

AlgoFace Makeup AR-tist Web SDK - Developer Manual

1. Introduction

AlgoFace Makeup AR-tist allows users (shoppers) to virtually try on color cosmetics from their personal devices or in-store without unboxing any physical products. Makeup AR-tist empowers your users to visualize themselves in your products without ever leaving your website, social media channels or apps.

Makeup AR-tist Web SDK contains essential client-side software required to integrate Makeup AR-tist into you own web application. Entire software runs on the browser itself except when it has to validate the license. The SDK also includes two sample applications in plain HTML and JavaScript that demonstrate how to use the SDK.

[Try our live demo here](#)

2. To get started, follow these basic steps

1. Download the zip file and extract the contents to your desired location on disk
2. `cd` into the SDK folder.
3. Run a simple HTTP server. [Learn how to run a simple HTTP server here](#).
4. Open a browser, in the address bar type `http://localhost:<YOUR_PORT>` .

The page requires 3-5 seconds to load completely, as it needs to download model files into memory. Once it is loaded you are free to try our some of our examples presented.

3. SDK file structure and summary

The SDK contains,

- **examples (folder)**

- Contains various examples to help developers to start with.

- `test-camera/index.html`
 - allows developer to test our camera access implementation, developer is free to use their own implementation, but be sure to check out how live try-on implementation uses our camera access implementation.
 - `tryon-live/index.html`
 - allows developers to test live try-on implementation, which access live web cam feed from primary device camera. We also presented developer with few makeup examples for demonstration although our SDK is not limited to these few examples, check out our `VmtHelper` documentation in section 5.3.
 - `tryon-upload-img/index.html`
 - allows developers to test try-on implementation on a uploaded or pre-loaded image. It is important to note that try-on on live feed and static image have slightly different implementations. check out section 4.5 that discusses this difference in detail.

- **helpers (folder)**

- These helper files will allow developers to keep the code clean, modular, while hiding all complexities.

- `camera-controller.js`
 - allows the application to control the web camera feed, includes implementations for starting the camera, stopping the camera, pausing the feed and playing the feed. A callback can be presented to this object using `onEachFrame(cb)` call, which will be called on each frame. We use this callback to call our try-on's detect and render routines.
 - `identify-ua.js`
 - Utility script that identifies user agent information such as operating system, browser, locale etc.
 - `post-vmt.js`
 - This file is loaded after `vmt.js`, as it contains memory management calls, which allocate and free web assembly memory for objects like strings and arrays. The developer may never have to use these calls if they are using our wrapper object `VmtHelper` which automatically handles memory management.
 - `pre-vmt.js`
 - This file is loaded before `vmt.js`, in this file we will implement some methods that will override implementations in `vmt.js`. more details will be discussed in section 4.2.
 - `vmt-helper.js`

- This file is crucial for any type of new application development or integration. This file should be loaded after `post-vmt.js`, it loads a object called `VmtHelper` which is required to implement live try-on or uploaded image try-on correctly. We will dive deep into the API and usage in sections 5.3.
- **vmt (folder)**
 - This is our core software, do not modify or delete any of these files
 - `vmt.data`
 - Contains static assets such as masks, mask address mapping etc.
 - Developer may choose to store this file in a CDN and provide path in `pre-vmt.js`
 - `vmt.js`
 - This file should be loaded before `post-vmt.js` and after `pre-vmt.js`.
 - Loads/fetched `vmt.wasm` and `vmt.data` from the specified locations.
 - `vmt.wasm`
 - This is core software binary file.
 - Developer may choose to store this file in a CDN and provide path in `pre-vmt.js`
- `index.html`
 - index page for demonstration.

4. Integrating Makeup AR-tist in your application

This section will discuss the process and best practices you will need to integrate AlgoFace Makeup AR-tist in you own application.

4.1 Importing SDK into your application

To import SDK into your application, use `<script>` tags in `index.html` and load `pre-vmt.js`, `vmt.js`, `post-vmt.js` and `vmt-helper.js` in the same order.

```
<script src="/helpers/pre-vmt.js"></script>
<script src="/vmt/vmt.js"></script>
<script src="/helpers/post-vmt.js"></script>
<script src="/helpers/vmt-helper.js"></script>
```

If you are implementing try-on with live camera feed you may want to import additional java script files into your application, such as

```
<script src="/helpers/identify-ua.js"></script>
<script src="/helpers/camera-controller.js"></script>
```

The above scripts tags will work assuming that you have copied both `helpers` and `vmt` folders into your application root or where file `index.html` is located.

4.2 Configuring Web Assembly

In this step will be pre-configure the behavior of web assembly (`vmt.wasm`) to certain events. We can do this by modifying `pre-vmt.js` file. We recommend you to pre-configure or verify at least the following events.

-Standard web assembly output

```
var vmt_module = {
  "print" : (text) => console.log(`[MYCUSTOMINFO] ${text}`),
  "printErr" : (text) => console.error(`[MYCUSTOMERR] ${text}`),
  .
  .
  .
  .
};
```

-Locating files on a CDN or a remote server

```
var vmt_module = {
  .
  .
  "locateFile" : (path, prefix) => {

    if(path.endsWith("vmt.data")) return `https://mycdn.com/vmt.data`;
    if(path.endsWith("vmt.wasm")) return `https://mycdn.com/vmt.wasm`;
    return prefix + path;
  },
  .
  .
  .
};
```

You can learn more about the pre-configuring the behavior of `vmt_module` using [this](#) link. Note that when the link refers to object `Module` it is actually referring to `vmt_module` in this SDK's context.

4.3 Implementing Live Try-on

In the HTML body include two canvases as follows

```

<!-- BASE CANVAS -->
<canvas style="display: none;" id="tryon-live-canvas-0"></canvas>
<!-- DISPLAY CANVAS -->
<canvas style="display: block;" id="tryon-live-canvas-1"></canvas>

```

Base canvas is hidden and will be used by `CameraController` to paint live camera feed on it. Which will later be used by `VmtHelper` to read image data from.

Where as display canvas is which `VmtHelper` paints output image after processing image data from base canvas.

After you load the import scripts, and define the HTML body of your application you import another script or include `<script>` tag, inside where you will implement live try-on.

You should implement live try-on only after `vmt.js` has completely loaded and ready to process image frames. When this happens an event called `vmt_is_ready` is dispatched onto the document.

We start our live try-on implementation after this event. First we register our input and output canvases with `VmtHelper`, then indicate it that we want to process continuous stream of image frames using `setLive(true)` call. Now, tell the camera controller where to dump live camera feed once the camera starts, then use `onEachFrame(cb)` function to register a callback that will be triggered every time a new frame is dumped on base canvas. In this call back you shall call `cycleForLive()`, which will process and dumps the output on display canvas automatically. After the above data registrations, you start the camera.

```

/** wait for vmt.js to load completey */
document.addEventListener('vmt_is_ready',()=>{

    /** register a base canvas that is hidden, and your display canvas
     * that shows final output to user
     * in our live-tryon.html example
     * tryon-live-canvas-0 is basde canvas
     * tryon-live-canvas-1 is display canvas */
    var success = window.VmtHelper.registerCanvases(
        "tryon-live-canvas-0",
        "tryon-live-canvas-1"
    );

    if(success){

        /** tell VmtHelper if we are dealing with live camera feed or a static
         * single image */
        window.VmtHelper.setLive(true);
    }
}

```

```

    /** tell the camera controller were to dump the live camera feed
    * ideally we shall dump it on base canvas and on every frame image
    * we will get image from base canvas process it and paint the final
    * out put on display canvas for users to see */
    window.CameraController.streamToCanvasId("tryon-live-canvas-0");

    /** tell camera controller what to do after painting a new frame
    * on base canvas though a callback function as parameter */
    window.CameraController.onEachFrame(=>{
        /** processed frame from base canvas and paints output on display ca
        * since VmtHelper already knwos which canvas to pick from and
        * paint to from line 9 in this snippet */
        window.VmtHelper.cycleForLive();
    });

    /** start the camera */
    window.CameraController.start();
}
});

```

4.4 Implementing Upload Image Try-on

In the HTML body include two canvases as follows

```

<!-- BASE CANVAS -->
<canvas style="display: none;" id="tryon-upload-img-canvas-0"></canvas>
<!-- DISPLAY CANVAS -->
<canvas style="display: block;" id="tryon-upload-img-canvas-1"></canvas>

```

Base canvas is hidden and will be used to paint uploaded image. Which will later be used by `VmtHelper` to read image data from.

Where as display canvas is which `VmtHelper` paints output image after processing image data from base canvas.

After you load the import scripts, and define the HTML body of your application you import another script or include `<script>` tag, inside where you will implement upload image try-on.

You should implement upload image try-on only after `vmt.js` has completely loaded and ready to process image frames. When this happens an event called `vmt_is_ready` is dispatched onto the document.

We start our upload image try-on implementation after this event. Notice that we have a separate `onchange` event which may load before `vmt_is_ready` event occurs, this approach

was purposefully taken in this particular implementation because, if the `onchange` event is defined after `vmt_is_ready` we may end up having a stale upload button on the DOM with does nothing. For better user experience, this decision is made. Ideally a page would load only after the `vmt_is_ready` event.

First we register our input and output canvases with `VmtHelper`, then indicate it that we want to process single image using `setLive(false)` call. Now, when an image is uploaded, we draw that image on the base canvas using following generic image upload routine. After the image is loaded we shall call `cycleForSingleImage()`, which will process and dumps the output on display canvas automatically.

```
/** onchange event implementation of file input */
document.getElementById("tryon-upload-img-input-file").onchange = (evt) => {

    /** generic uploaded file - read routine
     * developer should add filters to allow only certain types of image fi
     * files within a certain resolution and files within a certain size */
    var reader = new FileReader();
    reader.onload = (evt1)=>{
        var img = new Image();
        img.onload = ()=>{
            /** draw the image on base canvas */
            var base_canvas = document.getElementById("tryon-upload-img-canvas-0");
            var base_ctx = base_canvas.getContext("2d");
            base_canvas.width = img.width;
            base_canvas.height = img.height;
            base_ctx.drawImage(img, 0, 0);

            /** call the following function to process the
             * image and paint the output to display canvas */
            window.VmtHelper.cycleForSingleImage();
        }
        img.src = evt1.target.result;
    }
    reader.readAsDataURL(evt.target.files[0]);
};

/** wait for vmt.js to load completey */
document.addEventListener('vmt_is_ready', ()=>{

    /** register a base canvas that is hidden, and your display canvas
     * that shows final output to user
     * in our live-tryon.html example
     * tryon-live-canvas-0 is basde canvas
     * tryon-live-canvas-1 is display canvas */
```

```

var success = window.VmtHelper.registerCanvases(
    "tryon-upload-img-canvas-0",
    "tryon-upload-img-canvas-1"
);
if(success){
    /** tell VmtHelper if we are dealing with live camera feed or a static
    * single image */
    window.VmtHelper.setLive(false);
}
});

```

4.5 Why different implementations for Live and Static(Uploaded) Image Try-on?

For live stream of images, we detect the face and landmarks once for the first frame and estimate them for consecutive frames. Also we need to apply and display makeup that has been set on every frame.

For static, uploaded images, we detect face and landmarks once per new upload. Also we need to apply and display makeup that has been set every time a makeup element is applied or removed is changed.

4.6 Now applying makeup

Once the above setup is complete now you may want to apply foundation, blush, eye shadow etc. Ideally you will have a button, when you click on it you apply a certain makeup element. So you will bind the button's `onClick` event with an apply makeup call. You can do it by following the example below.

```

<button onClick="window.VmtHelper.removeFoundation();window.VmtHelper.applyFound
<button onClick="window.VmtHelper.removeBlush();window.VmtHelper.applyBlush(247,
<button onClick="window.VmtHelper.removeEyeshadow();window.VmtHelper.applyEyesha

```

note that we remove any previous applied foundation, blush or eye shadow before applying new foundation, blush or eye shadow. if the remove is not called, the try-on will apply new makeup layer on top of the previously applied layer.

5. Helper APIs & documentation

5.1 Camera Controller

Camera Controller primarily controls camera stream. It can be access in your browser console by typing `window.CameraController`

`start()`

return value

Undefined

description

Starts the camera and stream image frames to registered canvas.

`stop()`

return value

Undefined

description

Stops the camera and stream image frames to registered canvas.

`pause()`

return value

Undefined

description

Pauses the stream image frames to registered canvas.

`play()`

return value

Undefined

description

Plays the stream image frames to registered canvas.

`streamToCanvasId(canvas_id)`

parameters

canvas_id : String

return value

Undefined

description

Registers the canvas to which camera shall stream.

`render()`

return value

Undefined

description

Upon calling this function, a loop is started that paints images from camera.

`onEachFrame(cb)`

parameters

cb : Function

return value

Undefined

description

The callback function is called every time a new frame is rendered. Best place to put image processing routines.

5.2 Post VMT (Advanced Topic)

Includes routines for allocating and freeing memory in web assembly. Also `vmt_is_ready` event is generated from within this file.

`_arrayToHeap(typedArray)`

parameters

typedArray : Uint8ClampedArray

return value

Uint8ClampedArray / Undefined

description

Copies array to web assembly's memory context and return this new array.

`_freeArray(heapBytes)`

parameters

heapBytes : Uint8ClampedArray

return value

Undefined

description

Frees memory for the passed array

`_stringToHeap(str)`

parameters

str : String

return value

Number / Undefined

description

Copies string to web assembly's memory context and return its address.

`_freeString(buffer)`

parameters

buffer : Number

return value

Undefined

description

Frees memory for the passed string address

5.3 VMT Helper

VMT Helper is an instantiated object, and can be accessed from browser console by typing `window.VmtHelper` . This object provides a clean API to perform virtual makeup tasks. It provides a way to register your base and display canvases, wraps `vmt_module` methods which performs detection, tracking , makeup application and rendering.

```
registerCanvases(base_canvas_id, display_canvas_id)
```

parameters

base_canvas_id : String

display_canvas_id : String

return value

Boolean

description

Registers base canvas and display canvas. This has to be done, so that other methods in this object instance knows where to pick input images from and where to render output images to.

```
SetLive(bool)
```

parameters

bool : Boolean

return value

Undefined

description

Tells the object instance if it is being used to process live stream on images or just one single image.

```
cycleForLive()
```

return value

Undefined

description

Performs the following actions in order

- get image data from base canvas
- copy image data to web assembly's memory context
- perform landmark detection, tracking and apply makeup by modifying memory
- use this modified image data to render output image on display canvas
- frees unused memory

notes

- base and display canvases should be registered prior to this call
- `setLive(true)` should be called prior to this call

```
cycleForSingleImage(render_only=false)
```

parameters

render_only : Boolean (Default : false)

return value

Undefined

description

Performs the following actions in order

- get image data from base canvas
- copy image data to web assembly's memory context
- perform landmark detection for certain number of iterations (so the detection is accurate), this step is performed if `render_only` parameter is false other wise skips to next step.
- applies makeup by modifying memory
- use this modified image data to render output image on display canvas
- frees unused memory

notes

- base and display canvases should be registered prior to this call
- `setLive(false)` should be called prior to this call

```
clearMakeup()
```

return value

Undefined

description

Removes all makeup, also renders output on display canvas.

```
applyFoundation(r, g, b, a)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

return value

Undefined

description

Applies foundation, also renders output on display canvas.

```
removeFoundation()
```

return value

Undefined

description

Removes foundation, also renders output on display canvas.

```
applyPerfector(r, g, b, a, mask)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

mask : String

return value

Undefined

description

Applies perfector, also renders output on display canvas.

```
removePerfector()
```

return value

Undefined

description

Removes perfector, also renders output on display canvas.

```
applyBlender(r, g, b, a)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

return value

Undefined

description

Applies blender, also renders output on display canvas.

```
removeBlender()
```

return value

Undefined

description

Removes blender, also renders output on display canvas.

```
applyBlush(r, g, b, a, mask, shmr_mask=null, shmr_intensity=null)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

mask : String

shmr_mask : String (Default : null)

shmr_intensity : Number (0 - 1) (Default : null)

return value

Undefined

description

Applies blush, also renders output on display canvas.

```
removeBlush()
```

return value

Undefined

description

Removes blush, also renders output on display canvas.

```
applyLipstick(r, g, b, a, fnsh_intensity, fnsh_type, gltr_mask=null,  
gltr_intensity=null, shmr_mask=null, shmr_intensity=null)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

fnsh_intensity : Number (0 - 1)

fnsh_type: Number (0: MATTE, 1:SATIN, 2:GLOSS)

gltr_mask : String (Default : null)

gltr_intensity : Number (0 - 1) (Default : null)

shmr_mask : String (Default : null)

shmr_intensity : Number (0 - 1) (Default : null)

return value

Undefined

description

Applies lipstick, also renders output on display canvas.

```
removeLipstick()
```

return value

Undefined

description

Removes lipstick, also renders output on display canvas.

```
applyLipliner(r, g, b, a, mask)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

mask : String

return value

Undefined

description

Applies lipliner, also renders output on display canvas.

```
removeLipliner()
```

return value

Undefined

description

Removes lipliner, also renders output on display canvas.

```
applyHighlighter(r, g, b, a, mask, shmr_mask=null, shmr_intensity=null)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

mask : String

shmr_mask : String (Default : null)

shmr_intensity : Number (0 - 1) (Default : null)

return value

Undefined

description

Applies highlighter, also renders output on display canvas.

```
removeHighlighter()
```

return value

Undefined

description

Removes highlighter, also renders output on display canvas.

```
applyContour(r, g, b, a, mask)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

mask : String

return value

Undefined

description

Applies contour, also renders output on display canvas.

```
removeContour()
```

return value

Undefined

description

Removes contour, also renders output on display canvas.

```
applyBronzer(r, g, b, a, mask)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

mask : String

return value

Undefined

description

Applies bronzer, also renders output on display canvas.

```
removeBronzer()
```

return value

Undefined

description

Removes bronzer, also renders output on display canvas.

```
applyEyebrow(r, g, b, a, mask)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

mask : String

return value

Undefined

description

Applies eyebrow, also renders output on display canvas.


```
removeEyebrow()
```

return value

Undefined

description

Removes eyebrow, also renders output on display canvas.

```
applyEyeshadow(r, g, b, a, mask, shmr_mask=null, shmr_intensity=null)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

mask : String

shmr_mask : String (Default : null)

shmr_intensity : Number (0 - 1) (Default : null)

return value

Undefined

description

Applies eyeshadow, also renders output on display canvas.

```
removeEyeshadow()
```

return value

Undefined

description

Removes eyeshadow, also renders output on display canvas.

```
applyEyelash(r, g, b, a, mask)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

mask : String

return value

Undefined

description

Applies eyelash, also renders output on display canvas.

```
removeEyelash()
```

return value

Undefined

description

Removes eyelash, also renders output on display canvas.

```
applyEyeliner(r, g, b, a, mask)
```

parameters

r(ed) : Number (0 - 255)

g(reen) : Number (0 - 255)

b(lue) : Number (0 - 255)

a(lpha) : Number (0 - 1)

mask : String

return value

Undefined

description

Applies eyeliner, also renders output on display canvas.

```
removeEyeliner()
```

return value

Undefined

description

Removes eyeliner, also renders output on display canvas.

6. `vmt_module` API documentation (Advanced Topic)

When `vmt.js`, `vmt.data` and `vmt.wasm` files are correctly loaded into your java script application, it shall create this object, which exposes C++ methods as java script methods that can be called to control the flow and execution of web assembly, in this case makeup ar-tist core logic. `VmtHelper` wraps `vmt_module` to hide its complexity. The API documentation is below, you may try them out in your browser console.

6.1 Quick Test

```
vmt_module._test();
```

Quick test to check if the API is working, returns "hello world".

6.2 Reinitialize WASM

```
vmt_module._main();
```

Re-initializes entire web assembly application. Use this with caution, may re-allocate new memory.

6.3 Debug Messages

```
vmt_module._enableDebugMessages();
```

Enables debug messages to be printed to standard output.

```
vmt_module._disableDebugMessages();
```

Disables debug messages from printing to standard output.

6.4 WASM ready?

```
vmt_module._isVmtReady();
```

This returns a boolean flag, indicating if the web assembly is ready to process images. Never try to process images until WASM is ready.

6.5 Max Iterations

When given an image, the facial landmark tracker ideally runs 5-7 times in order to determine a more accurate geometric representation of the face at each iteration, before it converges. A parameter called `MAX_ITERS` dictates the number of iterations the tracker will take. The default value is 5 for a static image and 1 for a video stream.

```
vmt_module._getMaxIterations();
```

Returns `MAX_ITERS` value.

```
vmt_module._setMaxIterations(4);
```

Sets `MAX_ITERS` value to 4.

6.6 Image Rotation

```
vmt_module._getRotationsMade();
```

When a single image is passed and there is no face detected, we rotate the image 4 times by 90 degrees. If we detect a face, we proceed, else throw a no face detected message. The function returns the number of rotations made for a particular image.

6.7 List Masks

```
vmt_module._listMasks();
```

Prints the list of all masks used to standard output. Some makeup categories require masks along with color information.

6.8 Set Makeup

Collects product, color and mask information and stores it. The colors are rendered upon calling `render()`.

Sets foundation

```
vmt_module._setFoundation(120, 120, 55, 0.6);  
#params: RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Note: JavaScript strings cannot be passed directly into the below functions. Memory needs to be allocated, string needs to be copied to memory and starting address of this memory is passed. This process has to be done when passing mask name (string) to set makeup functions. Following helper function can do this process.

```
function _stringToHeap(str) {  
  try {  
    let length = vmt_module.lengthBytesUTF8(str);  
    let buffer = vmt_module._malloc(length + 1);  
    vmt_module.stringToUTF8(str, buffer, length + 1);  
    return buffer  
  } catch (e) {  
    console.error("[ERR] _stringToHeap err = ", e);  
    return undefined;  
  }  
}
```

To conserve memory, use the below code to free the buffer.

```
vmt_module._free(buffer);
```

Sets perfector

```
vmt_module._setPerfector(_stringToHeap("foundation_conceal_1"), 120, 120, 55, 0.  
#params: MASK[&STRING], RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets blender

```
vmt_module._setBlender(120, 120, 55, 0.6);  
#params: RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets blush

```
vmt_module._setBlush(_stringToHeap("blush_1"), 120, 120, 55, 0.6);  
#params: MASK[&STRING], RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets blush shimmer effect

```
vmt_module._setBlushShimmerEffect(_stringToHeap("blush_shimmer"), 0.6);  
#params: MASK[&STRING], ALPHA [0-1]
```

Sets lipstick

```
vmt_module._setLipstick(120, 120, 55, 0.6);  
#params: RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets lipstick glitter effect

```
vmt_module._setLipGlitterEffect(_stringToHeap("lipGlitter_1"), 0.6);  
#params: MASK[&STRING], ALPHA [0-1]
```

Sets lipstick shimmer effect

```
vmt_module._setLipShimmerEffect(_stringToHeap("lipShimmer_1"), 0.6);  
#params: MASK[&STRING], ALPHA [0-1]
```

Sets lip liner

```
vmt_module._setLipliner(_stringToHeap("lipliner_1"), 120, 120, 55, 0.6);  
#params: MASK[&STRING], RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets highlighter

```
vmt_module._setHighlighter(_stringToHeap("highlighter_1"), 120, 120, 55, 0.6);  
#params: MASK[&STRING], RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets highlighter shimmer effect

```
vmt_module._setHighlighterShimmerEffect(_stringToHeap("highlighter_shimmer"), 0.6);  
#params: MASK[&STRING], ALPHA [0-1]
```

Sets contour

```
vmt_module._setContour(_stringToHeap("bronzer_contour"), 120, 120, 55, 0.6);  
#params: MASK[&STRING], RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets bronzer

```
vmt_module._setBronzer(_stringToHeap("bronzer_fullface"), 120, 120, 55, 0.6);  
#params: MASK[&STRING], RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets eyebrows

```
vmt_module._setEyebrow(_stringToHeap("eyebrow_powder"), 120, 120, 55, 0.6);  
#params: MASK[&STRING], RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets eyelash

```
vmt_module._setEyelash(_stringToHeap("eyelash_1"), 120, 120, 55, 0.6);  
#params: MASK[&STRING], RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets eye shadow

```
vmt_module._setEyeshadow(_stringToHeap("eyeshadow_1"), 120, 120, 55, 0.6);  
#params: MASK[&STRING], RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

Sets eye shadow shimmer effect

```
vmt_module._setEyeshadowShimmerEffect(_stringToHeap("eyeshadow_shimmer"), 0.6);  
#params: MASK[&STRING], ALPHA [0-1]
```

Sets eyeliner

```
vmt_module._setEyeliner(_stringToHeap("eyeliner_1"), 120, 120, 55, 0.6);  
#params: MASK[&STRING], RED[0-255], GREEN[0-255], BLUE[0-255], ALPHA [0-1]
```

6.9 Reset Makeup

Resets all makeup. Need to call `render()` after resetting to clear makeup on the image.

```
vmt_module._resetAllMakeup();
```

Resets foundation and/or perfector

```
vmt_module._resetFoundation();
```

Resets blender

```
vmt_module._resetBlender();
```

Resets blush

```
vmt_module._resetBlush();
```

Resets blush shimmer

```
vmt_module._resetBlushShimmerEffect();
```

Resets lipstick

```
vmt_module._resetLipstick();
```

Resets lipstick shimmer

```
vmt_module._resetLipShimmerEffect();
```

Resets lipstick glitter

```
vmt_module._resetLipGlitterEffect();
```

Resets lip liner

```
vmt_module._resetLipliner();
```

Resets contour

```
vmt_module._resetContour();
```

Resets bronzer

```
vmt_module._resetBronzer();
```

Resets highlighter

```
vmt_module._resetHighlighter();
```

Resets highlighter shimmer

```
vmt_module._resetHighlighterShimmerEffect();
```

Resets eyebrow

```
vmt_module._resetEyebrow();
```

Resets eyeliner

```
vmt_module._resetEyeliner();
```

Resets eye shadow

```
vmt_module._resetEyeshadow();
```

Resets eye shadow shimmer

```
vmt_module._resetEyeshadowShimmerEffect();
```

Resets eye lash

```
vmt_module._resetEyelash();
```

6.10 Landmark Detection

Before we can apply makeup, the algorithm needs to determine facial landmarks. First, because your image is drawn onto a canvas in your HTML document, we will obtain the image buffer from the canvas' context as below.

```
let image_buf = canvas_context.getImageData(0, 0, w, h);
```

Use the following function to allocate memory to heap

```
function _arrayToHeap(typedArray) {  
  try {  
    let numBytes = typedArray.length * typedArray.BYTES_PER_ELEMENT  
    let ptr = vmt_module._malloc(numBytes)  
    heapBytes = vmt_module.HEAPU8.subarray(ptr, ptr + numBytes)
```



```

    heapBytes.set(typedArray)
    return heapBytes
} catch (e) {
    console.error("[ERR] _freeAToHeap err = ", e);
    return undefined;
}
}

```

Pass image data to the above allocation function, it shall return starting address of the allocated memory.

```
let frame_bytes = _arrayToHeap(image_buf.data);
```

If `frame_bytes` is already defined and pointing to a memory location, make sure you free it before you reassign as follows.

```
vmt_module._free(frame_bytes.byteOffset);
```

Now we will pass the following for landmark detection. The function will return true, if detection was successful.

```
let detected = vmt_module._landmarkDetection(width, height, frame_bytes.byteOffset);
```

You may now call set and render makeup functions after detecting landmarks.

6.11 Render Makeup

We will pass the starting address of the allocated memory for the image along with width and height.

```
vmt_module._renderMakeup(width, height, frame_bytes.byteOffset);
```

Then we will use the same buffer which was passed to update the image on the canvas.

```
let image_buf.set(frame_bytes);
canvas_context.putImageData(image_buf, 0, 0);
```

6.12 Live Stream Landmark Detection & Rendering

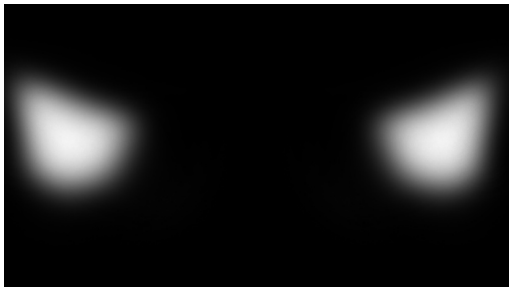

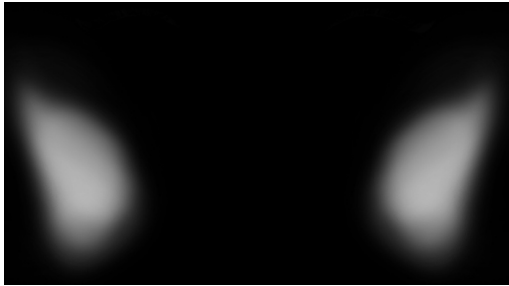
There are small differences between the landmark detection and rendering life cycle of a static image and a live stream. Live stream only has to perform detection once when no face is detected. Static images have to perform 5-7 iterations on each image every time.


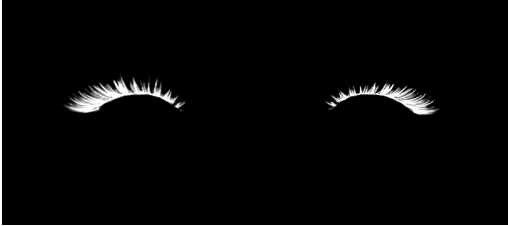
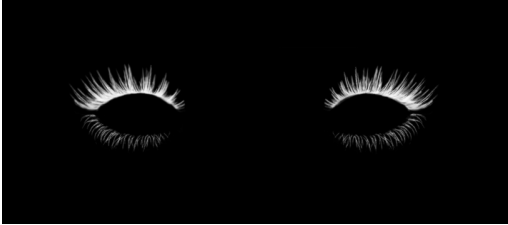
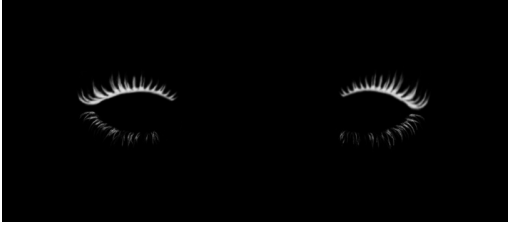
Additionally, on live stream `render()` has to be implicitly called for every frame to have the real time effect. The live stream work flow will be as follows.

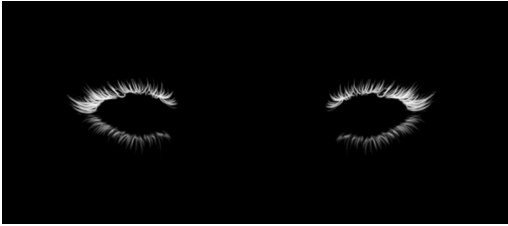



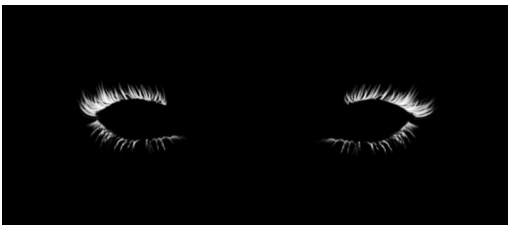

```
let image_buf = canvas_context.getImageData(0, 0, w, h);
let frame_bytes = _arrayToHeap(image_buf.data);
let detected = vmt_module._landmarkDetectionFullLifeCycle(width, height, frame_b
if(!detected) return;
let image_buf.set(frame_bytes);
canvas_context.putImageData(image_buf, 0, 0);
```


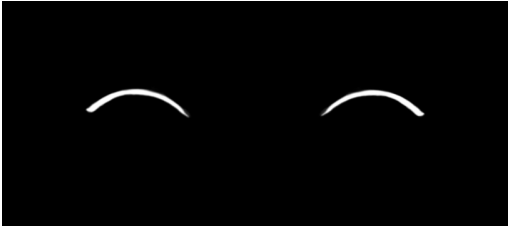
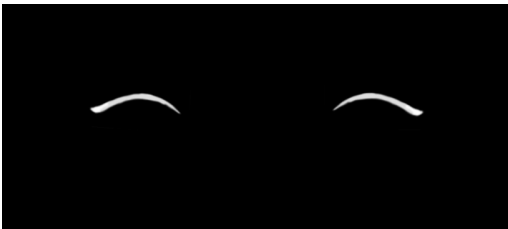

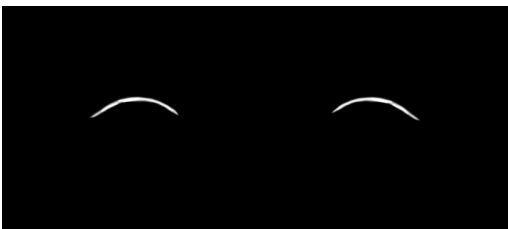

this code shall be called for every frame in a live stream.

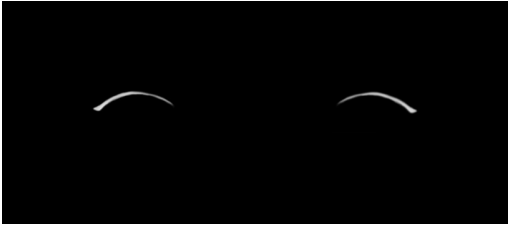
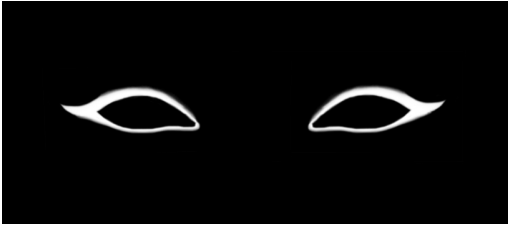


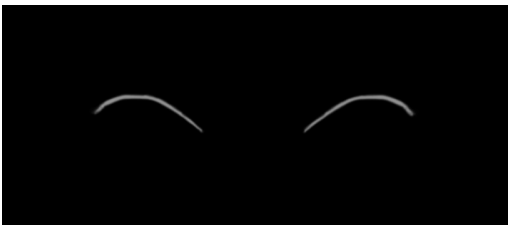

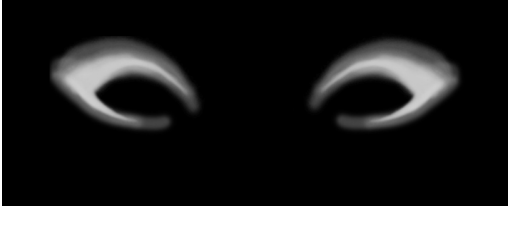
7. Makeup Mask Directory

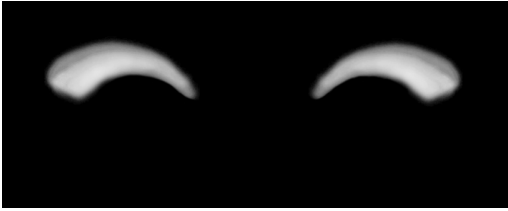
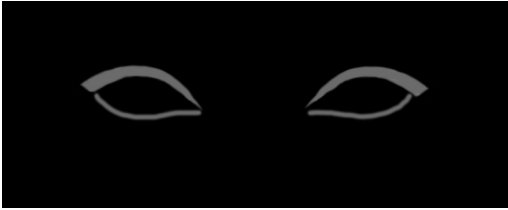


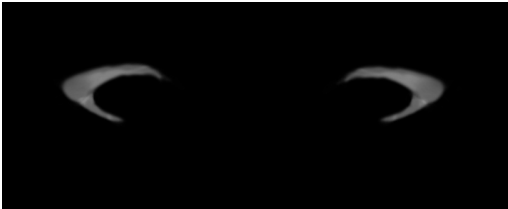


## Mask Name	## Mask Category	## Mask Image
blush_1	BLUSH	
blush_2	BLUSH	
blush_3	BLUSH	

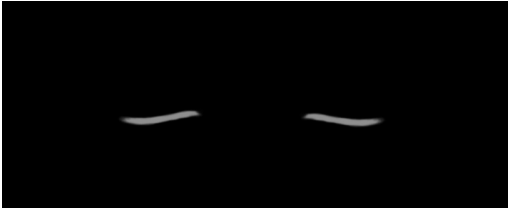


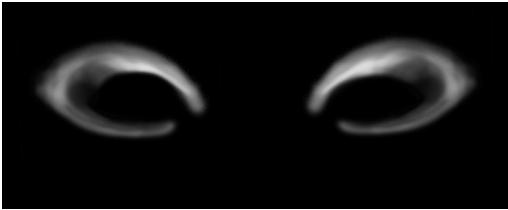

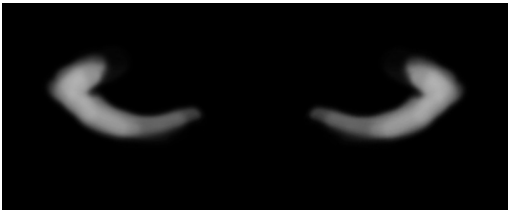

## Mask Name	## Mask Category	## Mask Image
blush_shimmer	BLUSH - SHIMMER	 A black and white mask showing two soft, glowing, fan-shaped areas on the cheeks, representing blush shimmer.
eyelash_1	EYELASH	 A black and white mask showing two sets of long, dark, curved eyelashes on the upper eyelids.
eyelash_2	EYELASH	 A black and white mask showing two sets of long, dark, curved eyelashes on the upper eyelids, with a slightly different shape than mask 1.
eyelash_3	EYELASH	 A black and white mask showing two sets of long, dark, curved eyelashes on the upper eyelids, with a slightly different shape than mask 2.
eyelash_4	EYELASH	 A black and white mask showing two sets of long, dark, curved eyelashes on the upper eyelids, with a slightly different shape than mask 3.
eyelash_5	EYELASH	 A black and white mask showing two sets of long, dark, curved eyelashes on the upper eyelids, with a slightly different shape than mask 4.



## Mask Name	## Mask Category	## Mask Image
eyelash_6	EYELASH	
eyelash_7	EYELASH	
eyelash_8	EYELASH	
eyelash_9	EYELASH	
eyelash_10	EYELASH	
eyelash_11	EYELASH	
eyelash_12	EYELASH	






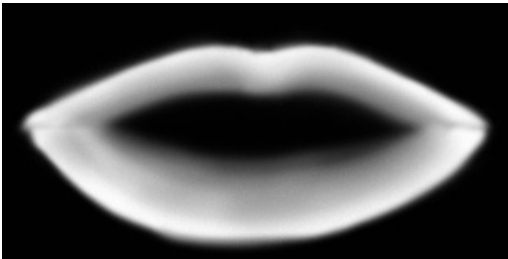
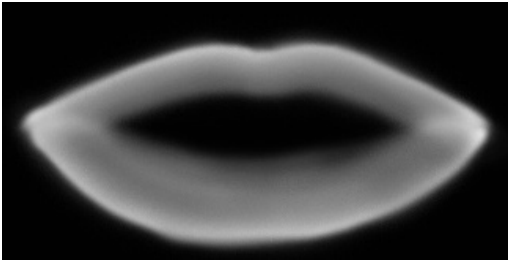
## Mask Name	## Mask Category	## Mask Image
eyelash_13	EYELASH	
eyeliner_1	EYELINER	
eyeliner_2	EYELINER	
eyeliner_3	EYELINER	
eyeliner_4	EYELINER	
eyeliner_5	EYELINER	
eyeliner_6	EYELINER	

## Mask Name	## Mask Category	## Mask Image
eyeliner_7	EYELINER	
eyeliner_8	EYELINER	
eyeliner_9	EYELINER	
eyeliner_10	EYELINER	
eyeliner_11	EYELINER	
eyeliner_12	EYELINER	
eyeshadow_1	EYESHADOW	

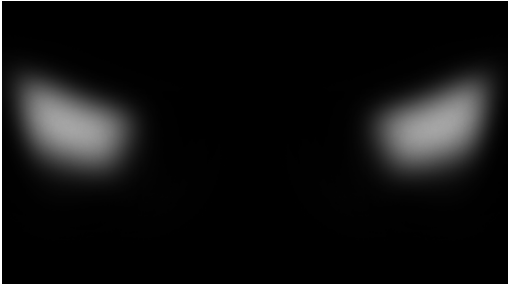

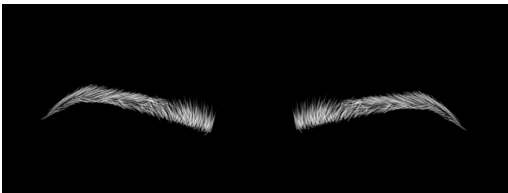

## Mask Name	## Mask Category	## Mask Image
eyeshadow_2	EYESHADOW	
eyeshadow_3	EYESHADOW	
eyeshadow_4	EYESHADOW	
eyeshadow_5	EYESHADOW	
eyeshadow_6	EYESHADOW	
eyeshadow_7	EYESHADOW	
eyeshadow_8	EYESHADOW	






## Mask Name	## Mask Category	## Mask Image
eyeshadow_9	EYESHADOW	
eyeshadow_10	EYESHADOW	
eyeshadow_11	EYESHADOW	
eyeshadow_12	EYESHADOW	
eyeshadow_13	EYESHADOW	
eyeshadow_14	EYESHADOW	
eyeshadow_15	EYESHADOW	



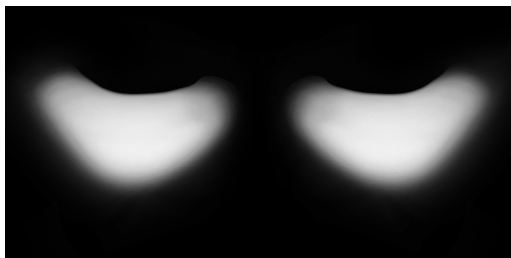
## Mask Name	## Mask Category	## Mask Image
eyeshadow_16	EYESHADOW	
eyeshadow_17	EYESHADOW	
eyeshadow_18	EYESHADOW	
eyeshadow_19	EYESHADOW	
eyeshadow_20	EYESHADOW	
eyeshadow_21	EYESHADOW	
eyeshadow_22	EYESHADOW	

## Mask Name	## Mask Category	## Mask Image
eyeshadow_23	EYESHADOW	
eyeshadow_24	EYESHADOW	
eyeshadow_25	EYESHADOW	
eyeshadow_26	EYESHADOW	
eyeshadow_shimmer	EYESHADOW - SHIMMER	
lipliner_1	LIPLINER	
lipliner_2	LIPLINER	

## Mask Name	## Mask Category	## Mask Image
lipliner_3	LIPLINER	
lipGlitter_1	LIPSTICK - GLITTER	
lipGlitter_2	LIPSTICK - GLITTER	
lipGlitter_3	LIPSTICK - GLITTER	
lipGlitter_4	LIPSTICK - GLITTER	
lipShimmer_1	LIPSTICK - SHIMMER	

## Mask Name	## Mask Category	## Mask Image
highlighter_1	HIGHLIGHTER	
highlighter_2	HIGHLIGHTER	
highlighter_shimmer	HIGHLIGHTER-SHIMMER	
eyebrow_powder	EYEBROW	
eyebrow_powder_3	EYEBROW	
eyebrow_powder_4	EYEBROW	
eyebrow_powder_5	EYEBROW	

## Mask Name	## Mask Category	## Mask Image
eyebrow_creme_lowerUpper_2	EYEBROW	 A grayscale mask showing two distinct, curved, light-colored areas representing eyebrows on a black background.
eyebrow_creme_lowerUpper_1	EYEBROW	 A grayscale mask showing two distinct, curved, light-colored areas representing eyebrows on a black background, similar to the first mask but with slightly different shading.
bronzer_contour	BRONZER	 A grayscale mask showing two light-colored, teardrop-shaped areas on the sides of the face, representing contouring or bronzer application.
bronzer_sunkiss	BRONZER	 A grayscale mask showing a wide, horizontal, light-colored band across the cheeks and chin, representing a bronzer or 'sunkiss' effect.
bronzer_fullface	BRONZER	 A grayscale mask showing a large, light-colored area covering the entire face, representing a full-face bronzer or contouring mask.

## Mask Name	## Mask Category	## Mask Image
bronzer_fullface_contour_2	BRONZER	
bronzer_fullface_contour_3	BRONZER	
foundation_conceal_1	PERFECTOR	

8. FAQ's

Why do I see this message [ERR] invalid license. contact sdk provider ?

This could mean that your trial license has expired, you may want to contact info@algoface.ai to renew or get a new developer/commercial license.

Why do I see uncaught <number> **error?**

When wrong data type is passed to a function exposed by web assembly, you may see this kind of error. Please check the JavaScript code where this error originated.

9. Contact Us

support@algoface.ai