

## FastImage

**FastImage finds the size or type of an image given its uri by fetching as little as needed**

### The problem

Your app needs to find the size or type of an image. This could be for adding width and height attributes to an image tag, for adjusting layouts or overlays to fit an image or any other of dozens of reasons.

But the image is not locally stored - it's on another asset server, or in the cloud - at Amazon S3 for example.

You don't want to download the entire image to your app server - it could be many tens of kilobytes, or even megabytes just to get this information. For most common image types (GIF, PNG, BMP etc.), the size of the image is simply stored at the start of the file. For JPEG files it's a little bit more complex, but even so you do not need to fetch much of the image to find the size.

FastImage does this minimal fetch for image types GIF, JPEG, PNG, TIFF, BMP, ICO, CUR, PSD, SVG, WEBP and JXL. And it doesn't rely on installing external libraries such as RMagick (which relies on ImageMagick or GraphicsMagick) or ImageScience (which relies on FreeImage).

You only need supply the uri, and FastImage will do the rest.

### Features

- Reads local (and other) files - anything that is not parseable as a URI will be interpreted as a filename, and - will attempt to open it with `File#open`.
- Automatically reads from any object that responds to `:read` - for instance an IO object if that is passed instead of a URI.
- Follows up to 4 HTTP redirects to get the image.
- Obey the `http_proxy` setting in your environment to route requests via a proxy. You can also pass a `:proxy` argument if you want to specify the proxy address in the call.
- Add a timeout to the request which will limit the request time by passing `:timeout => number_of_seconds`.

- 
- Returns `nil` if it encounters an error, but you can pass `:raise_on_failure => true` to get an exception.
  - Provides a reader for the content length header provided in HTTP. This may be useful to assess the file size of an image, but do not rely on it exclusively - it will not be present in chunked responses for instance.
  - Accepts additional HTTP headers. This can be used to set a user agent or referrer which some servers require. Pass an `:http_header` argument to specify headers, e.g., `:http_header => {'User-Agent' => 'Fake Browser'}`.
  - Gives you information about the parsed display orientation of an image with Exif data (jpeg or tiff).
  - Handles Data URIs correctly.

## Security

Take care to sanitise the strings passed to FastImage; it will try to read from whatever is passed.

## Examples

```
1 require 'fastimage'
2
3 FastImage.size("https://switchstep.com/images/ios.gif")
4 => [196, 283] # width, height
5 FastImage.type("http://switchstep.com/images/ss_logo.png")
6 => :png
7 FastImage.type("/some/local/file.gif")
8 => :gif
9 File.open("/some/local/file.gif", "r") {|io| FastImage.type(io)}
10 => :gif
11 FastImage.size("http://switchstep.com/images/favicon.ico", :
12   raise_on_failure=>true, :timeout=>0.01)
13 => raises FastImage::ImageFetchFailure
14 FastImage.size("http://switchstep.com/images/favicon.ico", :
15   raise_on_failure=>true, :timeout=>2)
16 => [16, 16]
17 FastImage.size("http://switchstep.com/images/faulty.jpg", :
18   raise_on_failure=>true)
19 => raises FastImage::SizeNotFound
20 FastImage.new("http://switchstep.com/images/ss_logo.png").
21   content_length
22 => 4679
23 FastImage.size("http://switchstep.com/images/ss_logo.png", :http_header
24   => {'User-Agent' => 'Fake Browser'})
```

---

```
20 => [300, 300]
21 FastImage.new("http://switchstep.com/images/ExifOrientation3.jpg").
    orientation
22 => 3
23 FastImage.size("
    wAAACH5BAEAAAAALAAAAABAAEAAICRAEOw==")
24 => [1, 1]
```

## Installation

### Gem

```
1 gem install fastimage
```

### Bundler

Add fastimage to your Gemfile.

```
1 gem 'fastimage'
```

Then you're off - just use `FastImage.size()` and `FastImage.type()` in your code as in the examples.

## Documentation

<http://sdsykes.github.io/fastimage/rdoc/FastImage.html>

## Maintainer

FastImage is maintained by Stephen Sykes (sdsykes). SamSaffron also helps out from time to time (thanks!).

## Benchmark

It's way faster than conventional methods for most types of file when fetching over the wire. Compared here by using OpenURI which will fetch the whole file.

```
1 require 'benchmark'
2 require 'fastimage'
3 require 'open-uri'
```

---

```

4
5 uri = "http://upload.wikimedia.org/wikipedia/commons/b/b4/
   Mardin_1350660_1350692_33_images.jpg"
6 puts Benchmark.measure {URI.open(uri, 'rb') {|fh| p FastImage.size(fh)
   }}
7 [9545, 6623]
8   0.059088   0.067694   0.126782 ( 0.808131)
9
10 puts Benchmark.measure {p FastImage.size(uri)}
11 [9545, 6623]
12   0.006198   0.001563   0.007761 ( 0.162021)

```

The file is fetched in about 0.8 seconds in this test (the number in brackets is the total time taken), but as FastImage doesn't need to fetch the whole thing, it completes in 0.16s.

You'll see similar excellent results for the other file types.

```

1 require 'benchmark'
2 require 'fastimage'
3 require 'open-uri'
4
5 uri = "https://upload.wikimedia.org/wikipedia/commons/a/a9/
   Augustine_Herrman_1670_Map_Virginia_Maryland.tiff"
6 puts Benchmark.measure {URI.open(uri, 'rb') {|fh| p FastImage.size(fh)
   }}
7 [12805, 10204]
8   1.332587   2.049915   3.382502 ( 19.925270)
9
10 puts Benchmark.measure {p FastImage.size(uri)}
11 [12805, 10204]
12   0.004593   0.000924   0.005517 ( 0.100592)

```

Some tiff files however do not have their metadata near the start of the file.

```

1 require 'benchmark'
2 require 'fastimage'
3 require 'open-uri'
4
5 uri = "https://upload.wikimedia.org/wikipedia/commons/1/14/Center-
   Filled_LIMA.tif"
6 puts Benchmark.measure {URI.open(uri, 'rb') {|fh| p FastImage.size(fh)
   }}
7 [22841, 19404]
8   0.350304   0.321104   0.671408 ( 3.053605)
9
10 puts Benchmark.measure {p FastImage.size(uri)}
11 [22841, 19404]
12   0.163443   0.214301   0.377744 ( 2.880414)

```

Note that if you want a really fast result for this file type, `image_size` might be useful as it has an opti-

---

misation for this (fetching only the needed data range).

```
1 require 'benchmark'
2 require 'image_size/uri'
3 require 'fastimage'
4
5 uri = "https://upload.wikimedia.org/wikipedia/commons/1/14/Center-
    Filled_LIMA.tif"
6 puts Benchmark.measure {p ImageSize.url(uri).size }
7 [22841, 19404]
8 0.008983 0.001311 0.010294 ( 0.128986)
9
10 puts Benchmark.measure {p FastImage.size(uri)}
11 [22841, 19404]
12 0.163443 0.214301 0.377744 ( 2.880414)
```

## Tests

You'll need to bundle, or `gem install fakeweb` and possibly also `gem install test-unit` to be able to run the tests.

```
1 ruby test/test.rb
```

## References

- Pennysmall's - Find jpeg dimensions fast in pure Ruby, no image library needed
- Antti Kupila - Getting JPG dimensions with AS3 without loading the entire file
- image\_size gem
- EXIF Reader

## FastImage in other languages

- Python by bmuller
- Swift by kaishin
- Go by rubenfonseca
- PHP by tommoor
- Node.js by ShogunPanda
- Objective C by kylehickinson
- Android by qstumn
- Flutter by ky1vstar

---

### **Also of interest**

- C++ by xiaozhuai
- Rust by xiaozhuai

### **Licence**

MIT, see file “MIT-LICENSE”