Mod9 Visual Search API
Version 0.1

The Mod9 Visual Search API allows images to be uploaded, processed, and analyzed in with respect to visual similarity within categories. It is a RESTful API, in which resource paths are typically denoted as:

```
/$ACCOUNT/$CATEGORY/$IMGID/$VIEW
```

Subpaths of the above structure are also resources. Some paths may be appended with query arguments, for example:

```
?arg1=val1&arg2=val2
```

This server implements some common HTTP methods (PUT, DELETE, GET,POST) as well as a special method (UNLOAD) that provides advanced internal server control.

1. API base URL and version

The API server is located at a user-specified host and is configured to serve requests on port **80**. The API endpoints are relative to this base URL:

```
http://$YOURSERVER/
```

The current version of the API is **v0.1**, which should be displayed if you GET the base URL above:

```
{"version":"0.1", "accounts":[...], "cached":{...}}
```

2. PUT requests: create or replace resources

```
PUT /$ACCOUNT
PUT /$ACCOUNT/$CATEGORY
```

Creates an account and/or category; returns 200 or 201. (This is like mkdir –p)
Examples might be /asos/womens_shoes or /fashionfinder/$USERID

```
PUT /$ACCOUNT/$CATEGORY/$IMGID
```

Creates or replaces an image resource; returns 200 or 201.
The request body should be a JSON object structured as follows:

```
{"url":"IMAGE_URL_HERE", "data"="BASE64_DATA_HERE",
"prodhash":"SHORT_HASH_HERE"}
```

If the data is non-null, the server will try to fetch the image from the specified URL. The representation above will be stored on the server, along with some other

internal information related to the image processing and visual search data structures. Any other specified keys specified in the request will be stored as well. The optional prodhash field is a hack: use it to specify a unique product identifier that can be used to bias shape matches -- e.g. products with the same brand and model name should have identical shape, but different color. If a field is not specified, and the resource exists, that field is not updated. The Content-Type is assumed to be 'application/json'; if specified as 'image/*' the request body should be raw image data. Below are several examples:

```
PUT /myshop/shoes/42
...
Content-Type: application/json

{"url":"http://dummyimage.com/100x100.jpeg"}
```

This creates or replaces the resource by downloading the given URL. If the image is not accessible at a public URL, but you have the image data available locally, you can send it in two alternative ways:

```
PUT /myshop/shoes/42
...
Content-Type: image/jpeg

$RAW_BYTES_OF_IMAGE_DATA
```

You can send the raw image data by specifying a Content-Type header.

```
PUT /myshop/shoes/42
...
Content-Type: application/json

{"data":"$BASE_ENCODING_OF_IMAGE_DATA"}
```

Or you can send the image data in the JSON object, as a base-64 encoding. This is slightly less efficient in terms of bandwidth, but the advantage is that you could specify other JSON object keys if desired. For example:

```
PUT /myshop/shoes/42
...
Content-Type: application/json

{"url":"http://dummyimage.com/100x100.jpeg",
"prodhash":"adidas-samba", "my_key":"some_val_of_mine"}
```

The special "prodhash" key will be used to bias visual similarity results: images with matching values for this key should differ only in color. Any other specified keys are used to provide metadata that is retained but generally ignored by the Visual

Search API service. (In the future, user-specified metadata may be useful for filtering results.)

`PUT /$ACCOUNT/$CATEGORY/$IMGID?bg_remove=True`
This query argument would be associated with a future capability allowing image backgrounds to be automatically removed. **(TODO!)**

3. DELETE requests: remove resources

`DELETE /$ACCOUNT`
`DELETE /$ACCOUNT/$CATEGORY`
`DELETE /$ACCOUNT/$CATEGORY/$IMGID`
Deletes a specified path, including all resources at subpaths. Returns 204.
Note that this action is not reversible, and there's not such much as a warning. Be careful! Note also that the API service does not currently enforce any sort of authentication or access control; so anyone with access to the API could delete (and access) your data. It is recommended that security measures be enacted by the application of network-level firewall, such as using EC2's security groups.

4. GET requests: view resources

GET requests can retrieve resources according to various views. Responses to these requests are typically generated on-the-fly, although the server will try to intelligently preload features and cache distances when possible.

4.1 Metadata

`GET /`
`GET /$ACCOUNT`
`GET /$ACCOUNT/$CATEGORY`
Returns a list of accounts, or categories in an account, or imgids in a category. Also returns some statistics about how many features or distances are cached; this should only concern an advanced user. If the path is the base URL (first example above), the server's API version is given.

`GET /$ACCOUNT/$CATEGORY/$IMGID`
Returns a compact JSON object describing an image's associated metadata -- not the internally stored image or visual search representations.

4.2 Display images

`GET /$ACCOUNT/$CATEGORY/$IMGID/image`
Returns the raw image, as internally stored (i.e. what was downloaded or uploaded). The Content-type header should reflect the appropriate MIME-type.

`GET /$ACCOUNT/$CATEGORY/$IMGID/image.{jpeg,jpg,png,gif,bmp,tiff}`
Return the image in a converted file format.

`GET /$ACCOUNT/$CATEGORY/$IMGID/image.$FMT?resize=$WIDTHx$HEIGHT`

Resize image dimensions to the specified width and height, in pixels.

`GET /$ACCOUNT/$CATEGORY/$IMGID/image.$FMT?flop=True`
Reflect the image about the vertical axis, i.e. flip it left-to-right. This is useful for displaying images such as shoes that look good when pointed the same direction.

`GET /$ACCOUNT/$CATEGORY/$IMGID/image.$FMT?normalize=True`
Scale and shift the image so that the bounding box around non-white pixels is horizontally centered and vertically aligned with the bottom baseline. This makes images from various sources look nice for display, as they are all similarly sized and look as if they're sitting on a shelf, rather than floating in mid-air.

`GET /$ACCOUNT/$CATEGORY/$IMGID/image.$FMT?bg_remove=True`
Automatically remove a background from an image. This is difficult. **(TODO!)**

4.3 Color naming

`GET /$ACCOUNT/$CATEGORY/$IMGID/color`
Returns color name(s) automatically derived from this image. **(TODO!)**

4.4 Visual similarity

`GET /$ACCOUNT/$CATEGORY/$IMGID/similar`
This is the core functionality of visual search. Returns a list of pairs: imgids and visual similarity matching score.

`GET /$ACCOUNT/$CATEGORY/$IMGID/similar?num_results=N`
This optional query argument indicate how many results to return.

`GET /$ACCOUNT/$CATEGORY/$IMGID/similar?color_weight=0.5`
This optional query argument indicates how much significance to apply to the color vs. shape similarity. A value of 1.0 specifies that only color is to be considered, while a value of 0.0 specifies that only shape should be considered.

`GET /$ACCOUNT/$CATEGORY/$IMGID/similar?categories=$CAT1,$CAT2,..`
By default, the results will be considered from among all other images in the same category; if specified, results may come from multiple categories specified as a comma-delimited query argument. Note that the imgids in the results are not associated with a category, and identical imgids in different categories will be overwritten by the rightmost specified category.

`GET /$ACCOUNT/$CATEGORY/$IMGID/similar?cached=True`
This optional query argument indicates that previously loaded features or computed distances should be exploited. This may speed up performance in some cases, but may also result in inconsistent results if the caches are stale.

5. POST requests

POST requests augment the functionality of the Visual Search API service.

5.1 Parameterize visual similarity

```
POST /$ACCOUNT/$CATEGORY/$IMGID/similar
```

In addition to supporting the query arguments provided of the corresponding GET method, this resource view may be accessed with a POST request including a JSON object as the body. This is used to provide more fine-tuned filtering capabilities. For example, the request could send an arbitrary list of imgids to search, or perhaps even a bitvector if the imgids can be assumed to be range-limited indices. It may also be possible to search with a filter applied to user-specified metadata; for example, price and brand names. **(TODO!)**

5.2 Update or create resources

```
POST /$ACCOUNT/$CATEGORY/$IMGID
```

Similar to PUT, but will *update* rather replace existing resources: representation retains any prior metadata that are not overwritten by the specified request.

```
POST /$ACCOUNT/$CATEGORY
```

Similar to PUT /$ACCOUNT/$CATEGORY/$IMGID, except the server will create a hash from the image data and use that as the $IMGID. This returns a 201, and the path to the new resource in the response's Location header. This may be useful for posting new images to be queried against a given category.

6. UNLOAD requests: drop caches

```
UNLOAD /
UNLOAD /$ACCOUNT
UNLOAD /$ACCOUNT/$CATEGORY
```

This non-standard HTTP method specifies that the server should drop cached data. Subsequent visual similarity requests may be a bit slower as the caches are rebuilt.

7. Contact

This API service was designed and implemented by Arlo Faria:

api@mod9.com
+1 (510) 705-3223