Tracking public transportation vehicles by cellular network signals

http://svn.auditory.ru/repos/tatmon/

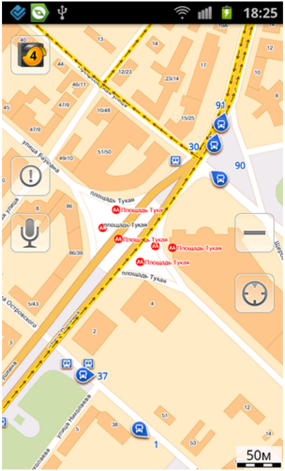
By Maxim Kovalev
maxim.kovalev@sv.cmu.edu

Carnegie Mellon University, PhD sutudent

August 2013

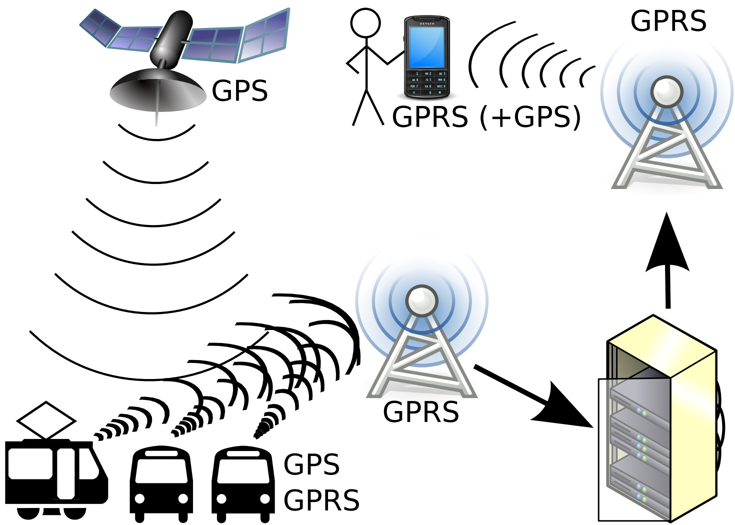
The goals of the project

- **Overall** — make easier traveling within cities!
- Create a cheaper tracking system without decreasing accuracy;
- Eliminate satellite navigation;
- Increase the accuracy of GSM cell-bases localization.
  - Test the new method.



Buses of the city of Kazan in real time.

The general architecture



Existing positioning methods

Satellite navigation

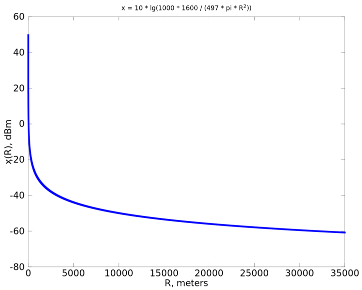
- Satellites
- Based on signal delay
- Very precise, simple formulas
- 1 – 50 meters of error

Cellular

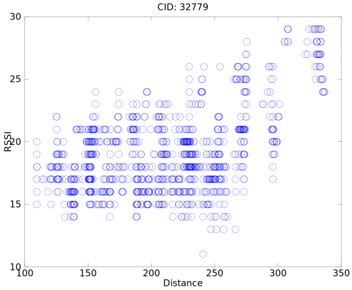
- GSM cells
- Based on signal levels
- Not precise, sophisticated methods
- 10 – 500 meters of error

Hypothesis

Over-extrapolation affects the precision of cell-based positioning



Theoretical estimate of signal attenuation



Experimental data

Public transportation

A predefined route:

- Makes the problem one-dimensional
- Allows us to obtain more statistical data

Existing statistical methods

Parameter	Mahalanobis distance	Bayesian classifier
Continuousness of an argument	-	-
Continuousness of a value	+	-
Resistance to anomalies	-	+

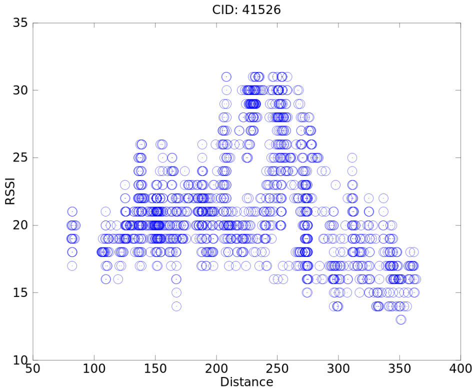
My algorithm

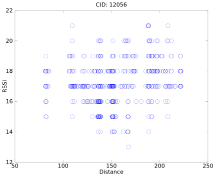
The algorithm being proposed:

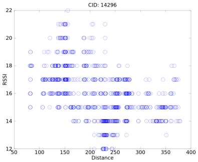
1. Considers the continuousness of a random value – a signal level;
2. Considers data from adjacent points;
3. Is resistant to anomalies.

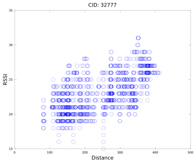
How does it work?

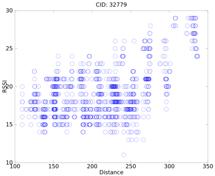
Selecting from a datbase



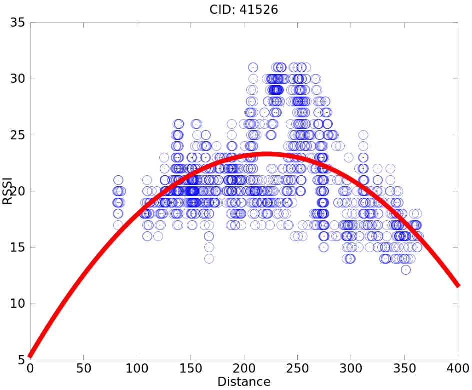


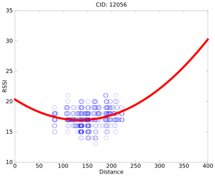


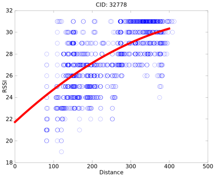


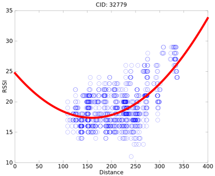


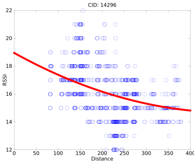
Approximation

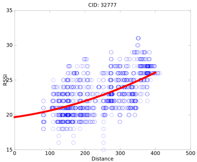












Pseudo-probability density

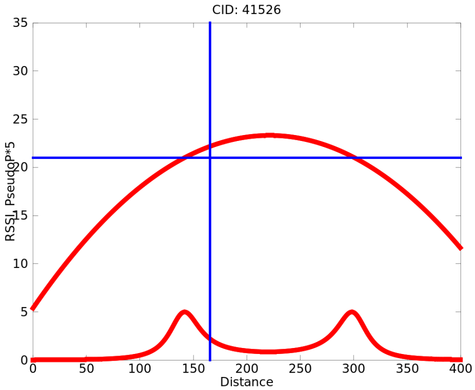
Requirements:

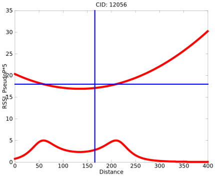
1. $\forall_{f(x),y} P(f(x), y) \in (0, 1]$
2. $\forall_{f(x),y} f(x) = y \Leftrightarrow P(f(x), y) = 1$
3. $\lim_{|f(x)-y|\to\infty} P(f(x), y) = 0$

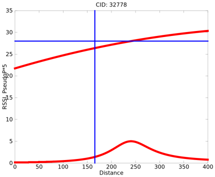
Implementation:

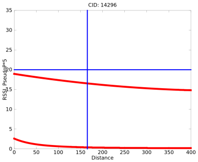
$$P(f(x), y) = \frac{1}{1 + (f(x) - y)^2}$$

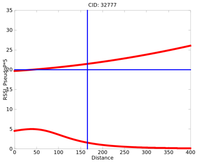
Calculating pseudo-density

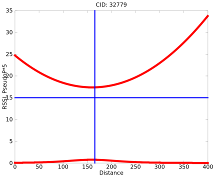






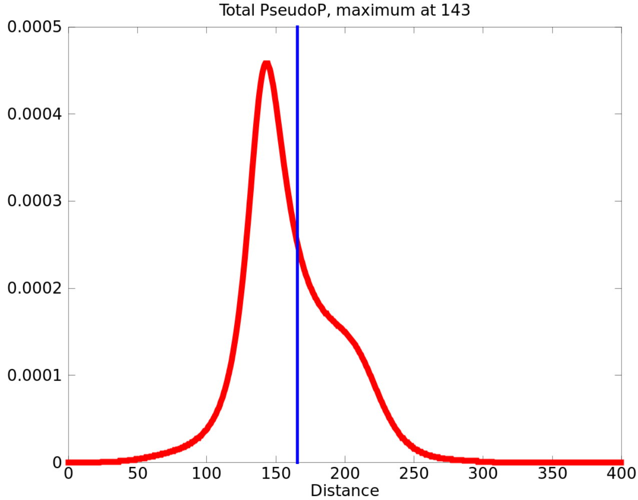




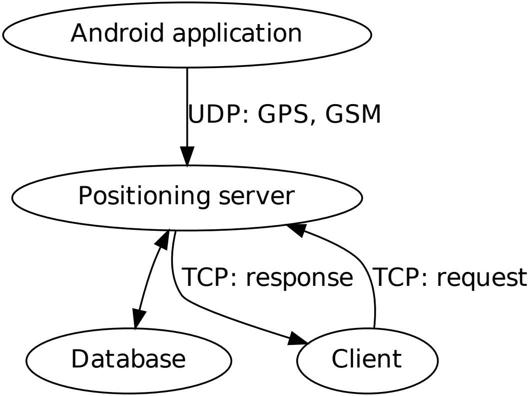


Resulting pseudo-density

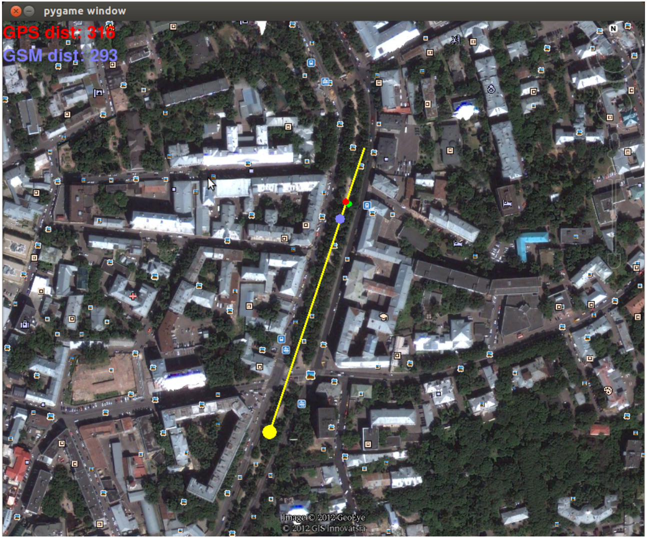
Result: 143 meters, actual position (according to GPS): 165,5 метров.



Architecture

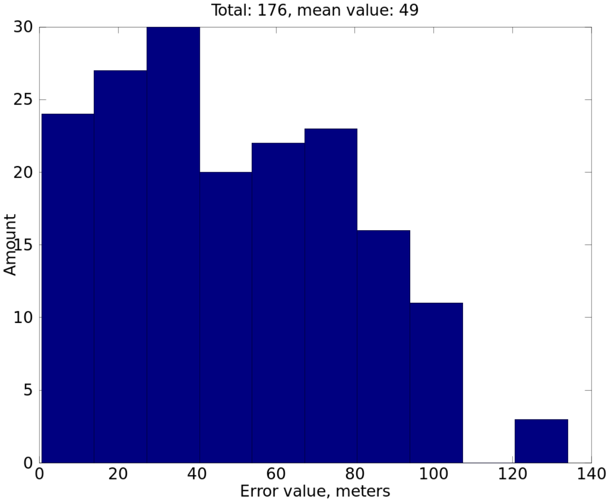


Testing


pygame window
GPS dist: 316
GSM dist: 293
Image © 2012 GeoEye
© 2012 GIS Innovatsia

Histogram of errors

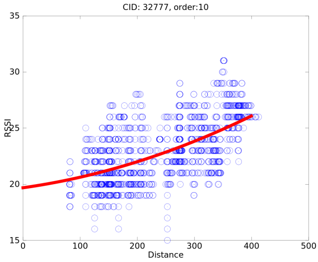
176 experiments, the mean value of errors: 49 meters.



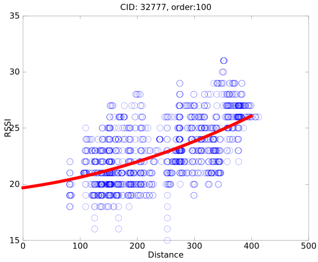
Comparison

Parameter	GPS	Triangulation	My algorithm
Precision, meters	10	200	50
Cost	GPS+GSM	GSM	GSM
Coverage	Worldwide	Citywide	Route-wide

Analyzing results



Maximal degree of a polynomial: 10



Maximal degree of a polynomial: 100

Conclusion

1. The new method works;
2. Its precision is better than that of triangulation;
3. Its precision is worse than that of GPS;
4. Additional improvements are required and possible.

Tracking public transportation vehicles by cellular network signals

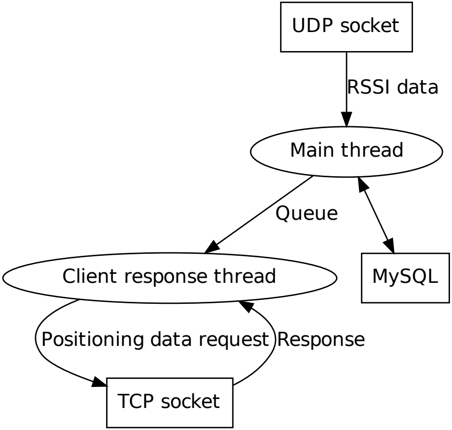
http://svn.auditory.ru/repos/tatmon/

By Maxim Kovalev
maxim.kovalev@sv.cmu.edu

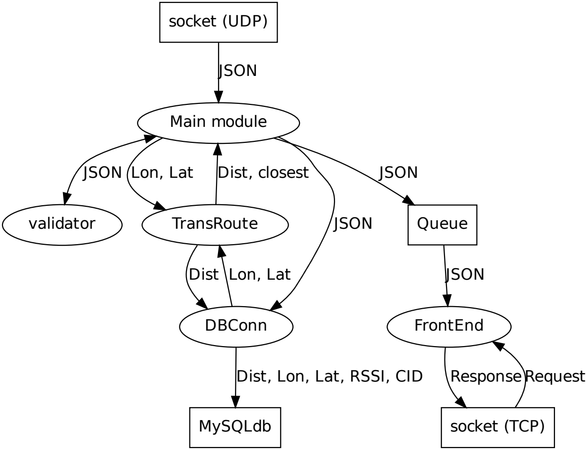
Carnegie Mellon University, PhD sutudent

August 2013

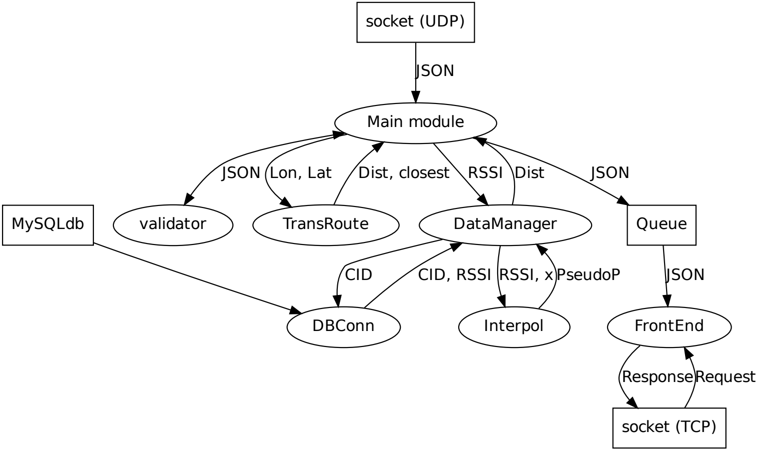
Threads of the server-side



Collecting data mode



Positioning mode



Mobile application

Example of a message


```
{ "GSM":{
"cellcount":2,
"cells":[
{"CID":11531, "Psc":−1, "RSSI":26, "type":"EDGE"},
{"CID":32779, "Psc":−1, "RSSI":22, "type":"EDGE"}
]
},
"GPS": {
"lng":37.64814019203186,
"ltd":55.7543760538101 2,
"acc":24.0
}}
```

Algorithm input

$$data = \begin{pmatrix} Dist_0 & RSSI_0 \\ Dist_1 & RSSI_1 \\ \vdots & \vdots \\ Dist_{len(data)-1} & RSSI_{len(data)-1} \end{pmatrix}$$

Creating variables

$$self.X = \begin{pmatrix} 1 & Dist_0 & Dist_0^2 & \cdots & Dist_0^{order} \\ 1 & Dist_1 & Dist_1^2 & \cdots & Dist_1^{order} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & Dist_{len(data)-1} & Dist_{len(data)-1}^2 & \cdots & Dist_{len(data)-1}^{order} \end{pmatrix}$$

$$self.Y = \begin{pmatrix} RSSI_0 \\ RSSI_1 \\ \vdots \\ RSSI_{len(data)-1} \end{pmatrix}$$

Normal equations

$$self.theta = (self.X^{\intercal} \cdot self.X)^{+} \cdot self.X^{\intercal} \cdot self.Y$$

```
def solve_theta(self):
    self.theta = numpy.transpose(self.X)
    self.theta = numpy.dot(self.theta, self.X)
    self.theta = numpy.linalg.pinv(self.theta)
    self.theta = numpy.dot(self.theta,\
        numpy.transpose(self.X))
    self.theta = numpy.dot(self.theta, self.Y)
```