-whjr-curriculum-uploads.whjr.online/98afeda5-e321-4354-9333-ffdc9dd53028.html">https://s3-whjr-curriculum-uploads.whjr.online/98afeda5-e321-4354-9333-ffdc9dd53028.html</a>
Teacher Reference 5	In-Class Quiz	<a href="https://s3-whjr-curriculum-uploads.whjr.online/93515a03-14e9-4fc4-b50e-d25c1e800a0a.pdf">https://s3-whjr-curriculum-uploads.whjr.online/93515a03-14e9-4fc4-b50e-d25c1e800a0a.pdf</a>
Student Activity 1	Boilerplate Code	<a href="https://github.com/procodingclass/PRO-C121-Boilerplate-Code">https://github.com/procodingclass/PRO-C121-Boilerplate-Code</a>
Student Activity 2	Read more about Morphological Transformations	<a href="https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html">https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html</a>