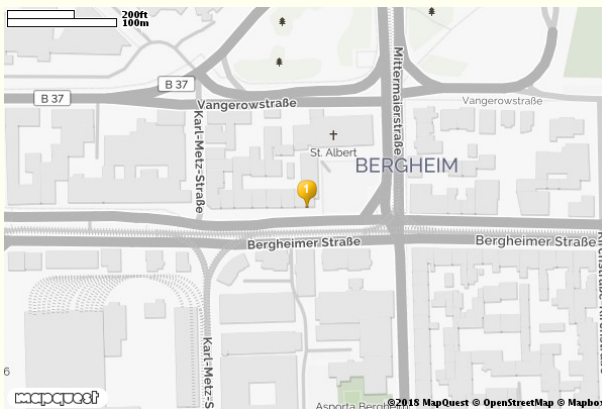


getmap.sty

v1.10

Downloading maps from Open-StreetMap, Google Maps or Google Street View



2018/07/18

Package author:
Josef Kleber

1 Options	5
1.1 General options	5
1.1.1 mode	5
1.1.2 inputencoding	5
1.1.3 overwrite	5
1.1.4 file	5
1.2 osm mode	5
1.2.1 key	5
1.2.2 scale	6
1.2.3 zoom	6
1.2.4 xsize	6
1.2.5 ysize	6
1.2.6 imagetype	6
1.2.7 type	6
1.2.8 color	6
1.2.9 number	7
1.3 gm mode	7
1.3.1 scale	7
1.3.2 zoom	7
1.3.3 xsize	7
1.3.4 ysize	7
1.3.5 imagetype	7
1.3.6 type	7
1.3.7 color	7
1.3.8 number	8
1.3.9 language	8
1.3.10 markers	8
1.3.11 visible	9
1.3.12 path	10
1.3.13 pathfile	11
1.4 gsv mode	11
1.4.1 xsize	11
1.4.2 ysize	11
1.4.3 heading	11
1.4.4 pitch	11
1.4.5 fov	12
2 Command(s)	12
2.1 \getmap	12
3 Examples	13
4 The getmapd\ Lua script	16
5 How to define routes	17
5.1 OpenStreetMap	17
5.2 Google Maps	18
5.2.1 Long routes	21

6 Implementation	23
7 References	31
8 Change History	32
9 Index	33

Abstract

The goal of this package is the simplest possible provision of map images (OpenStreetMap, Google Maps and Google Street View are supported). In the simplest case, it is sufficient to specify an address. The package loads the map using the `\write18` feature, which you must activate to use this package. The image will be downloaded by an external Lua script. You can use this script also from the command line.

Acknowledgment

I want to thank Norbert Preining, who did most of the recoding of `osmimage` (Bash \rightarrow Lua). Moreover many thanks to Taco Hoekwater, Akira Kakuto, Reinhard Kotucha and Heiko Oberdiek for their valuable contributions. Finally, I want to thank Doug Currie for helping me to implement the algorithm for encoded polylines in Lua.

1 Options

The following options can be used as package options with global scope, as well as options for the `\getmap` command with local scope!

1.1 General options

1.1.1 mode (osm|gm|gsv)

This option sets the mode, that is the source of the images. OpenStreetMap, Google Maps or Google Street View! Please note that – if used as local option (mixed modes) – the default values of the scale, zoom, type and color options for the respective mode are reset to guarantee correct download URLs!

1.1.2 inputencoding

This option specifies the input encoding of your file. The download script requires the strings encoded in utf8. For the safe conversion the input encoding of the file is required. Normally, you don't have to specify an encoding. The package tries to evaluate the encoding given to `inputenc` or assumes utf8. Usually that should work.

1.1.3 overwrite (false|true)

With this option, you can specify whether the image should be downloaded in any case. By default, the option is set to `false` in order to save bandwidth and compilation time. Nevertheless a check is performed on the existence of the image and the image will be downloaded, if it is not present. In the case of `true`, the image will be downloaded anyway! BTW, `overwrite` is equivalent to `overwrite=true`.

Note

changed default value to `getmap` in version 1.2!

1.1.4 file (getmap)

This option allows you to specify the name of the image (without extension).

1.2 osm mode

1.2.1 key (Fmjtd|luur20u22d,75=o5-9aylh6)

In `osm` mode, the download script requires a key in order to use the service of MapQuest. By default, it uses a key, which is registered for `getmap`. But you can register and use your own key with this option. The default key is stored in

getmap.cfg. You can copy this file to your local TeX tree and store your own key there¹! This file will be found after running texhash!

1.2.2 scale (3385)

This option allows you to specify a display scale for the map image in the range of 1692 – 221871572. You will not necessarily see a difference between 5000 and 5500. A scale value of 3385 corresponds to a zoom level of 17.

1.2.3 zoom

This option allows you to specify a zoom level in the range of 1 – 18. This option overwrites a possibly given scale.

Note

changed default
value to 600 in ver-
sion 1.2!

1.2.4 xsize (600)

This option specifies the width of the map in pixels. If you only want to slightly increase or decrease the map extract, you should adjust the size of the map. You still have full control over the size of the map in the document with the options of \includegraphics. (max: 3840)

1.2.5 ysize (400)

This option specifies the height of the map in pixels. (max: 3840)

1.2.6 imagetype (png|jpeg|jpg|gif)

This option allows you to specify the type of the image.

1.2.7 type (map|sat|hyb)

This option specifies the type of the map. It seems as if there would be only a few regions of Mother Earth, for which satellite and hybrid images are available.

1.2.8 color (yellow_1)

This option specifies the color of the marker. Possible colors:

<http://open.mapquestapi.com/staticmap/icons.html>

¹Mapquest will deliver an url-encoded key, which must be decoded to ASCII, e.g. by [Url decode](#)

1.2.9 number (1)

This option specifies the number of the marker.

1.3 gm mode

1.3.1 scale (1)

For the free version of Google Maps the image size is limited to 640x640. You can set scale to a value of 2, to get exactly the same map in doubled size in pixels.

1.3.2 zoom (17)

This option allows you to specify a zoom level in the range of 0 – 21.

1.3.3 xsize (600)

This option specifies the width of the map in pixels. If you only want to slightly increase or decrease the map extract, you should adjust the size of the map. You still have full control over the size of the map in the document with the options of \includegraphics. (max: 640)

1.3.4 ysize (400)

This option specifies the height of the map in pixels. (max: 640)

1.3.5 imagetype (png|png8| png32|gif|jpg (progressive)|jpg-baseline (flat))

This option allows you to specify the type of the image.

1.3.6 type (roadmap|satellite|hybrid|terrain)

This option specifies the type of the map.

1.3.7 color (blue)

This option specifies the color of the marker. Possible colors:

black, brown, green, purple, yellow, blue, gray, orange, red, white or in hex format 0x3399FF

1.3.8 number (1)

This option specifies the number of the marker. Google Maps also allows uppercase letters: [A-Z]!

1.3.9 language (en)

This option specifies the language of the map labels. Of course, not all languages are supported for all countries. At least, english and one of the national languages should be supported. Possible option values: en, de, fr, es, it, fi, ...

1.3.10 markers

This option allows you to set more than just the standard marker, which will no longer be used! You don't have to specify an address, as Google Maps will deliver an image with all markers on the map. Nevertheless, you can specify an address, which will define the center of the map. This option expects one or more URL parameters like:

`&markers=size:mid|color:blue|label:S|loc1|loc2|...`



```

1 \getmap[
2   file=bmus1, mode=gm,
3   markers={&markers=size:mid|label:B|color:green|52.521847,13.394398%
4             &markers=label:P|color:green|Pergamonmuseum, Berlin%
5             &markers=label:N|color:blue|52.520063,13.397525}%
6   }{}
7 \includegraphics[width=10cm]{bmus1}

```


Earlier versions of this document used POIs for all museums. After an update from Google Maps, the quality of the geo codings deteriorated, at least in this example from Berlin. Using addresses or geographical coordinates usually solves this problem. See [7] for more information.

The parameters size, color and label are optional!

size tiny, mid, small

color red, black, brown, green, purple, yellow, blue, gray, orange, white, 0x1188FF

label [0-9][A-Z] (only in mid size!)

The default is a mid-sized red bubble with a black point!

1.3.11 visible

With this option you can specify a list of locations (separated by a pipe), which must be on the map!



```

1 \getmap[
2 file=bmus2, mode=gm
3 markers={&markers=size:mid|label:B|color:green|52.521847,13.394398%
4           &markers=label:P|color:green|Pergamonmuseum, Berlin%
5           &markers=label:N|color:blue|52.520063,13.397525},%
6 visible={Brandenburger Tor, Berlin|Reichstagsufer 1, Berlin}}{
7 \includegraphics[width=10cm]{bmus2}

```

1.3.12 path

With this option you can define one or more paths! It expects one or more URL parameters like:

`&path=weight:5|color:orange|loc1|loc2|...`

```

1 \getmap[file=bmus3, mode=gm, language=de,
2 markers={&markers=size:mid|label:B|color:green|52.521847,13.394398%
3           &markers=label:P|color:green|Pergamonmuseum, Berlin%
4           &markers=label:N|color:blue|52.520063,13.397525},%
5 path={&path=weight:5|color:orange|52.521847,13.394398|
6       Pergamonmuseum, Berlin|52.520063,13.397525|
7       James-Simon-Park,Berlin|52.522649,13.402523%
8       &path=weight:5|color:purple|James-Simon-Park, Berlin|
9       Weinmeisterstraße 6, Berlin}}}
10 \includegraphics[width=10cm]{bmus3}

```



You can also use `fillcolor` to mark areas! In paths, you can also specify RGB32 colors, in which the last byte defines opacity, e.g. 55 (33%).

```

1 \getmap[file=cpny, mode=gm, zoom=13,
2 path={&path=weight:2|color:orange|fillcolor:0xff641A55|
3       40.764302, -73.973004|40.768044, -73.981903|
4       40.800642, -73.958193|40.796887, -73.949226|
5       40.764302, -73.973004}}{Central Park, New York}
6 \includegraphics[width=10cm]{cpny}

```



With small enough spaces between way points you can also defines routes!

1.3.13 pathfile

This option specifies the file holding the path specification. It will be loaded by the Lua script. You can use the `filecontents*` environment to keep the definition in your document. It should be a one line utf8-encoded file!

1.4 gsv mode

1.4.1 xsize (600)

This option specifies the width of the map in pixels. (max: 640)

1.4.2 ysize (400)

This option specifies the height of the map in pixels. (max: 640)

1.4.3 heading (0)

This option specifies the heading (direction) in degrees in the range of 0 – 360. (0: north, 90: east, ...)

1.4.4 pitch (0)

This option specifies the pitch (angle) of the camera view in degrees in the range of -90 – 90.

1.4.5 fov (90)

This option specifies the field of horizontal view (kind of zoom) in degrees in the range of 0 – 120.

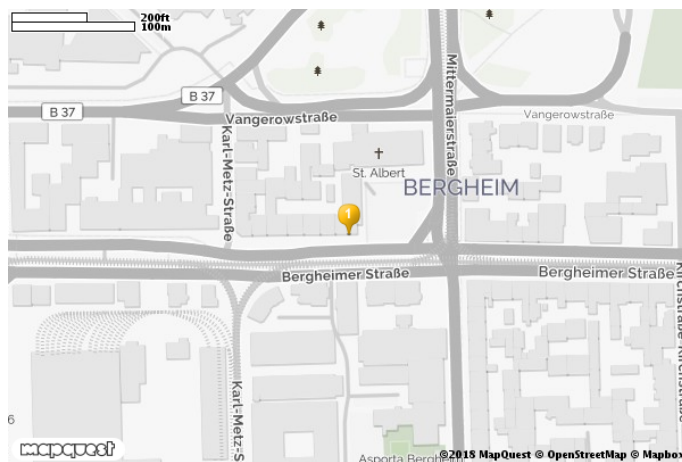
2 Command(s)

2.1 \getmap

`\getmap[options]{address}`

With the `\getmap` command you can download a map, if you enable `\write18` (TeXLive: `-shell-escape`, MiKTeX: `--enable-write18`). This is only necessary if you actually download an image. You can use the options described above to specify the properties of the downloaded image. After executing the command, the image is available in the current working directory!

In the simplest case, you only need an address, a POI or geographic coordinates (latitude, longitude) to download the map. `{address}` must be fully expanded and must not contain macros! By default, the image is saved under the name `getmap.png`! If you need only one map (e.g. the office of Dante e.V.) in your document, it can be as simple as:



```
1 \getmap{Bergheimer Straße 110A, 69115 Heidelberg, Germany}
2 \includegraphics[width=9cm]{getmap}
```

3 Examples



The same map as before from Google Maps:



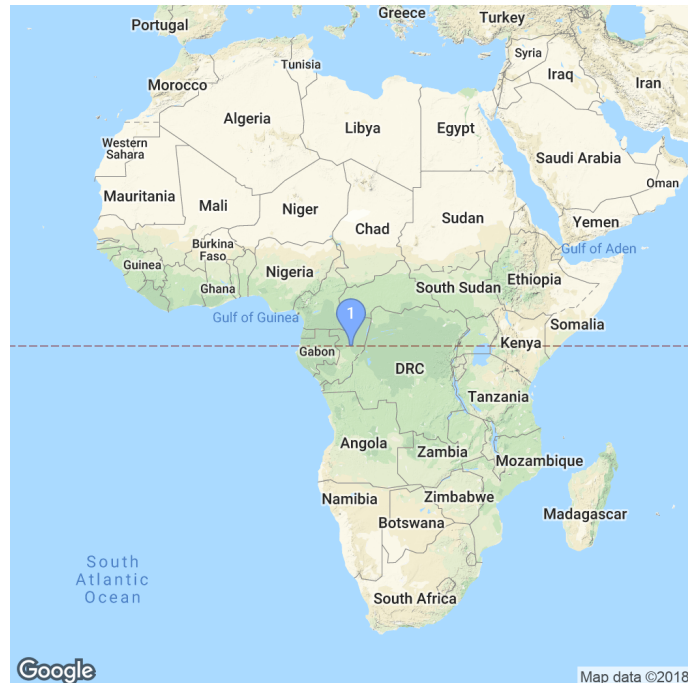
```
1 \getmap[file=dantegm,mode=gm]{Bergheimer Straße 110A,%
2                               69115 Heidelberg, Germany}
3 \includegraphics[width=9cm]{dantegm}
```

The same map as satellite image:



```
1 \getmap[file=dantegmsat,mode=gm,type=satellite]
2       {Bergheimer Straße 110A, 69115 Heidelberg, Germany}
3 \includegraphics[width=9cm]{dantegmsat}
```


L'afrique, mon amour!

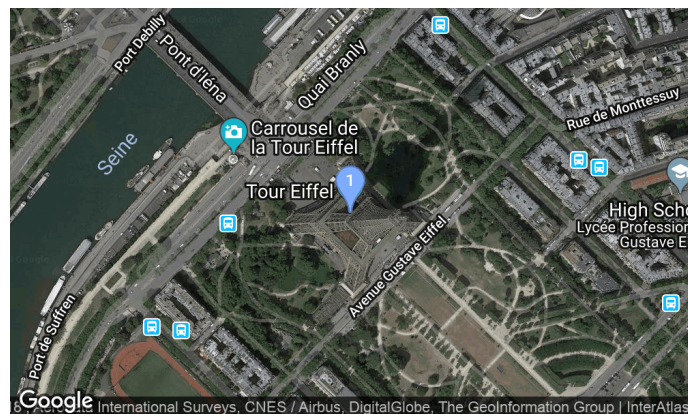


```

1 \getmap[file=africa,mode=gm,type=terrain,xsize=500,ysize=500,%
2   scale=2,zoom=3]{0,16}
3 \includegraphics[width=9cm]{africa}

```

L'amour, ...



```

1 \getmap[file=paris,mode=gm,type=hybrid,xsize=500,ysize=300,%
2   scale=2,zoom=16]{Tour Eiffel, Paris}
3 \includegraphics[width=9cm]{paris}

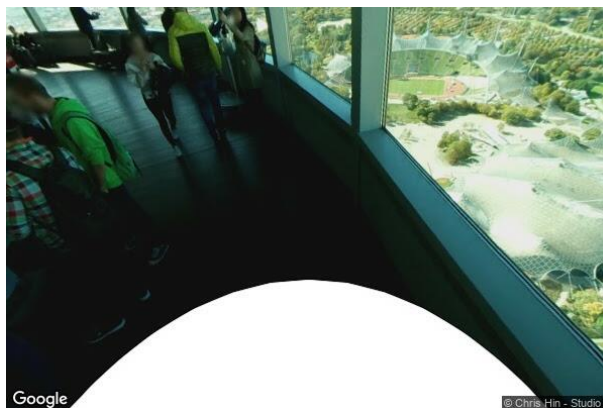
```

Street View now:



```
1 \getmap[file=parisgsv,mode=gsv,heading=320,pitch=30,fov=40,%  
2 xsize=300,ysize=600]{Avenue Piere-Loti, Paris}  
3 \includegraphics[width=4cm]{parisgsv}
```

View from Olympic Tower Munich (Olympic Stadium and Park):



```
1 \getmap[file=mucoly,mode=gsv,heading=260,pitch=-40,fov=90]%  
2 {Olympiaturm}  
3 \includegraphics[width=8cm]{mucoly}
```

4 The getmapdl Lua script

Basically, the getmapdl Lua script downloads static map images depending on command line options and allows to parse kml, gpx and gps (a plain list of geographical coordinate pairs (latitude, longitude) on each line) files and outputs gps or encoded polylines (epl). The script offers the following modes (-m):

osm downloads a static map image based on OpenStreetMap data

gm downloads a static map image based on Google Maps data

gsv downloads an image based on Google Street View data

kml2epl parses a kml file and outputs geographical coordinates of places and encoded polylines (epl) for routes and lines to STDOUT

kml2gps parses a kml file and outputs geographical coordinates

gpx2epl parses a gpx file and outputs encoded polylines

gpx2gps parses a gpx file and outputs a list of geographic coordinate pairs (gps)

gps2epl parses a gps file and outputs epl

gps2gps parses a gps file and outputs – based on a given bound – a reduced list of gps coordinates

The first three modes are used by \getmap. You may use the script also from the command line! getmapdl -h will give you a list of available command line options.

The other modes are useful for creating encoded polylines (epl), which is the route format of Google Maps. You can parse the following example from Google Maps in gpx format

```

1 <trkseg>
2   <trkpt lon="-120.2" lat="38.5"/>
3   <trkpt lon="-120.95" lat="40.7"/>
4   <trkpt lon="-126.453" lat="43.252"/>
5 </trkseg>

```

with

```

1 $ getmapdl -m gpx2epl -G test.gpx
2 _p~iF~ps|U_ulLnnqClqNvxq'@

```

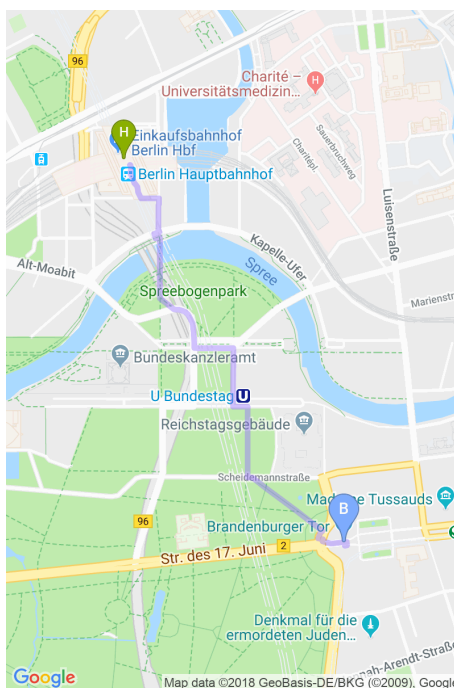
This encoded polyline can be used for the path or pathfile option of \getmap.

5 How to define routes

Routes are described by so called encoded polylines and can be used with `enc:polyline_data` as location specifier in a path. This string can contain all sorts of troublesome characters for \LaTeX . `\getmap` can deal with them, with the exception of curly braces! These will break your \LaTeX document. As a work-around, use the `pathfile` option. Please note that the length of the URL is limited to 2048 bytes. So, there's no way to support extreme long paths!

5.1 OpenStreetMap

OpenStreetMap does not offer routing service directly, but you can use an OpenStreetMap based route service² to create your route and export it to a gpx file³. It's basically a xml-packaged list of geographical coordinates. You can use the `getmapdl` script to convert a route to encoded polylines, e.g. a pedestrian route from Berlin Central Station to Brandenburg Gate:



```

1 \begin{filecontents*}{berlin.epl}
2 &path=weight:5|color:purple|enc:_xq_IcgrpA?AFE@?^BFE@A^U@CLQXEZU?
3 gCR?B?DBF@@?vA?D?D?BAHE@JBN@JLGF CG[DC~C?@?F?R?vA?p@iB@i@Fe@JWRSTO
4 f@Gh@C^A?e@?gE?w@r@?lB@hA?'@?M?aA?]dI??0?0?Cn@cBfBeF|AeEHNvNBc@H

```

²<http://maps.openrouteservice.org>

³This also means that you can visualize your own routes tracked with hardware or a software app!

```

5 @H_AEWa?0AMNA@N
6 \end{filecontents*}
7 \getmap[file=berlin, mode=gm, language=de, scale=2,
8     xsize=400, ysize=600,
9     markers={\markers=size:mid|label:H|color:green|Berlin, Hbf
10         \markers=label:B|color:blue|Brandenburger Tor, Berlin},
11     pathfile={berlin.epl}}{}
12 \includegraphics[width=6cm]{berlin}

```

5.2 Google Maps

One possible way is to use Google Maps' online interactive map tool⁴!



```

1 \begin{filecontents*}{muc.epl}
2 &path=weight:5|color:orange|enc:okydHa}peAX0Hi@jANv@A@kJJmFAiDJuA
3 HmDXcBb@cC'@gC^_DV}CJaCF{APaBVkBdAyEf@qC'@{B\adSwB
4 \end{filecontents*}
5 \getmap[file=muc, mode=gm,
6     markers={\markers=size:mid|label:H|color:green|München, Hbf
7         \markers=label:S|color:yellow|Stachus, München
8         \markers=label:M|color:blue|Marienplatz, München},
9     pathfile={muc.epl}}{}
10 \includegraphics[width=10cm]{muc}

```

You can also use the **new** version of *My Maps*. It allows you to define markers, routes and arbitrary lines on different layers and to export these into a kml file, e.g. `berlin.kml`⁵:

⁴<https://developers.google.com/maps/documentation/utilities/polylineutility>

⁵<https://bitbucket.org/kleberj/getmap/downloads/Berlin.kml>

```

1 $ getmapdl -m kml2epl -K Berlin.kml
2 Route: Route von Berlin Hbf, Moabit nach Brandenburger Tor, Paris
3 er Platz, Berlin
4 k}q_IufrpA?iFQ?gCDQBMHSm@Wq@GQIK]{AUaA}AaICIMm@Kg@EUiByIi@cDSqA_@
5 uBIa@a@gBpB[fAMhCS|@Gd@Ev@Ep@GfAIpC[bAMr@IbAMp@G\GtBUdCSp@MrAK~AQ
6 pAMhAKx@IjDc@VAB?D@@?B?dCU'AKjAMRCPLGntEBhABh@BVF'@D'@D\rB{@ZKDAH
7 CLAHABHBFBHFFDNHRHJHHDDHDBHF?F?VCJCJILQJUJg@HQJMJKHAFaj?X?bCRH@R?
8 AWQ?K@iBKq@GYAK@EBIFEFEFECFAD
9
10
11 Point: Berlin Hbf, Moabit [Europaplatz 1, 10557 Berlin, Deutschl
12 and]
13 52.52581820000001,13.3695451
14
15
16 Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21,
17 10117 Berlin, Deutschland]
18 52.5159099,13.3773202
19
20
21 Point: Berlin Hbf [arrive with train]
22 52.5249948,13.368988
23
24
25 Point: Reichstag [nice view from the roof]
26 52.5185973,13.3758974
27
28
29 Point: Brandenburger Tor [once behind the wall]
30 52.5163514,13.3789873
31
32
33 Route: Route von Berlin Hbf, Moabit nach Pariser Platz, Berlin
34 k}q_IufrpA?iFQ?@hH@H@F@B@B@?DB?~|B@F@DB@B@~FCB?X@V??0?0?0?wF@W?o
35 @?k@?Y?eAvAD'@?'@ARCFAP?hAAZ?B?D?J?B?DAJB^@T?~B?MsAAIAEAM?S@QBM?A
36 @IDKBKDKFKFIJEFCJAJATAL?XAN?P?VALBCbB?L@RB?kH@}Gp@]|@0bA?BAz@JTF^
37 LZN?wF@aCE]~Am@\KREJCNANAH@HBJHHFLPP\DBFFDB@@@?D@F?JAJCKiEAQ?K@I?
38 KBSXC
39
40
41 Point: Berlin Hbf, Moabit [Europaplatz 1, 10557 Berlin, Deutschl
42 and]
43 52.52581820000001,13.3695451
44
45
46 Point: Deutscher Bundestag Redaktion Das Parlament, Berlin [Plat

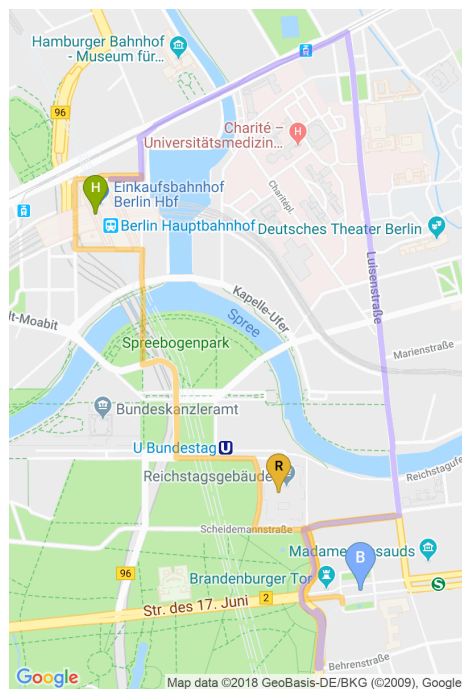
```

```

47 z der Republik 1, 10557 Berlin, Deutschland]
48 52.518502000000005,13.3751849
49
50
51 Point: Pariser Platz, Berlin [Pariser Platz 1, 10117 Berlin, Deu
52 tschland]
53 52.5160749,13.3783013

```

Now, you can take these data for your map:



```

1 \begin{filecontents*}{berlin2.epl}
2 &path=weight:5|color:orange|enc:k|q_IufrpA?iFQ?@hH@H@F@B@B@?DB?\?
3 |B@F@DB@B@~FCB?X@V??0?0?0?wF@W?o@?k@?Y?eAvAD'@?'@ARCFAP?hAAZ?B?D?
4 J?B?DAJB^@T?~B?MsAAIAEAM?S@OBM?A@IDKBKDKFKFIJEFCJAJATAL?XAN?P?VAL
5 BCbB?L@RB?kH@}Gp@]|@0bA?BAz@JTF^LZN?wF@aCE]~Am@\KREJCNANAH@HBJHHF
6 LPP\DBFFDB@@@?D@F?JAJCKiEAQ?K@I?KBSXC&path=weight:5|color:purple|
7 enc:k|q_IufrpA?iFQ?gCDQBMHSm@Wq@GQIK]{AUaA}AaICIMm@Kg@EUiByIi@cDS
8 qA_@uBIa@a@gBpB[fAMhCS|@Gd@Ev@Ep@GfAIpC[bAMr@IbAMP@G@GtBUdCSp@MrA
9 K~AQpAMhAKx@IjDc@VAB?D@@?B?dCU'AKjAMRCPlGntEBhABh@BVF'@D'@D\rB{@Z
10 KDAHCLAHAHBHBFHFFDNHRHJHDDHDHBF?F?VCJJCJILQJUJg@HQJMKHFAJ?X?bCR
11 H@R?AWQ?K@iBKq@GYAK@EBIFEFEFECFAD
12 \end{filecontents*}
13 \getmap[file=berlin2, language=de, xsize=400, ysize=600,
14 scale=2, mode=gm,
15 markers={&markers=size:mid|label:H|color:green|52.5249948,13.3689

```

```

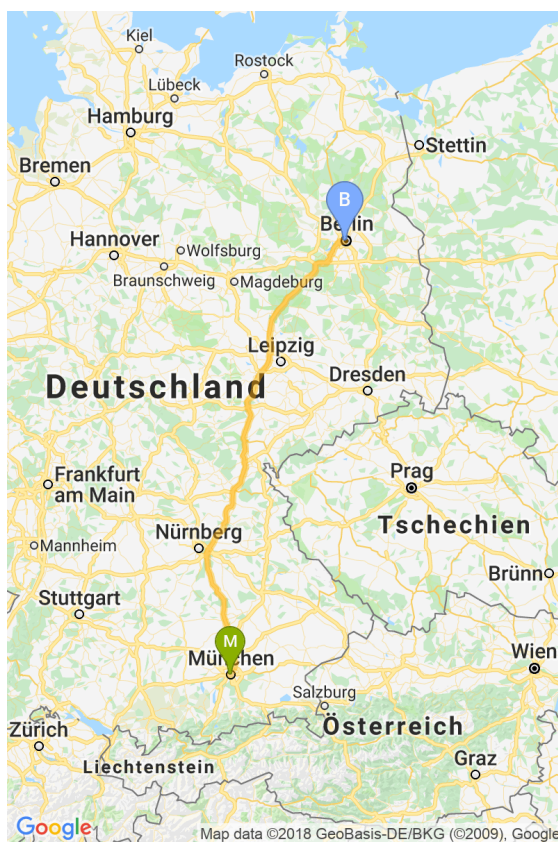
16 88
17     &markers=size:mid|label:R|color:yellow|52.5185973,13.375
18 8974
19     &markers=label:B|color:blue|52.5163514,13.3789873},
20 pathfile={berlin2.epl}}{}%
21 \includegraphics[width=6cm]{berlin2}%

```

5.2.1 Long routes

Long routes are defined by a huge number of way points, but the URL length is limited to 2048 bytes. The following [example](#)⁶ (Stachus, Munich → Brandenburg Gate, Berlin) consists of more than 6000 way points. A created polyline would hugely exceed the URL length limit!

After transferring the kml file into a gps file, you can use the `gps2gps` mode of `getmapdl` to reduce the number of way points by a given bound. It takes a new pair of gps coordinates only if the difference of latitude or longitude is larger than the given bound! Finally, you can use the new gps file to create an encoded polyline.



⁶<https://bitbucket.org/kleberj/getmap/downloads/MucBer.kml>

```

1 $ getmapdl -m kml2gps -K MucBer.kml >MucBer.gps
2
3 $ cat MucBer.gps
4 Route: Route von Stachus, München nach Brandenburger Tor, Pariser
5 Platz, Berlin
6 48.13903,11.56556
7 48.1392,11.56562
8 [ ... many, many way points ...]
9 52.5159,13.37735
10 52.51591,13.37732
11
12
13 Point: Stachus, München [Karlsplatz 10, 80335 München, Deutschla
14 nd]
15 48.13903,11.56556
16
17
18 Point: Brandenburger Tor, Pariser Platz, Berlin [Ebertstraße 21,
19 10117 Berlin, Deutschland]
20 52.51591,13.37732
21
22 $ getmapdl -m gps2gps -G MucBer.gps -B 0.025 >MucBerR.gps
23
24 Route 1: reduced gps coordinates (Bound = 0.025): 6119 -> 193
25
26 $ getmapdl -m gps2epl -G MucBerR.gps
27
28 Route: Route von Stachus, München nach Brandenburger Tor, Pariser
29 Platz, Berlin
30
31 }cydHw{qeAc|CmbCcoCq{CswDk'Be~C_oAqcDe_@wdD~eAs_Dt~AsbDn{A{aEhk@m
32 aD'i@kcDhAk}CoHcnD|Hg{CvVk{Cmx@i|Cpe@c|Cp|@kaClxDk~CzxCqnDzkAk'Df
33 tAc}CrrAonDtqBu|C~m@e~CoGc}Csd@wnD{EyjDr@s}C'@}iDoBskCn|CgbCp_D_p
34 B~aD_cB~aDc}CgOssBr}Ci'Cj{CwzAl_Dc|Cnp@w{C~dAagDnk@o{Cnu@o{ChcCcl
35 EhdBalDni@u{CpUagFfRa}CfLq|CxQu{Cr\m{C{sBwqC}|Cm}CucDuhD}jDa}GiuD
36 { |CkuBm~CeqCkoAwdDkyAkmDc'CkiDmaEm'Ck|CeCakBwFD}_D{gBo|CbUw}C~yAw
37 |Cqc@{ }C}'BumC_|Cqr@kvDmSk}Cu{CiaAkDbGi_DoGw_DyLk'DxH{ _Dmb@q}Cyr
38 @{cAkaDcs@{_DarAy{C_|By_Dy~Ck'Ci}CeMm|CqiCi|CmzAmgDkUm'D_wBg}CiVu
39 ~C|aBg|Cva@i{Cc\}zBv~Cw'DyyBq_D_gB{|CoEu{ChLacDfSy|CeBk{CjsC}~Ca~
40 @_}Cil@w|Coz@e_DidCecDivBe|Cqc@{{CemAa_DcyBq|C}|A_|CtuAsfE|dDwzDr
41 z@e~CvXo~C|c@s{CicCirB}jDkq@q}C{dDiuBkcDsnAm}CsnAk}DuI{|CkMmvFmsE
42 y|CmuBc~Cw{C_'Do~CqhEujHekBc|CkvCmaDufDqgC_}Csh@e|Cnq@kuDemAm}Ccz
43 A{bEgg@cvdLyAiaDbUi{Cjw@i}CsLy}F}I_cDc{ @gjDi_@o_Ds}{|D_c@o|Cy}A}q
44 BubDo}CowCoyBe|Cy_Du_CotAg}C}}@obD{sCu{CkrB_jDqoFcmCsmBq_D_l@w}Ca
45 z@mdD}eBa'DmuBonE_nA_|Cw'BaaDaLDqfCu_CwyDenBqwDqzA{~C}mDkgEcdAmcD
46 si@{'DabBq{Ccx@s{CsfDwsDueDvR_hD_WcuAu_Dsj@g~DkF{kFf_@waDcOa'DkDc

```

```

47 |C}_DceCiu@{aDu|Ckv@m{CcpAiz@k|CmeD}eBeaEedCg'CeaDe'Cy{CwdBipEgJ_
48 gDw0y{Fq@at@

```

Taking a look into the log file, we find:

```

1 getmapdl.lua:
2 url = http://maps.googleapis.com/maps/api/staticmap? ...
3 url length = 1866 bytes
4 output = mucber.png

```

With 193 way points we almost reached the URL length limit of 2048 bytes. The accuracy of the encoded polyline is obviously good enough. So, about 200 way points seem to be a good choice. But the length of an encoded pair of gps coordinates depends on the space between two points and may vary between 2 and 8 bytes!

6 Implementation

```
1 <(*package>
```

First, we provide the \LaTeX package getmap.

```

2 \NeedsTeXFormat{LaTeX2e}%
3 \ProvidesPackage{getmap}[2018/07/18 v1.10 getmap.sty - Josef Kleber (C) 2014,2016,2018]%

```

We need a few packages!

```

4 \RequirePackage{xkeyval}%
5 \RequirePackage{stringenc}%
6 \RequirePackage{ifthen}%

```

Newer versions of \LuaTeX v0.85+ no longer supports \write18 ! Therefore, we use \shellesc instead.

```
7 \RequirePackage{shellesc}%
```

Fix a bug with problematic url encodings coming up with \luatex v1.0.7 (TeXLive 2018) fixed by Akira Kakuto (see: [TeXLive mailing list](#))

```

8 \RequirePackage{ifxetex}%
9 \ifxetex%
10 \def\StringEncodingConvert#1#2#3#4%
11 {%
12 \def#1{#2}%
13 }%
14 \fi%
15 \ifluatex%
16 \def\StringEncodingConvert#1#2#3#4%
17 {%
18 \def#1{#2}%

```



```

19 }%
20 \fi%

```

We provide a macro `\GM@JK@define@key`, which defines package options with global scope and options for `\getmap` with local scope. It takes four arguments `{⟨prefix⟩}`, `{⟨package⟩}`, `{⟨option⟩}` and `{⟨default⟩}`.

```

21 \newcommand*\GM@JK@define@key[4]%
22 {%
23   \expandafter\gdef\csname#1@#3\endcsname{#4}%
24   \define@key{#2.sty}{#3}[#4]%
25   {%
26     \expandafter\gdef\csname#1@#3\endcsname{##1}%
27   }%
28   \define@key{#2}{#3}%
29   {%
30     \expandafter\def\csname#1@#3\endcsname{##1}%
31   }%
32 }%
33 \newcommand*\GM@JK@define@key@detok[4]%
34 {%
35   \expandafter\gdef\csname#1@#3\endcsname{#4}%
36   \define@key{#2.sty}{#3}[#4]%
37   {%
38     \expandafter\gdef\csname#1@#3\endcsname{\detokenize{##1}}%
39   }%
40   \define@key{#2}{#3}%
41   {%
42     \expandafter\def\csname#1@#3\endcsname{\detokenize{##1}}%
43   }%
44 }%

```

Now, we can use this macro to define our options.

```

45 \GM@JK@define@key{GM@JK}{getmap}{mode}{osm}%
46 \GM@JK@define@key{GM@JK}{getmap}{key}{}%
47 \GM@JK@define@key{GM@JK}{getmap}{xsize}{600}%
48 \GM@JK@define@key{GM@JK}{getmap}{ysize}{400}%
49 \GM@JK@define@key{GM@JK}{getmap}{scale}{3385}%
50 \GM@JK@define@key{GM@JK}{getmap}{zoom}{}%
51 \GM@JK@define@key{GM@JK}{getmap}{type}{map}%
52 \GM@JK@define@key{GM@JK}{getmap}{imagetype}{png}%
53 \GM@JK@define@key{GM@JK}{getmap}{color}{yellow_1}%
54 \GM@JK@define@key{GM@JK}{getmap}{number}{1}%
55 \GM@JK@define@key{GM@JK}{getmap}{heading}{0}%
56 \GM@JK@define@key{GM@JK}{getmap}{fov}{90}%
57 \GM@JK@define@key{GM@JK}{getmap}{pitch}{0}%
58 \GM@JK@define@key{GM@JK}{getmap}{language}{en}%
59 \GM@JK@define@key@detok{GM@JK}{getmap}{markers}{}%
60 \GM@JK@define@key@detok{GM@JK}{getmap}{path}{}%
61 \GM@JK@define@key@detok{GM@JK}{getmap}{visible}{}%
62 \GM@JK@define@key{GM@JK}{getmap}{pathfile}{}%
63 \GM@JK@define@key{GM@JK}{getmap}{file}{getmap}%

```



```

64 \GM@JK@define@key{GM@JK}{getmap}{inputencoding}{}%
65 \GM@JK@define@key{GM@JK}{getmap}{overwrite}{true}%

```

For options without default value, we define reasonable default values! We overwrite the default for overwrite, because we don't want overwrite to be true by default, but that overwrite is equivalent to `overwrite=true`!

Moreover, we load `getmap.cfg` to set the default key. You can copy this file to your local $\text{T}_\text{E}\text{X}$ tree and replace the key with your own!

We try to use the input encoding specified for `inputenc` or `utf8` instead.

```

66 \gdef\GM@JK@overwrite{false}%
67 \gdef\GM@JK@key{}%
68 %
69 \IfFileExists{getmap.cfg}%
70 {%
71   \input{getmap.cfg}%
72 }%
73 {%
74   \gdef\GM@JK@key{Fmjtd|luur20u22d,75=o5-9aylh6}%
75 }%
76 %
77 \@ifpackageloaded{inputenc}%
78 {%
79   \gdef\GM@JK@inputencoding{\inputencodingname}%
80 }%
81 {%
82   \gdef\GM@JK@inputencoding{utf8}%
83 }%
84 %

```

Later, we will need a switch, if `\write18` is enabled.

```

85 \newif\ifGM@JK@writexviii\GM@JK@writexviiiifalse%
86 %

```

We execute the package options to define and set the option macros.

```

87 \ExecuteOptionsX{mode,xsize,ysize,scale,zoom,type,imagetype,color,number,file,heading,fov,
88 %
89 \ProcessOptionsX\relax%
90 %

```

We need to reset some defaults in `gm` mode.

```

91 %
92 \ifthenelse{\equal{\GM@JK@mode}{gm}}{%
93 {%
94   \gdef\GM@JK@scale{1}%
95   \gdef\GM@JK@zoom{17}%
96   \gdef\GM@JK@type{roadmap}%
97   \gdef\GM@JK@color{blue}%
98 }%
99 {}%

```

100 %

We check if `\pdf@shellescape` is available to test if `\write18` is enabled.

If false, we assume `\write18` is available and hope for the best.

If true, we set the switch `\GM@JK@writexviii` accordingly!

```

101 %
102 \ltx@ifundefined{pdf@shellescape}%
103 {%
104   \PackageInfo{getmap}{\pdf@shellescape is undefined}%
105   \PackageInfo{getmap}{can not test if \write18 is available}%
106   \GM@JK@writexviii>true%
107 }%
108 {%
109   \PackageInfo{getmap}{\pdf@shellescape is available}%
110   \ifnum\pdf@shellescape=1\relax%
111     \PackageInfo{getmap}{\write18 enabled}%
112     \GM@JK@writexviii>true%
113   \else%
114     \GM@JK@writexviii>false%
115   \fi%
116 }%
117 %

```

We define a macro that is executed as `\ShellEscape` call. First, we test if `\write18` is enabled and issue a package error if not! Otherwise we execute `\ShellEscape` depending on the mode

```

118 \newcommand*\GM@JK@shellescape%
119 {%
120   \ifGM@JK@writexviii\relax%
121   \ifthenelse{\equal{\GM@JK@mode}{osm}}{%
122     {%
123       \ifluatex\directlua{os.setlocale('','ctype')}\fi%
124       \ShellEscape{getmapdl \space-l\space "\GM@JK@location@string"%
125         \space-m\space osm%
126         \space-k\space "\GM@JK@key@string"%
127         \space-x\space \GM@JK@xsize%
128         \space-y\space \GM@JK@ysize%
129         \space-z\space "\GM@JK@zoom"%
130         \space-s\space \GM@JK@scale%
131         \space-t\space \GM@JK@type%
132         \space-i\space \GM@JK@imagetype%
133         \space-c\space "\GM@JK@color"%
134         \space-n\space \GM@JK@number%
135         \space-o\space \GM@JK@file}%
136     }%
137   {%
138     \ifthenelse{\equal{\GM@JK@mode}{gm}}{%
139     {%
140       \ifluatex\directlua{os.setlocale('','ctype')}\fi%

```

```

141     \ShellEscape{getmapdl \space-l\space "\GM@JK@location@string"%
142                               \space-m\space gm%
143                               \space-x\space \GM@JK@xsize%
144                               \space-y\space \GM@JK@ysize%
145                               \space-z\space \GM@JK@zoom%
146                               \space-s\space \GM@JK@scale%
147                               \space-t\space \GM@JK@type%
148                               \space-i\space \GM@JK@imagetype%
149                               \space-c\space "\GM@JK@color"%
150                               \space-n\space \GM@JK@number%
151                               \space-L\space "\GM@JK@language"%
152                               \space-M\space "\GM@JK@markers@string"%
153                               \space-C\space "\GM@JK@location@string"%
154                               \space-P\space "\GM@JK@path@string"%
155                               \space-p\space "\GM@JK@pathfile"%
156                               \space-V\space "\GM@JK@visible@string"%
157                               \space-o\space \GM@JK@file}%
158   }%
159   {%
160     \ifthenelse{\equal{\GM@JK@mode}{gsv}}{%
161       {%
162         \ifluatex\directlua{os.setlocale('','ctype')}\fi
163         \ShellEscape{getmapdl \space-l\space "\GM@JK@location@string"%
164                                   \space-m\space gsv%
165                                   \space-x\space \GM@JK@xsize%
166                                   \space-y\space \GM@JK@ysize%
167                                   \space-H\space \GM@JK@heading%
168                                   \space-F\space \GM@JK@fov%
169                                   \space-T\space \GM@JK@pitch%
170                                   \space-o\space \GM@JK@file}%
171       }%
172     }%
173     \PackageError{getmap}{invalid mode}{invalid mode! Use osm, gm or gsv!}%
174   }%
175 }%
176 }%
177 \else%
178   \PackageError{getmap}{\writel8 disabled}%
179   {\writel8 disabled\MessageBreak%
180     Use -shell-escape (TeXLive)\MessageBreak%
181     or\space\space--enable-writel8 (MiKTeX)}%
182 \fi%
183 }%

```

`\getmap` Here, we define the user command to download the map.

```
\getmap[<options>]{<address>}
```

```

184 \newcommand*\getmap[2][]%
185 {%

```

We start a group to keep the setting of options local. Then we test the mode to

reset some defaults! Finally, we set the local options again to override defaults if necessary!

```

186 \begingroup%
187   \setkeys{getmap}{#1}%
188   \ifthenelse{\equal{\GM@JK@mode}{gm}}{%
189     {%
190       \def\GM@JK@scale{1}%
191       \def\GM@JK@zoom{17}%
192       \def\GM@JK@type{roadmap}%
193       \def\GM@JK@color{blue}%
194     }%
195   }%
196   \ifthenelse{\equal{\GM@JK@mode}{osm}}{%
197     {%
198       \def\GM@JK@scale{3385}%
199       \def\GM@JK@zoom{}%
200       \def\GM@JK@type{map}%
201       \def\GM@JK@color{yellow_1}%
202     }%
203   }%
204   \setkeys{getmap}{#1}%

```

In gsv mode, we have an implicit `imagetype=jpg`. Therefore, we have to set it to allow the later test on the existence of the image file!

```

205   \ifthenelse{\equal{\GM@JK@mode}{gsv}}{%
206     {\def\GM@JK@imagetype{jpg}}}%
207   \PackageInfo{getmap}{using \GM@JK@inputencoding\space encoding}%
208   \def\GM@JK@location{#2}%

```

texlua expects its arguments encoded in utf8!

```

209   \StringEncodingConvert%
210     {\GM@JK@location@string}%
211     {\detokenize\expandafter{\GM@JK@location}}%
212     {\GM@JK@inputencoding}{utf-8}%
213   \StringEncodingSuccessFailure%
214   {%
215     %success
216   }%
217   {% failure
218     \errmessage{Converting to UTF-8 failed}%
219   }%
220   \StringEncodingConvert%
221     {\GM@JK@key@string}%
222     {\detokenize\expandafter{\GM@JK@key}}%
223     {\GM@JK@inputencoding}{utf-8}%
224   \StringEncodingSuccessFailure%
225   {%
226     %success
227   }%
228   {% failure

```

```

229     \errmessage{Converting to UTF-8 failed}%
230 }%
231 \StringEncodingConvert%
232   {\GM@JK@markers@string}%
233   {\GM@JK@markers}%
234   {\GM@JK@inputencoding}{utf-8}%
235 \StringEncodingSuccessFailure%
236 {%
237   %success
238 }%
239 {% failure
240   \errmessage{Converting to UTF-8 failed}%
241 }%
242 \StringEncodingConvert%
243   {\GM@JK@path@string}%
244   {\GM@JK@path}%
245   {\GM@JK@inputencoding}{utf-8}%
246 \StringEncodingSuccessFailure%
247 {%
248   %success
249 }%
250 {% failure
251   \errmessage{Converting to UTF-8 failed}%
252 }%
253 \StringEncodingConvert%
254   {\GM@JK@visible@string}%
255   {\GM@JK@visible}%
256   {\GM@JK@inputencoding}{utf-8}%
257 \StringEncodingSuccessFailure%
258 {%
259   %success
260 }%
261 {% failure
262   \errmessage{Converting to UTF-8 failed}%
263 }%

```

We check, if overwrite is true and download the map. If not, we check if the image is already in the working directory. If not, we download the image!

```

264 \ifthenelse{equal{\GM@JK@overwrite}{true}}%
265 {%
266   \GM@JK@shellescape%
267 }%
268 {%
269   \IfFileExists{\GM@JK@file.\GM@JK@imagetype}%
270   {%
271     \PackageInfo{getmap}{overwrite=false; (\GM@JK@file.\GM@JK@imagetype)%
272       using existing file!}%
273   }%
274   {%
275     \PackageInfo{getmap}{overwrite=false; (\GM@JK@file.\GM@JK@imagetype)%
276       file does not exist! downloading ...}%

```

```
277         \GM@JK@shellescape%  
278     }%  
279 }%  
280 \endgroup%  
281 }%  
  
282 \</package>
```

7 References

- [1] Google, Inc. Encoded Polyline Algorithm Format, 2014.
<https://developers.google.com/maps/documentation/utilities/polylinealgorithm>.
- [2] Google, Inc. Google Street View Image API, 2014.
<https://developers.google.com/maps/documentation/streetview/index>.
- [3] Google, Inc. Interactive Polyline Encoder Utility, 2014.
<https://developers.google.com/maps/documentation/utilities/polylineutility>.
- [4] Google, Inc. Static Maps API V2 Developer Guide, 2014.
<https://developers.google.com/maps/documentation/staticmaps/>.
- [5] Josef Kleber. Berlin: Hbf Berlin - Brandenburger Tor (getmap Test), 2014.
<https://bitbucket.org/kleberj/getmap/downloads/Berlin.kml>.
- [6] Josef Kleber. MucBer: München Stachus -> Berlin Brandenburger Tor, 2014. <https://bitbucket.org/kleberj/getmap/downloads/MucBer.kml>.
- [7] Josef Kleber. Google Maps 'Implicit Positioning of the Map' broken?, 2016. <https://stackoverflow.com/questions/34653500/google-maps-implicit-positioning-of-the-map-broken>.
- [8] MapQuest, Inc. Compressed Lat/Lng Encoding/Decoding, 2014.
<http://open.mapquestapi.com/common/encodeddecode.html>.
- [9] MapQuest, Inc. Introducing the Data Manager API Web Service, 2014.
<http://developer.mapquest.com>.
- [10] MapQuest, Inc. MapQuest Open Platform Web Services, 2014.
<http://open.mapquestapi.com/>.
- [11] MapQuest, Inc. Static Map Service: Standard Icons, 2014.
<http://open.mapquestapi.com/staticmap/icons.html>.
- [12] MapQuest, Inc. Zoom To Scale Mapping, 2014.
<http://open.mapquestapi.com/staticmap/zoomToScale.html>.
- [13] OpenRouteService.org. Routing with user-generated, collaboratively collected free geodata., 2014. <http://openrouteservice.org>.

8 Change History

v1.0	General: CTAN upload	23	v1.6	General: getmapdl.lua: added gpx2epl, gps2epl and gpx2gps modes	23
v1.1	\getmap: Bugfix: problem in URL when using \usepackage[utf8]{inputenc}	27	v1.7	General: getmapdl.lua: added kml2gps and kml2epl modes	23
v1.10	General: getmapdl.lua: fix problem (unable to load socket.url module) caused by bug in luatex v1.0.7 (TeXLive2018)	23		getmapdl.lua: added multi track support for gpx2gps and gpx2epl mode	23
	Bugfix: fixes encoding problems with luatex v1.0.7 (TeXLive2018)	26	v1.8	General: getmapdl.lua: revised gps2epl mode	23
v1.2	General: added getmap.cfg to store default key (FR by Ulrike Fischer)	25		General: getmapdl.lua: added gps2gps mode to reduce way points with a given bound (default: 0.1)	23
	added support for Google Maps changed default values of xsize (600) and file (getmap)	24		getmapdl.lua: added check of URL length and output to log	23
	renamed osmimage.lua to getmapdl.lua	23		getmapdl.lua: added multi route support for gps2epl mode	23
v1.3	General: added support for Google Street View	23		getmapdl.lua: added rounding of gps coordinates to kml2gps and gpx2gps modes	23
v1.4	General: getmapdl.lua: added options language, markers, visible, path and pathfile in gm mode	23	v1.9	\getmap: Bugfix: restore mode defaults for scale, zoom, type and color options to guarantee correct download URLs	27
v1.5	General: added gpx2gps bash script	23		General: LuaTeX v0.85+ fix: use shellesc's \ShellEscape instead of \write18	26

Symbols

\@ifpackageloaded 77

D

\define@key 24, 28, 36, 40

\detokenize ... 38, 42, 211, 222

\directlua 123, 140, 162

G

getmap (Package) 5, 23

\getmap 184

\GM@JK@color 97, 133, 149, 193,
201\GM@JK@define@key .. 21, 45, 46,
47, 48, 49, 50, 51, 52, 53,
54, 55, 56, 57, 58, 62, 63,
64, 65\GM@JK@define@key@detok 33, 59,
60, 61\GM@JK@file 135, 157, 170, 269,
271, 275

\GM@JK@fov 168

\GM@JK@heading 167

\GM@JK@imagetype 132, 148, 206,
269, 271, 275\GM@JK@inputencoding .. 79, 82,
207, 212, 223, 234, 245,
256

\GM@JK@key 67, 74, 222

\GM@JK@key@string ... 126, 221

\GM@JK@language 151

\GM@JK@location 208, 211

\GM@JK@location@string .. 124,
141, 153, 163, 210

\GM@JK@markers 233

\GM@JK@markers@string 152, 232

\GM@JK@mode . 92, 121, 138, 160,
188, 196, 205

\GM@JK@number 134, 150

\GM@JK@overwrite 66, 264

\GM@JK@path 244

\GM@JK@path@string .. 154, 243

\GM@JK@pathfile 155

\GM@JK@pitch 169

\GM@JK@scale 94, 130, 146, 190,
198\GM@JK@shellescape .. 118, 266,
277\GM@JK@type . 96, 131, 147, 192,
200

\GM@JK@visible 255

\GM@JK@visible@string 156, 254

\GM@JK@writexviiifalse 85, 114

\GM@JK@writexviiiitruer 106, 112

\GM@JK@xsize 127, 143, 165

\GM@JK@ysize 128, 144, 166

\GM@JK@zoom . 95, 129, 145, 191,
199**I**

\IfFileExists 69, 269

\ifGM@JK@writexviii .. 85, 120

\ifluatex 15, 123, 140, 162

\ifxetex 9

\input 71

inputenc (Package) 5, 25

\inputencodingname 79

L

\ltx@ifUndefined 102

O

overwrite (Style option) . 25, 29

P

Package

getmap 5, 23

inputenc 5, 25

shellesc 23

\pdf@shellescape 104, 109, 110

S

\setkeys 187, 204

shellesc (Package) 23

\ShellEscape 124, 141, 163

\StringEncodingConvert 10, 16,
209, 220, 231, 242, 253\StringEncodingSuccessFailure
213, 224, 235, 246, 257

Style option

overwrite 25, 29