The Deviation of China Map as a Regression Problem

Wu Yongzheng

1 Introduction

2 Notation

We need to find out two functions, $f_x(x, y)$ and $f_y(x, y)$, to describe the deviation. x and y are the true longitude and latitude respectively. $f_x(x, y)$ and $f_y(x, y)$ are the deviation delta at point (x, y). In other words, if (x, y) is the true coordinate, then $(x+f_x(x, y), y+f_y(x, y))$ is the deviated coordinate. Since $|f_x(x + d, y) - f_x(x, y)|$ is much smaller than d (and same for y) we can also use the two functions to calculate the true coordinate from a deviated coordinate. Let (x, y) be the deviated coordinate, $(x - f_x(x, y), y - f_y(x, y))$ is the true coordinate.

In the following analysis, we will use octave. All the data used in the analysis has already been posted in my previous article[1].

3 The four 1D cases

We first solve the functions in one dimension by setting one parameter to be constant. I.e. instead of solving $f_x(x, y)$ directly, we first solve $f_x(x, c)$ and $f_x(c, y)$, where c is a constant. For example, we will look at $f_x(x, 39.36)$ and $f_x(110.16, y)$. Similarly, we will solve $f_y(x, c)$ and $f_y(c, y)$ before $f_y(x, y)$. In total, there are four 1D cases.

3.1 $f_x(const, y)$

$f_x(const, y)$ looks like to be the simplest, so we solve it first.

```
> data=load("-ascii", "iny-f-110.txt");
> y=data(:,2);
> dx=data(:,3);
> plot(y,dx)
```

Figure 1(left)

Figure 1(left) shows the plot, $f_x(110.16, y)$. The curve looks like a second order curve, but if we go that direction, it's going to be very tough. It is a lot simpler if we measure the deviation in meters rather than degrees. Note that earth's radius is 6378137 meters in Google Map. I didn't know this until the experiment was all finished. However, this shouldn't cause any error even if I set

[1] http://wuyongzheng.wordpress.com/2010/01/22/china-map-deviation-as-a-regression-problem/

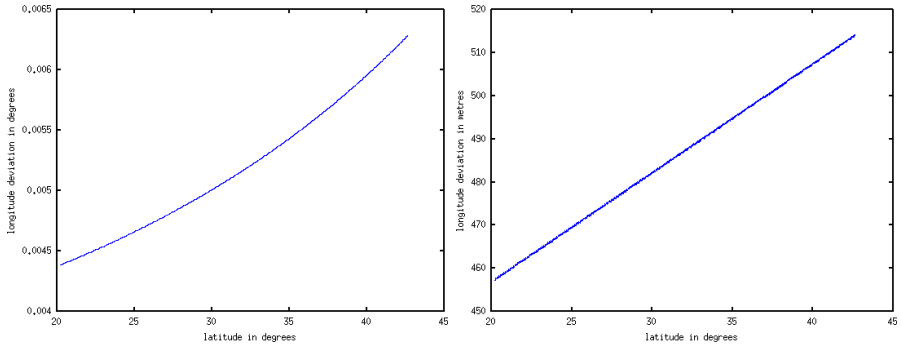

Figure 1: $f_x(110.16, y)$. Left: longitude deviation (y-axis) is in degrees; right: in metres.

the radius to 1 meter, because in the end, meters is converted back to degrees and the error cancels.

```
> plot(y,dx.*cos(y./180.*pi).*6371000.*pi./180)
```
Figure 1(right)

Figure 1(right) shows the plot where the deviation is measured in meters. It's a linear function, thus we can do a simple linear regression.

```
> dx=dx.*cos(y./180.*pi).*6371000.*pi./180;
> mx=[ones(size(y)),y]; std(mx*(pinv(mx)*dx)-dx)
ans =  0.14664
> pinv(mx)*dx
ans =
   406.3954
     2.5241
```

The standard error, 0.14664 meter, is acceptable, because the input data's resolution is about 0.6 meter[2] on average. The final function is

$$f_x(110.16, y) = 406.3954 + 2.5241y \tag{1}$$

After trying out other constant longitudes, we find out that the two coefficients are different for different longitude. We have

$$f_x(x, y) = g_1(x) + g_2(x) \times y \tag{2}$$

where $g_1(x)$ and $g_2(x)$ are not yet known.

In the following analysis, we will measure x and y in degrees, while measure $f_x(x, y)$ and $f_y(x, y)$ in meters. It makes our life a lot easier.

3.2 $f_y(x, const)$

$f_y(x, const)$ seems to be the second easiest one. It's quite obvious that there are sinusoidal and linear components in the function. To find out the frequency of the sinusoidal components, we do a Fourier transform.

[2]The input is converted from google map's unit. One unit is one pixel under the maximum zoom setting. Due to Mercator projection, higher latitude has higher resolution. The average 0.6 meter is computed based on the mean coordinate, which is near Xi'an city.

```
> data=load("-ascii", "inx-f-39.txt");
> x=data(:,1);
> size(x)
ans =
   9116      1
> x(1)
ans =  73.655
> x(9116)
ans =  119.23
> dy=data(:,4).*(6371000*pi/180);
```

```
> plot(x,dy)                                   Figure 2(left)
> plot(abs(fft(dy)))                           Figure 2(right)
```

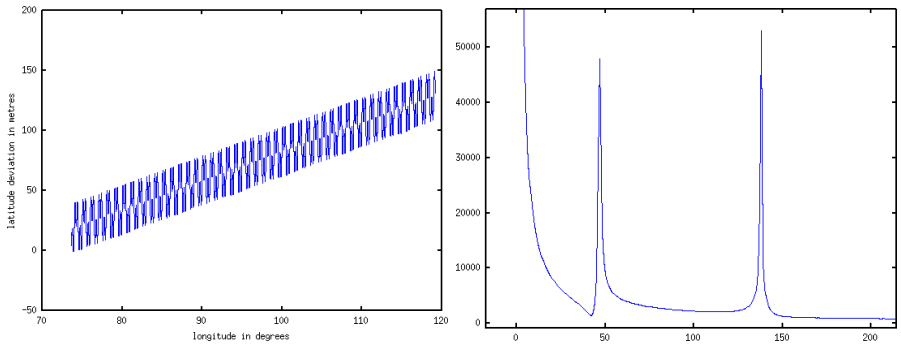

Figure 2: $f_y(x, 39.36)$. Left: the data points; right: FFT of the data

Figure 2(left) shows the data and Figure 2(right) shows the data in frequency domain. The two peaks in the frequency domains are at 47 and 138, thus the frequencies are 46/45.575=1.0093 and 137/45.575=3.0060, where 45.575=119.23-73.655 is the range of x. Since the range of x is not multiple of the periods, we don't get exact results, but after cutting the range, the exact results are 1 and 3. We then try to model it using $b + kx + a_1 sin(2\pi x) + a_2 cos(2\pi x) + a_3 cos(6\pi x) + a_4 cos(6\pi x)$.

```
> mx=[ones(size(x)),x,sin(2.*pi.*x),cos(2.*pi.*x),sin(6.*pi.*x),\
  cos(6.*pi.*x)];
> std(mx*(pinv(mx)*dy)-dy)
ans =  0.38114
```

```
> plot(x,dy,x,dy-mx*(pinv(mx)*dy))                Figure 3
```

The standard error, 0.38114, is acceptable. We can also see this in Figure 3. The regression error (difference between the data and our function) is close to zero. Let's see the coefficients:

```
> pinv(mx)*dy
ans =
  -160.45195
     2.42764
```

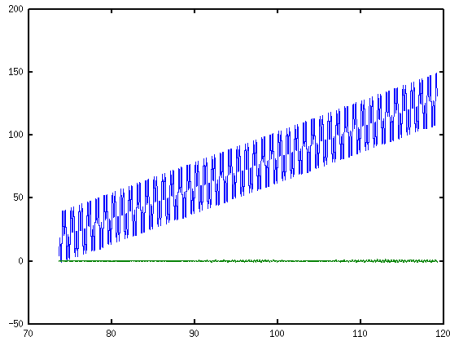

Figure 3: The data (blue) and the regression error (green).

```
   13.33287
   -0.27150
   13.30099
   -0.82075
```

We can combine sin and cos of the same frequency.

```
> atan(-0.82075/13.30099)
ans =  -0.061628
> atan(-0.27150/13.33287)
ans =  -0.020360
> mx=[ones(size(x)),x,sin(2.*pi.*x-0.020360),sin(6.*pi.*x-0.061628)];
> std(mx*(pinv(mx)*dy)-dy)
ans =  0.38114
> pinv(mx)*dy
ans =
  -160.4519
     2.4276
    13.3356
    13.3263
```

So the function is

$$\begin{aligned} f_y(x, 39.36) &= -160.45195 + 2.42764x + 13.33287sin(2\pi x) \\ &\quad -0.27150cos(2\pi x) + 13.30099cos(6\pi x) - 0.82075cos(6\pi x) \\ &= -160.45195 + 2.42764x + 13.3356sin(2\pi x - 0.061628) \\ &\quad +13.3263cos(6\pi x - 0.020360) \end{aligned} \tag{3}$$

After trying out other constant latitudes, we find out that the first two coefficients to be changing but the last 4 coefficients to be fixed. Thus, we know

$$\begin{aligned} f_y(x, y) &= g_1(y) + g_2(y)x + 13.3356sin(2\pi x - 0.061628) \\ &\quad +13.3263cos(6\pi x - 0.020360) \end{aligned} \tag{4}$$

where $g_1(y)$ and $g_2(y)$ are not yet known.

3.3 $f_x(x, const)$

Similar to previous one, we use Fourier transform to find out the sinusoidal components first.

```
> data=load("-ascii", "inx-f-39.txt");
> x=data(:,1);
> dx=data(:,3).*cos(39.36./180.*pi).*6371000.*pi./180;

> plot(x,dx)                                    Figure 4(left)
> plot(abs(fft(dx)))                            Figure 4(right)
```

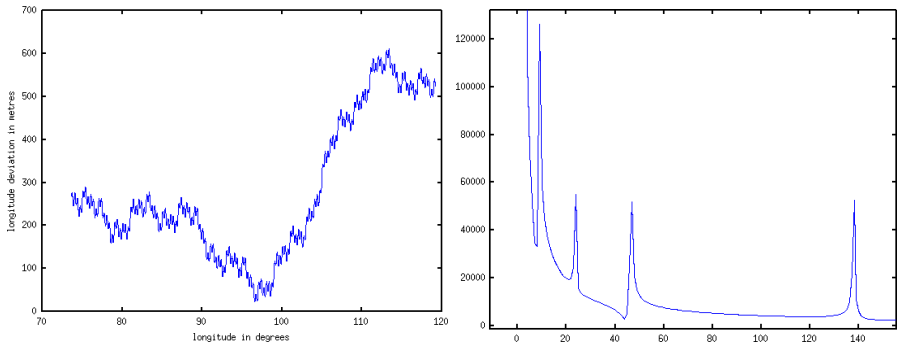

Figure 4: $f_x(x, 39.36)$. Left: the data points; right: FFT of the data

Figure 4(left) shows the curve that we are going to fit. Figure 4(right) shows the frequency domain. The peaks are at 9, 24, 47 and 138, so the frequencies are 8/45.575=0.17553, 23/45.575=0.50466, 46/45.575=1.0093 and 137/45.575=3.0060. The exact frequencies are 1/6, 1/2, 1 and 3. Now, we try to fit the curve with the 4 sinusoidal components and a linear component.

```
> mx=[ones(size(x)),x,sin(pi.*x./3),cos(pi.*x./3),sin(pi.*x),\
  cos((pi.*x)),sin(2.*pi.*x),cos(2.*pi.*x),sin(6.*pi.*x),\
  cos((6.*pi.*x))]; std(mx*(pinv(mx)*dx)-dx)
ans =  119.96
> plot(x,dx,x,dx-mx*(pinv(mx)*dx))
```

The standard error 119.96 meters is not acceptable. To see it, we plot the difference in Figure 5. It seems that there are some low frequency components. We can deal with low frequency components with a bunch of polynomials.

```
> mx=[ones(size(x)),(x./mean(x)),(x./mean(x)).^2,(x./mean(x)).^3,\
  (x./mean(x)).^4,(x./mean(x)).^5,(x./mean(x)).^6,(x./mean(x)).^7,\
  (x./mean(x)).^8,(x./mean(x)).^9,sin(pi.*x./3),cos(pi.*x./3),\
  sin(pi.*x),cos((pi.*x)),sin(2.*pi.*x),cos(2.*pi.*x),sin(6.*pi.*x),\
  cos((6.*pi.*x))]; std(mx*(pinv(mx)*dx)-dx)
ans =  0.78555
```

It fits well with a 9-order polynomial. Note that we divide x by $mean(x)$ to make sure the values in the matrix have low dynamic range so as to reduce

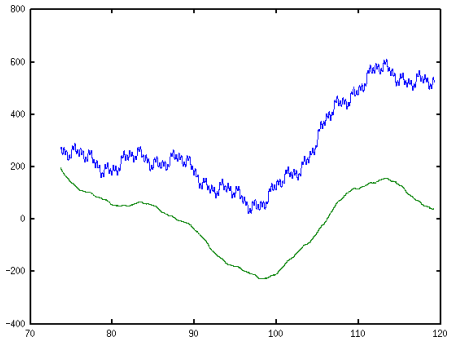

Figure 5: The data (blue) and the regression error (green).

numeric error. Mathematically, they don't make a difference. To see how our polynomial works, we can plot the polynomial and compare with the data (see Figure 6).

```
> coeff=pinv(mx)*dx;
> coeff(11:18)=0
coeff =
   1.3529e+08
  -1.3862e+09
   6.2344e+09
  -1.6167e+10
   2.6655e+10
  -2.8993e+10
   2.0816e+10
  -9.5166e+09
   2.5150e+09
  -2.9286e+08
   0.0000e+00
   0.0000e+00
   0.0000e+00
   0.0000e+00
   0.0000e+00
   0.0000e+00
   0.0000e+00
   0.0000e+00
> plot(x,dx,x,mx*coeff)
```

Although the 9-order polynomial fits the data well (error is 0.79 meters), the function is ugly. After lots of tries[3], I figured out that the low frequency component actually consists of a quadratic curve and two sinusoidal wave of period 60 and 24. Using this model, the regression error is 0.39 meters and the function is simple and beautiful.

[3]Most of the time has been spent here, but there is nothing to say about it because it's simply trying different functions. There are still a little bit of technique though. For example, the two green peaks in Figure 6 have distance about 24 in x-axis, thus there might be a sinusoidal component of period 24.

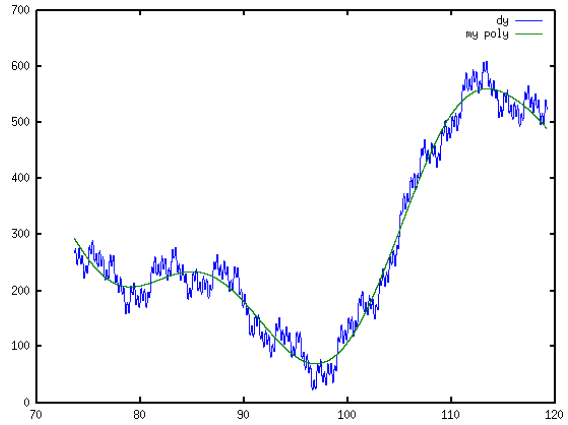

Figure 6: 9-order polynomial (green) and 9-order polynomial + sinusoidal (blue)

```
> mx=[ones(size(x)),x,x.*x,sin(pi.*x./30),cos(pi.*x./30),\
  sin(pi.*x./12),cos(pi.*x./12),sin(pi.*x./3),cos(pi.*x./3),\
  sin(pi.*x),cos((pi.*x)),sin(2.*pi.*x),cos(2.*pi.*x),sin(6.*pi.*x),\
  cos((6.*pi.*x))]; std(mx*(pinv(mx)*dx)-dx)
ans =  0.38668
> pinv(mx)*dx
ans =
   1.2052e+03
  -1.8404e+01
   9.4022e-02
   8.3824e-01
   2.0044e+02
  -7.0435e+01
  -7.0449e+01
  -2.6699e+01
   9.1218e-02
  -1.3342e+01
   1.3044e-01
   1.3342e+01
  -2.7504e-01
   1.3306e+01
  -8.2298e-01
```

Now let's merge sin and cos.

```
> atan(2.0044e+02/8.3824e-01)
ans =  1.5666
> atan(-7.0449e+01/-7.0435e+01)
ans =  0.78550
> atan(9.1218e-02/-2.6699e+01)
ans =  -0.0034165
> atan(1.3044e-01/-1.3342e+01)
ans =  -0.0097763
```

```
> atan(-2.7504e-01/1.3342e+01)
ans =  -0.020612
> atan(-8.2298e-01/1.3306e+01)
ans =  -0.061772
> mx=[ones(size(x)),x,x.*x,sin(pi.*x./30+1.5666),sin(pi.*x./12+0.78550),\
  sin(pi.*x./3-0.0034165),sin(pi.*x-0.0097763),sin(2.*pi.*x-0.020612),\
  sin(6.*pi.*x-0.061772)]; std(mx*(pinv(mx)*dx)-dx)
ans =  0.38668
> pinv(mx)*dx
ans =
   1.2051e+03
  -1.8402e+01
   9.4012e-02
   2.0045e+02
  -9.9620e+01
  -2.6699e+01
  -1.3343e+01
   1.3344e+01
   1.3332e+01
```

Finally, we have

$$\begin{aligned} f_x(x, 39.36) &= 1205.1 - 18.402x + 0.094012x^2 + 200.45sin(\pi x/30 + 1.5666) \\ &\quad -99.62sin(\pi x/12 + 0.78550) - 26.699sin(\pi x/3 - 0.0034165) \\ &\quad -13.343sin(\pi x - 0.0097763) + 13.344sin(2\pi x - 0.020612) \\ &\quad +13.332sin(6\pi x - 0.061772) \end{aligned} \tag{5}$$

To see how the components fit the data, we can graduately add the higher frequency components. Figure 5.5 shows this this idea. Figure 5.6 is a zoomed in region in Figure 5.5. The first function (in red) is the quadratic function $1205.1 - 18.402x + 0.094012x^2$. The second function (in green) adds one sinusoidal component to it: $1205.1 - 18.402x + 0.094012x^2 + 200.45sin(\pi x/30 + 1.5666)$. The third function (in blue) adds one more sinusoidal component to it, etc.

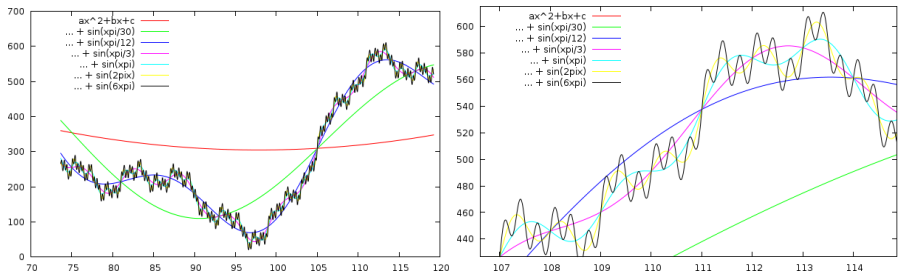

Figure 7: Decomposition of Components. Left: overall; right: zoomed in. The first function (red) has only the lowest frequency component. The second function (green) adds the second lowest frequency component to the first function. The third (blue) adds another component, etc.

After trying out other constant latitude, we found that only the first two coefficients are affected by latitude. The rest of the coefficients do not change. Thus, we have

$$\begin{aligned} f_x(x,y) &= g_1(y) - g_2(y)x + 0.094012x^2 + 200.45sin(\pi x/30 + 1.5666) \\ &\quad -99.62sin(\pi x/12 + 0.78550) - 26.699sin(\pi x/3 - 0.0034165) \\ &\quad -13.343sin(\pi x - 0.0097763) + 13.344sin(2\pi x - 0.020612) \\ &\quad +13.332sin(6\pi x - 0.061772) \end{aligned} \tag{6}$$

where $g_1(y)$ and $g_2(y)$ are not yet known.

3.4 $f_y(const, y)$

The process of finding $f_y(const, y)$ is similar to $f_x(x, const)$, except that the two high frequency sinusoidal components do not appear in $f_y(const, y)$.

```
> data=load("-ascii", "iny-a-110.txt");
> y=data(:,2);
> dy=data(:,4).*(6371000*pi/180);
> my=[ones(size(y)),y,y.*y,sin(pi.*y./30),cos(pi.*y./30),\
  sin(pi.*y./12),cos(pi.*y./12),sin(pi.*y./3),cos(pi.*y./3),\
  sin(pi.*y),cos((pi.*y))]; std(my*(pinv(my)*dy)-dy)
ans =    0.16683
> pinv(my)*dy
ans =
       162.1
     -22.422
     0.41569
     -172.78
      56.189
     -104.96
     -29.351
      13.251
      23.113
     -13.357
   -0.061311
> atan(56.189/-172.78)
ans =   -0.31442
> atan(-29.351/-104.96)
ans =    0.27267
> atan(23.113/13.251)
ans =     1.0502
> atan(-0.061311/-13.357)
ans =  0.0045901
> my=[ones(size(y)),y,y.*y,sin(pi.*y./30-0.31442),\
  sin(pi.*y./12+0.27267),sin(pi.*y./3+1.0502),\
  sin(pi.*y+0.0045901)]; std(my*(pinv(my)*dy)-dy)
ans =    0.16683
> pinv(my)*dy
ans =
```

```
 162.09
-22.422
0.41568
-181.69
-108.99
 26.643
-13.357
```

We have

$$\begin{aligned} f_y(110.16, y) &= 162.09 - 22.422y + 0.41568y^2 \\ &\quad -181.69sin(\pi y/30 - 0.31442) - 108.99sin(\pi y/12 + 0.27267) \\ &\quad +26.643sin(\pi y/3 + 1.05) - 13.357sin(\pi y - 0.00459) \qquad (7) \\ f_y(x, y) &= g_1(x) + g_2(x)y + 0.41568y^2 \\ &\quad -181.69sin(\pi y/30 - 0.31442) - 108.99sin(\pi y/12 + 0.27267) \\ &\quad +26.643sin(\pi y/3 + 1.05) - 13.357sin(\pi y - 0.00459) \qquad (8) \end{aligned}$$

4 The final 2D case: $f_x(x, y)$ and $f_y(x, y)$

From Equation 6 and 2 we know that $f_x(x, y)$ must be in the form of Equation 9. Similarly, from Equation 4 and 8 we know $f_y(x, y)$ must be in the form of Equation 10. What's left now is to determine the parameters.

$$\begin{aligned} f_x(x, y) &= a_1 + a_2x + a_3y + a_4xy + a_5x^2 \\ &\quad +a_6sin(\pi x/30 + b_1) + a_7sin(\pi x/12 + b_2) + a_8sin(\pi x/3 + b_3) \\ &\quad +a_9sin(\pi x + b_4) + a_{10}sin(2\pi x + b_5) + a_{11}sin(6\pi x + b_6) \qquad (9) \\ f_y(x, y) &= c_1 + c_2x + c_3y + c_4xy + c_5y^2 \\ &\quad +c_6sin(\pi y/30 + d_1) + c_7sin(\pi y/12 + d_2) + c_8sin(\pi y/3 + d_3) \\ &\quad +c_9sin(\pi y + d_4) + c_{10}sin(2\pi x + d_5) + c_{11}sin(6\pi x + d_6) \qquad (10) \end{aligned}$$

Applying the previous linear regression method on the data file `google-grid40.tsv`, we can determine all the parameters easily. The octave commands are skipped. The resulting equations are shown below and their standard errors are 0.4373 and 0.43998 meters respectively, which are larger than the four 1D cases, but still below the input data's resolution.

$$\begin{aligned} f_x(x, y) &= 1596.7 - 23.578x - 8.431y + 0.0993xy + 0.10056x^2 \\ &\quad -199.11sin(\pi x/30 - 1.5692) - 99.77sin(\pi x/12 + 0.78662) \\ &\quad -26.687sin(\pi x/3 - 0.0040926) - 13.344sin(\pi x - 0.012222) \\ &\quad +13.338sin(2\pi x - 0.024649) + 13.331sin(6\pi x - 0.073931) \qquad (11) \\ f_y(x, y) &= 42.304 - 1.522x - 10.376y + 0.10032xy + 0.041777y^2 \\ &\quad -217.35sin(\pi y/30 - 0.71213) - 106.38sin(\pi y/12 + 0.28733) \\ &\quad +26.698sin(\pi y/3 + 1.0481) - 13.335sin(\pi y + 0.00012317) \\ &\quad +13.329sin(2\pi x - 0.024639) + 13.319sin(6\pi x - 0.073933) \qquad (12) \end{aligned}$$

5 Evaluation

To check their correctness, we can compute a deviated coordinate and check whether it aligns with the deviated map. We now compute the deviated coordinate of the Monument to the People's Heroes using its true coordinate $(116.391667, 39.903056)$.[4] $f_x(116.391667, 39.903056) = 531.91$ meters and $f_y(116.391667, 39.903056) = 155.39$ meters. This means that the deviated coordinate is 531.91 meters to the east and 155.39 to the north. We then convert meters into degrees and adds them to the true coordinate.

$$\begin{aligned} x' &= 116.391667 + 531.91 \times 360/(2\pi \times 6371000 \times cos(\frac{2\pi}{360} \times 39.903056)) \\ &= 116.397903 \\ y' &= 39.903056 + 155.39 \times 360/(2\pi \times 6371000) \\ &= 39.904453 \end{aligned}$$

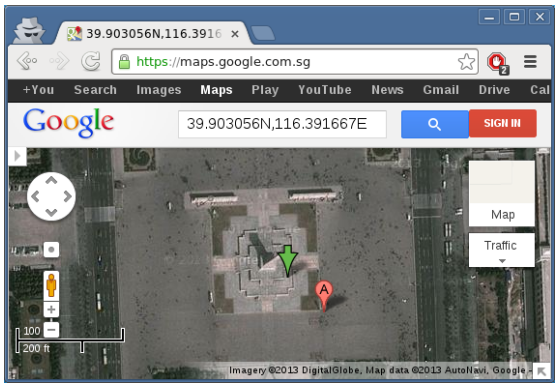

Figure 8: The monument's true coordinate in GMap's satellite image

The green arrow in Figure 8 shows the true coordinate in the Google Maps' satellite image. Figure 9 shows the deviated coordinate in the Google Maps' street map. The red balloon, which is the nearest landmark, should be ignored. The green arrows in both maps are slightly to the southeast of the monument center, but as long as they offset the same amount, it is as good.

[4] I cannot find any authoritative location of the monument. The one that I use is from Wikipedia.

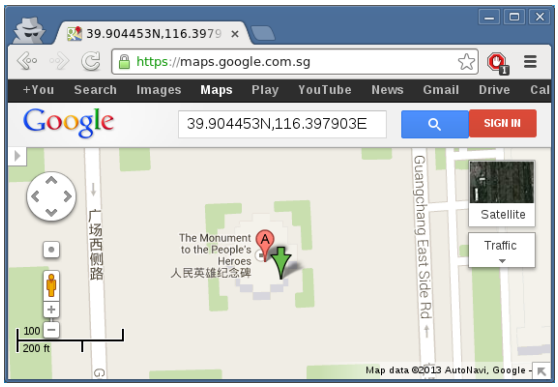

Figure 9: The monument's deviated coordinate in GMap's street map