Question 1: Abridged Molodensky and indirect three-parameter geocentric transformation

$\Phi = 51.95817$

$\lambda = 7.617608$

$h = 65.48196$

A) ABRIDGED MOLODENSKY:

R code:

```
#parameters for source ellipsoid:

phisec = 51*3600 + 57*60 + 24.7

lambdasec = 7*3600 + 37*60 + 4.8

phi = 51.95686*pi/180

lambda = 7.618*pi/180

h = 57

as = 6378137

fs = 1/298.257223563

es = sqrt(2*fs - fs*fs)

rhos = (as*(1 - es*es))/((1 - es*es*sin(phi)*sin(phi))^(3/2))

nus = as/(sqrt(1 - es*es*sin(phi)*sin(phi)))


#ellipsoid centres shift:

dx = -582

dy = -105

dz = -414


#target ellipsoid:

at = 6377397

ft = 1/299.152815351

da = at - as

df = ft - fs
```

```
#changes in latitude, longitude and ellipsoidal height to be added to source coordinates (Abridged
Molodensky):

dphi = (-dx*sin(phi)*cos(lambda) - dy*sin(phi)*sin(lambda) + dz*cos(phi) + (as*df +
fs*da)*sin(2*phi))/(rhos*sin((1/3600)*pi/180))

dlambda = (-dx*sin(lambda) + dy*cos(lambda))/(nus*cos(phi)*sin((1/3600)*pi/180))

dh = dx*cos(phi)*cos(lambda) + dy*cos(phi)*sin(lambda) + dz*sin(phi) + (as*df +
fs*da)*sin(phi)*sin(phi) - da


#calculation of target coordinates:

S = matrix(c(phisec, lambdasec, h), 3)

DS = matrix(c(dphi, dlambda, dh), 3)

T = S + DS

T


#conversion of target coordinates into degrees:

tphi = T[1,1] / 3600

tlambda = T[2, 1] / 3600

th = T[3, 1]


tphi

tlambda

th
```

Results:

```
> tphi

[1] 51.95817 = 51° 57' 29.412''

> tlambda

[1] 7.617608 = 7° 37' 3.3882''

> th
```

```
[1] 65.63652 meters
```

B) INDIRECT THREE-PARAMETER GEOCENTRIC TRANSFORMATION:

R code:

```
#1) conversion from source ellipsoid to source geocentric:

x = (nus + h)*cos(phi)*cos(lambda)

y = (nus + h)*cos(phi)*sin(lambda)

z = ((1 - es*es)*nus + h)*sin(phi)


#2) transformation from source geocentric to target geocentric:

SG = matrix(c(x, y, z), 3)

DSG = matrix(c(dx, dy, dz), 3)

TG = SG + DSG


#3) conversion from target geocentric to target ellipsoid:

bt = at - ft*at

et = sqrt(2*ft - ft*ft)

epsilont = (et*et)/(1 - et*et)

pt = sqrt(TG[1,1]^2 + TG[2,1]^2)

ut = atan((TG[3,1]*at)/(pt*bt))


lambdat = atan(TG[2,1]/TG[1,1])

phit = atan((TG[3,1] + epsilont*bt*sin(ut)*sin(ut)*sin(ut))/(pt - et*et*at*cos(ut)*cos(ut)*cos(ut)))

nut = at/(sqrt(1 - et*et*sin(phit)*sin(phit)))

ht = pt*(1/cos(phit)) - nut


#conversion of target coordinates into degrees:

phit = phit * 180/pi
```

```
lambdat = lambdat * 180/pi


phit

lambdat

ht
```

Results:

```
> phit

[1] 51.95817 = 51° 57' 29.412"

> lambdat

[1] 7.617608 = 7° 37' 3.3882"

> ht

[1] 65.69447 meters
```

Question 2

Possibilities:

- All values are only an approximation because of the several assumptions used for the calculations.
- The online service might as well use different datum shifts.
- Abridged Molodensky, being a simplified version, could make errors related to approximations bigger.

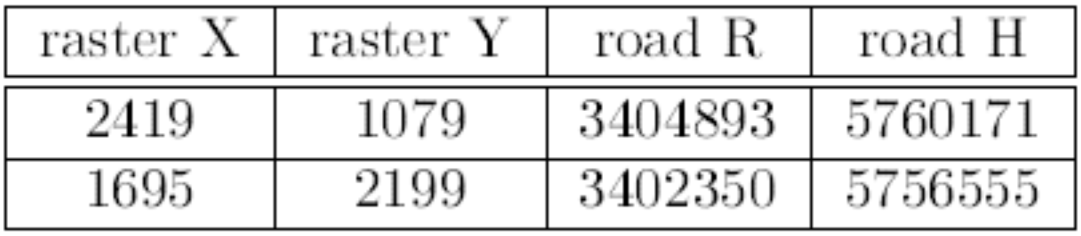
Question 3

Note: Different image processing programs give different kinds of coordinates. For example, the Windows 7 version of MS Paint will show coordinates in cm, whereas the Windows XP version of MS Paint will give coordinates in pixels. For both versions, the origin will be in the upper left corner. When using GIMP you'll get the same as for Windows XP Paint. Using Inkscape, you'll also get pixel coordinates but with the origin in the lower left corner.

Solution (using pixel coordinates with the origin in the upper left corner): The origin of the reference system of Muenster.tiff is located in the upper left corner, whereas the origin of the ms_strassen.shp is in the lower left corner. This results in decreasing values for Y in the muenster.tiff and increasing values for Y in the ms_strassen.shp from south to north. The problem is that simple transformations such as the Similarity Transformation cannot flip the coordinates horizontally, but can only scale, translate and rotate them.

To fix this, the y-axis of the target system should be “mirrored” before transformation. We can do this by subtracting the y coordinate value from the maximum number value of y which is 3438 in this case.

Question 4: Similarity Transformation



raster X	raster Y	road R	road H
2419	1079	3404893	5760171
1695	2199	3402350	5756555

R Script:

```
#Ground Control Points
point1<-c(3404893,5760171,2419,1079)
point2<-c(3402350,5756555,1695,2199)
rasterpoints<-c(2419,1079,1695,2199)

#Create Matrix from control points
A<-matrix(c(point1[1],point1[2],1,0,point1[2],-point1[1],0,1,point2[1],point2[2],1,0,point2[2],-
point2[1],0,1),nrow=4,ncol=4,byrow=T)

A

#parameter-vector
b<-solve(A,rasterpoints)
b
```

R Output:

```
> A
        [,1]     [,2] [,3] [,4]
[1,] 3404893  5760171    1    0
[2,] 5760171 -3404893    0    1
[3,] 3402350  5756555    1    0
[4,] 5756555 -3402350    0    1

> b
[1] -1.130260e-01  2.797083e-01 -1.223907e+06  1.604505e+06
```

………

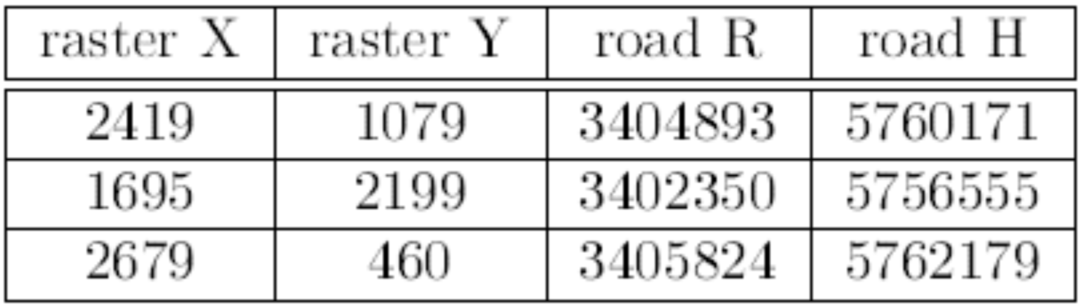
Therefore the four parameters of the equations

xT = c + a*XS + b*YS
yT = d - b*XS + a*YS

are:

a = -1.130260e-01
b = 2.797083e-01
c = -1.223907e+06
d = 1.604505e+06

Question 5



raster X	raster Y	road R	road H
2419	1079	3404893	5760171
1695	2199	3402350	5756555
2679	460	3405824	5762179

R Script:

```
#Ground Control Points
 point1<-c(3404893,5760171,2419,1079)
 point2<-c(3402350,5756555,1695,2199)
 point3<-c(3405824,5762179,2679,460)
 rasterpoints<-c(2419,1079,1695,2199,2679,460)

 #Create Matrix from control points
 A<-matrix(c(1,point1[1],point1[2],0,0,0,0,0,0,1,point1[1],point1[2],1,
point2[1],point2[2],0,0,0,0,0,0,1,point2[1],point2[2],1,
point3[1],point3[2],0,0,0,0,0,0,1,point3[1],point3[2]),nrow=6,ncol=6,byrow=T)

#parameter-vector
 b<-solve(A,rasterpoints)
 b
```

R Output:

```
> A
A [,1] [,2] [,3] [,4] [,5] [,6]
[1,] 1 3404893 5760171 0 0 0
[2,] 0 0 0 1 3404893 5760171
[3,] 1 3402350 5756555 0 0 0
[4,] 0 0 0 1 3402350 5756555
[5,] 1 3405824 5762179 0 0 0
[6,] 0 0 0 1 3405824 5762179

> b
[1] -9.601727e+05 2.952166e-01 -7.393749e-03
```

```
1.781246e+06 -6.124673e-03 [6] -3.054273e-01
```

………

Therefore the six parameters of the equations

xT = a0 + a1*XS + a2*YS
yT = b0 + b1*XS + b2*YS

The six resulting parameters of the equation are:
a0 : -9.601727e + 05
a1 : 2.952166e - 01
a2 : -7.393749e - 03
b0 : 1.781246e + 06
b1 : -6.124673e - 03
b2 : -3.054273e - 01

Question 6:

We first create a test equation for the similarity transformation, using results stored in matrix b (see Question 4 solution):

R code:
```
testpoint<-c(3405654,5758875)
calcs<-matrix(c(testpoint[1],testpoint[2],(b[3]+(b[1]*testpoint[1])+(b[2]*testpoint[2])),(b[4]-
(b[2]*testpoint[1])+(b[1]*testpoint[2])) ),nrow=2,ncol=2,byrow=T)
calcs
```

Result:
```
calcs
[,1] [,2]
[1,] 3405654.0000000000000 5758875.0000000000000
[2,] 1694.2667269203905 2199.2964707568899
```

………

The results obtained from the transformation are
X: 1694.267
Y: 2199.296
whereas the ‘true’ coordinates are
X: 2668
Y: 1458

We then do the same for the affine transformation:

R code:
```
testpoint<-c(3405654,5758875)

calcs<-matrix(c(testpoint[1],testpoint[2],b[1]+(b[2]*testpoint[1])
+(b[3]*testpoint[2]),b[4]+(b[5]*testpoint[1])+(b[6]*testpoint[2])),
nrow=2,ncol=2,byrow=T)
calcs
```

Result:

```
calcs
#[,1] [,2]
#[1,] 3405654.000 5758875.000
#[2,] 2653.242 1470.173
```

………

The results obtained from the transformation are
X: 2653
Y: 1470
whereas the ‘correct’ coordinates should be
X: 2668
Y: 1458

The affine transformation calculated y co-ordinates much more accurately than the similarity transformation. This may be because affine transformations take into account scale and rotation adjustments better whereas similarity transformations assume uniform scale along both axes and that the axes are orthogonal. As different scale factors and different angles between the coordinate axis can be present (shearing), the affine transformation is more 'fittable'. It is possible that both transformations are more accurate for one point (point A) than they are for another (Point B). This could be because point A was located more closely to the ground control points used to calculate the transformation parameters.