FOSS4G 2017

Visualization and analysis of active transportation patterns derived from public webcams

Anna Petrasova, J. Aaron Hipp, Helena Mitasova

Center for
Geospatial Analytics

North Carolina State University

Public webcams

Rich source of spatio-temporal information

- weather, traffic, changes in environment, phenology, ...
- active transportation behavior in urban areas

AMOS

The Archive of Many Outdoor Scenes

- collection of long-term timelapse imagery from publicly accessible outdoor webcams around the world
- 1,128,087,180 images taken from 29945 webcams
- a project of the Media and Machines Lab Washington University in St. Louis
- online browsing of images and download available
- metadata and tags to improve discoverability of webcams

From image to information

How to get from image to information useful for analysis?

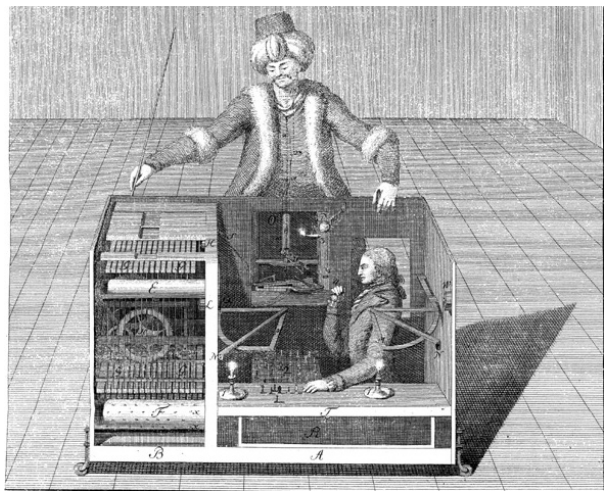
Artificial intelligence

- machine learning
- neural networks

https://xkcd.com/1838

Artficial artficial intelligence

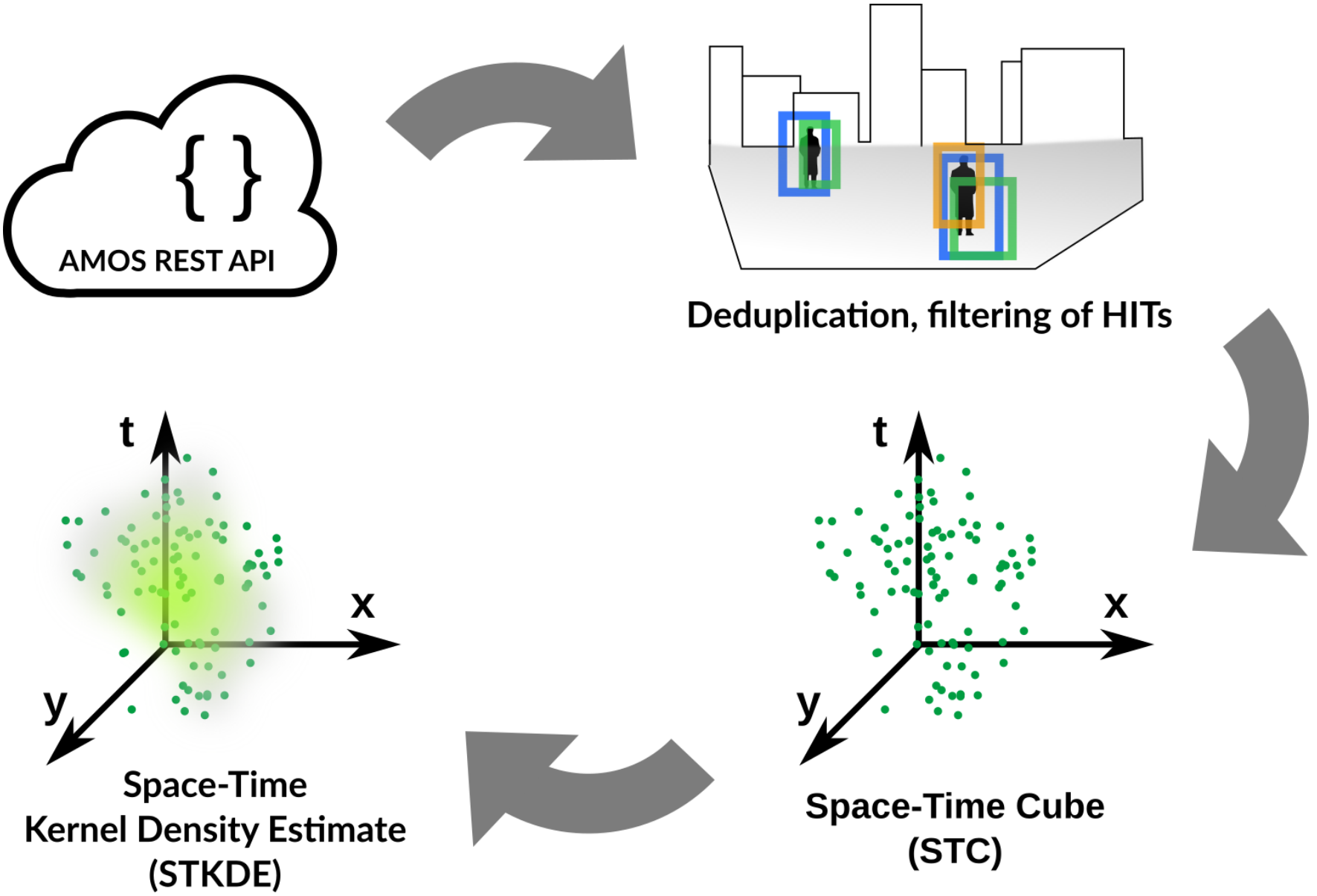
- Amazon Mechanical Turk
- crowdsourcing marketplace platform



fake chess-playing machine (late 18th century)

mTurk HITs (Human Intelligence Tasks)

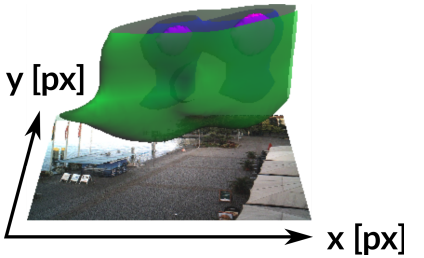
HITs processing workflow



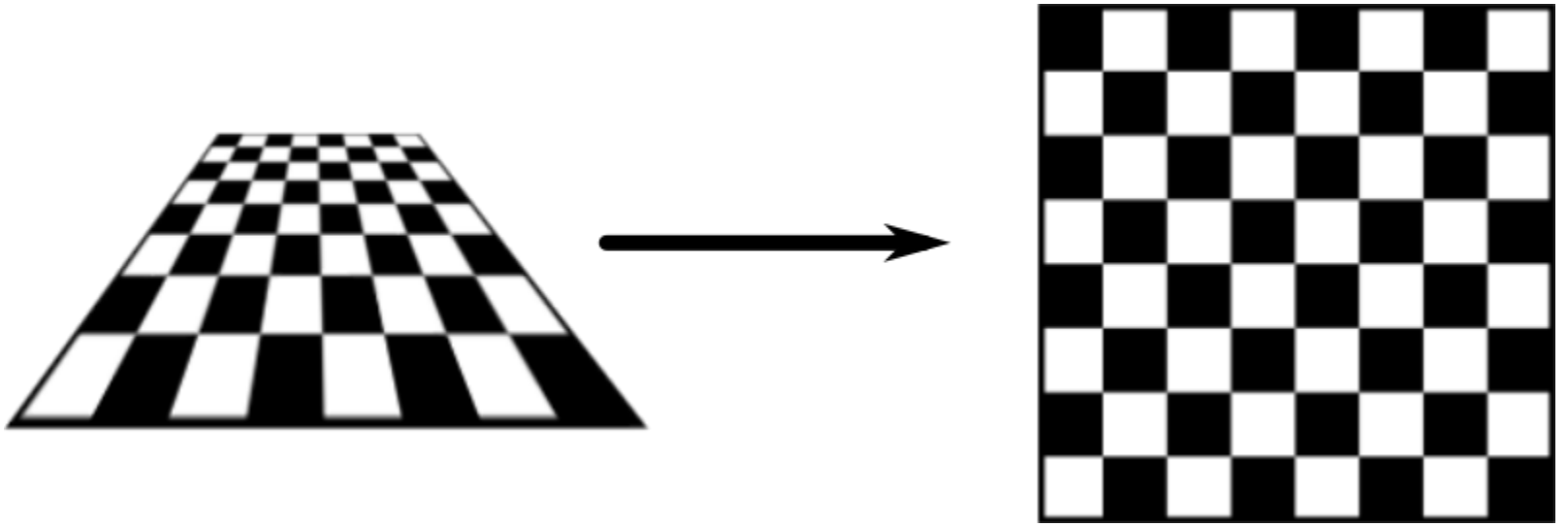
Georeferencing

Using coordinate system of the webcam image:

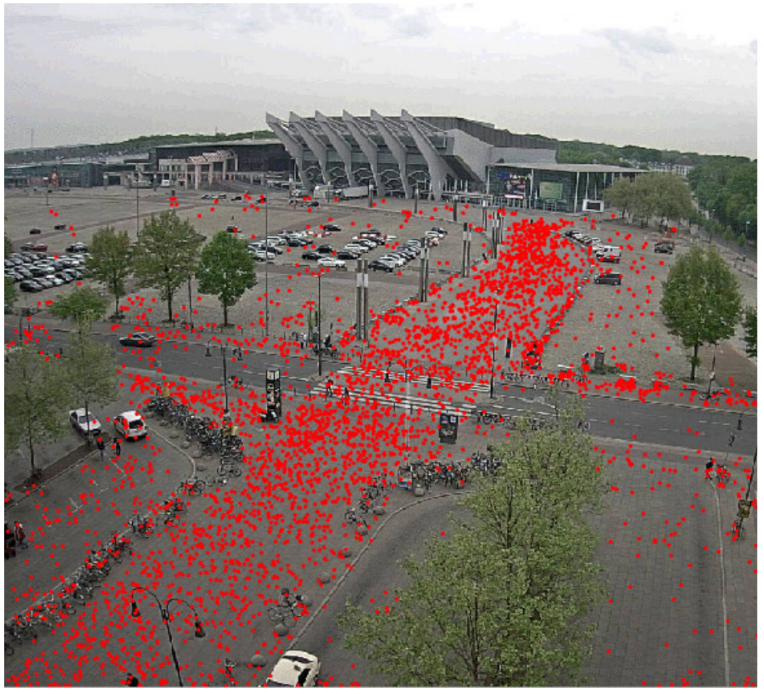
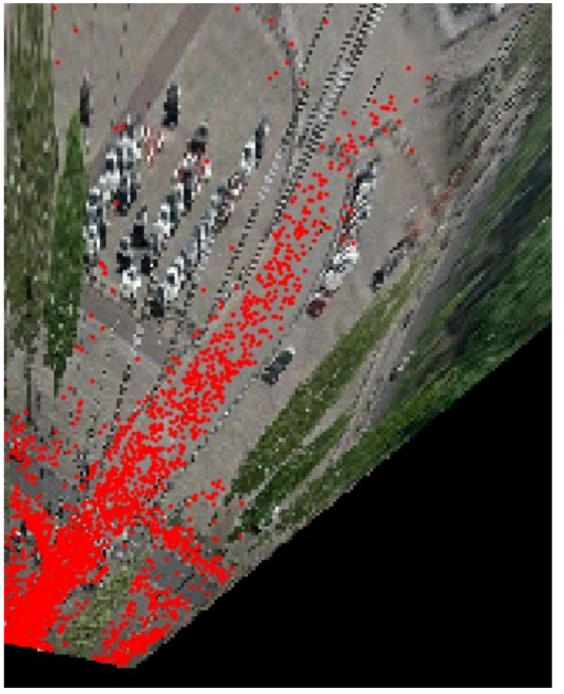
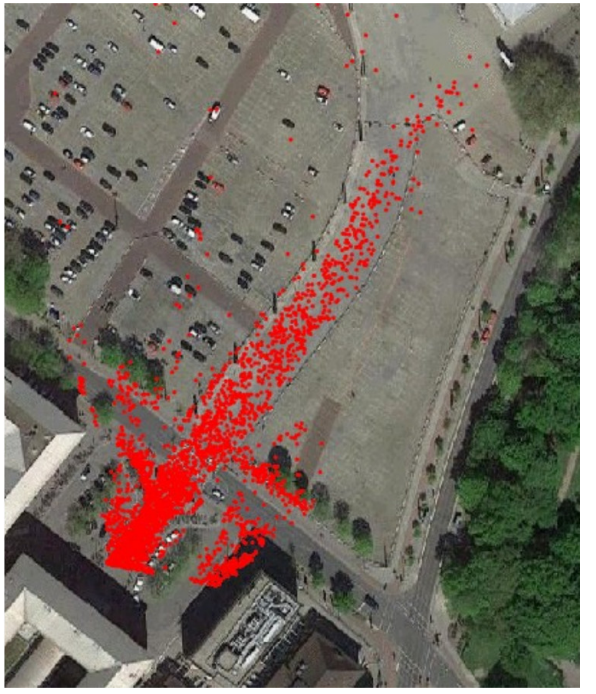
- distances in the image represent varying distances in reality
- we can't integrate other geospatial datasets (streets, POIs) or information from other webcams



Solution is to compute **projective transformation** by matching 4+ stable features in the webcam image to the same features in the orthophoto.

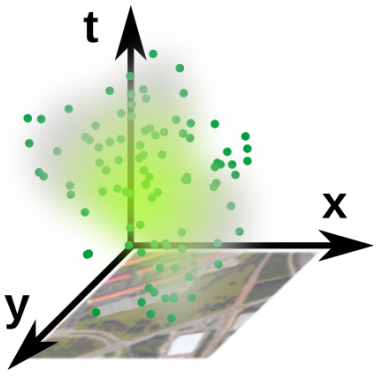


Georeferencing: example

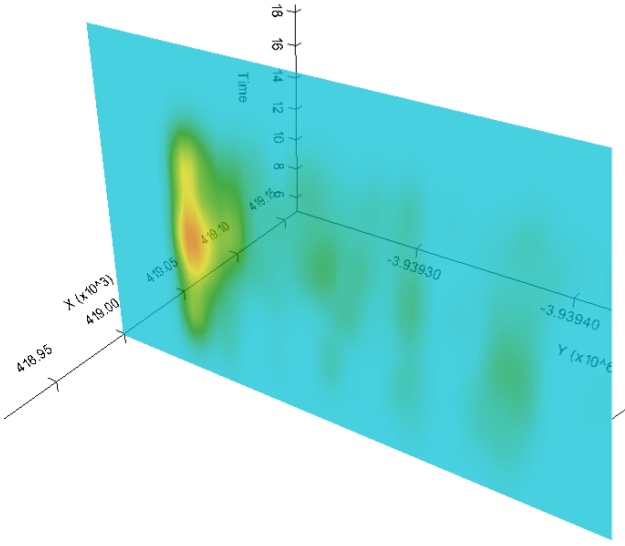
Caveats: some webcams change orientation, many objects such as benches, traffic marking are unsuitable as GCPs, stable objects such as statues can move too

STC visualization

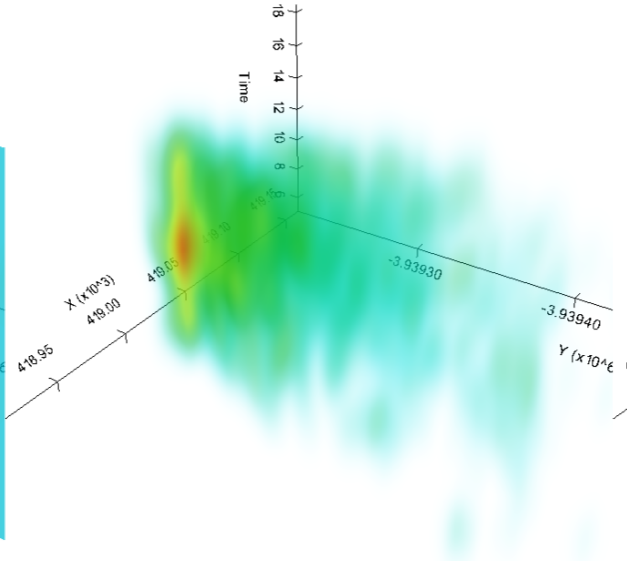


Space-time density of pedestrians represented as a 3D volume, computed using multivariate Kernel Density Estimation (KDE) with different spatial and temporal bandwidths

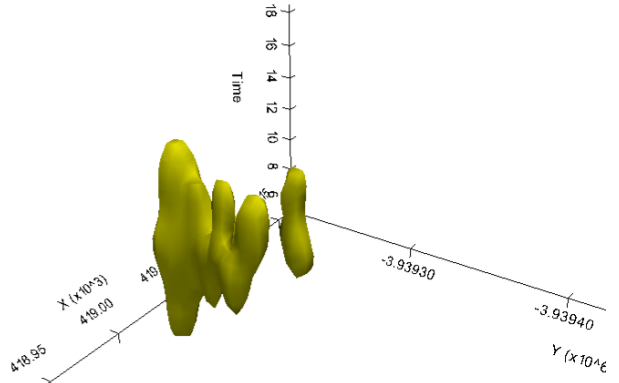
slice



volume rendering

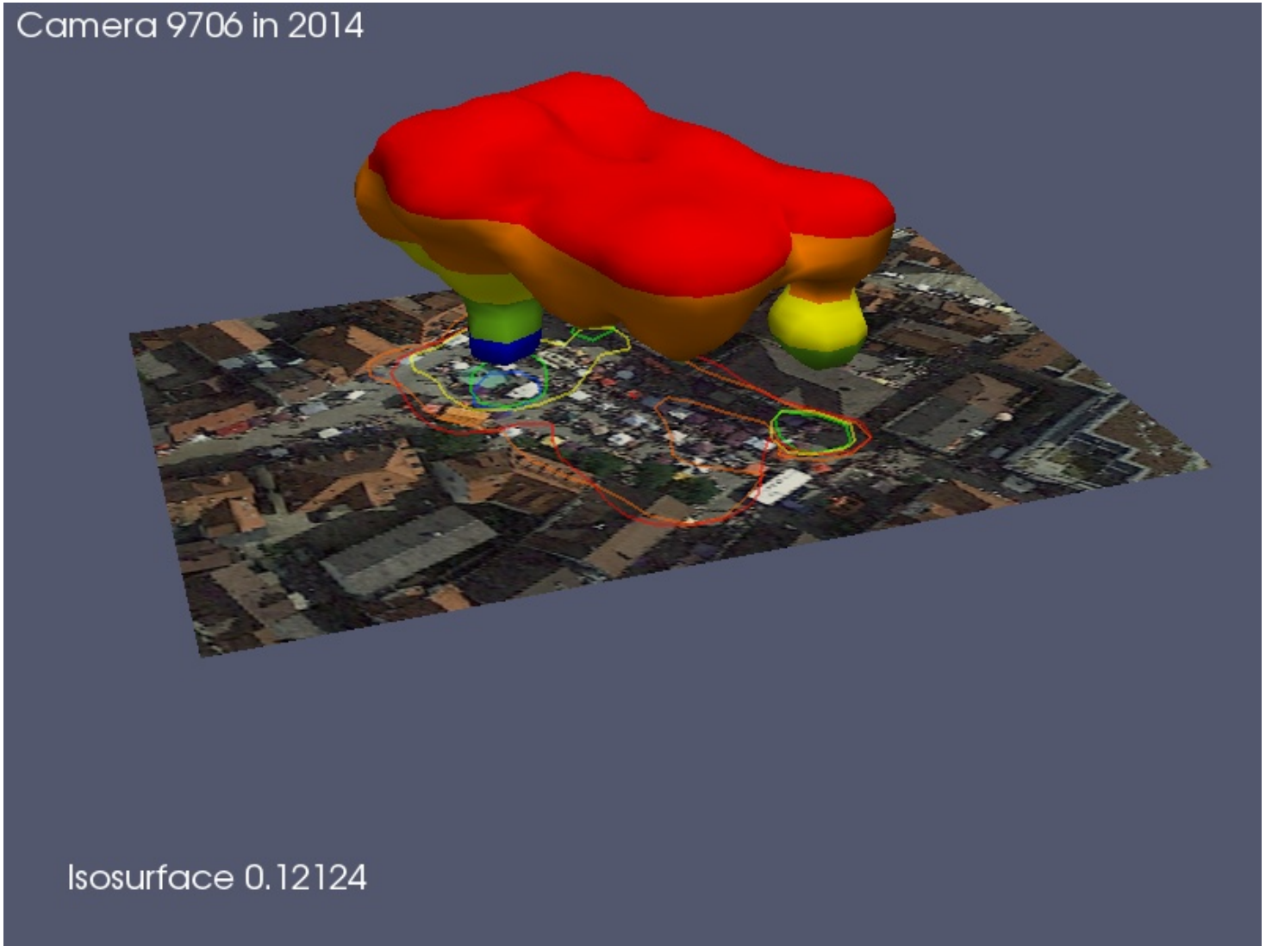


isosurface



Pedestrian density visualization

webcam 9706 (July), Ehingen, Germany



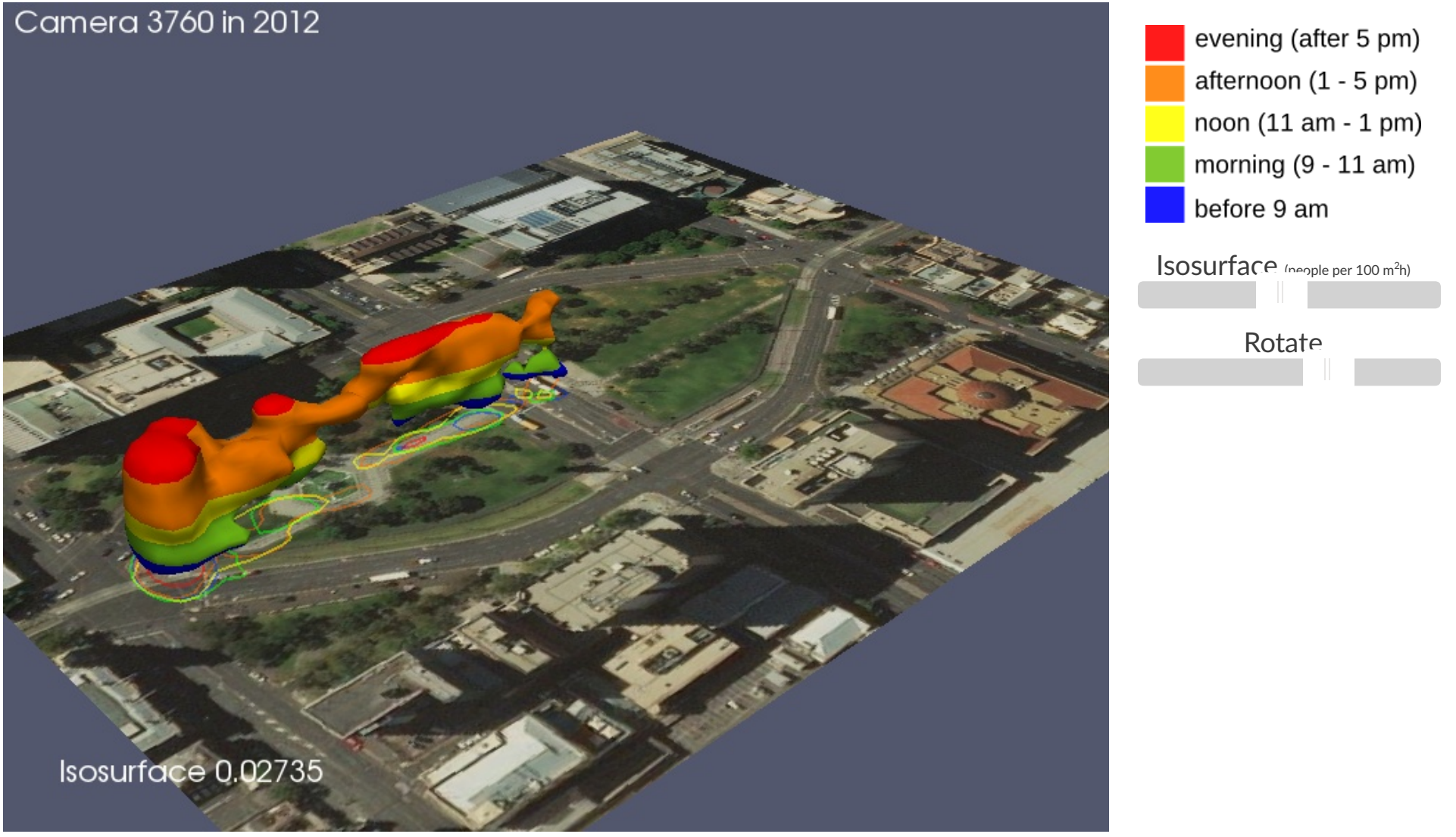
evening (after 5 pm)
afternoon (1 - 5 pm)
noon (11 am - 1 pm)
morning (9 - 11 am)
before 9 am

Isosurface (people per 100 m^2h)

Rotate

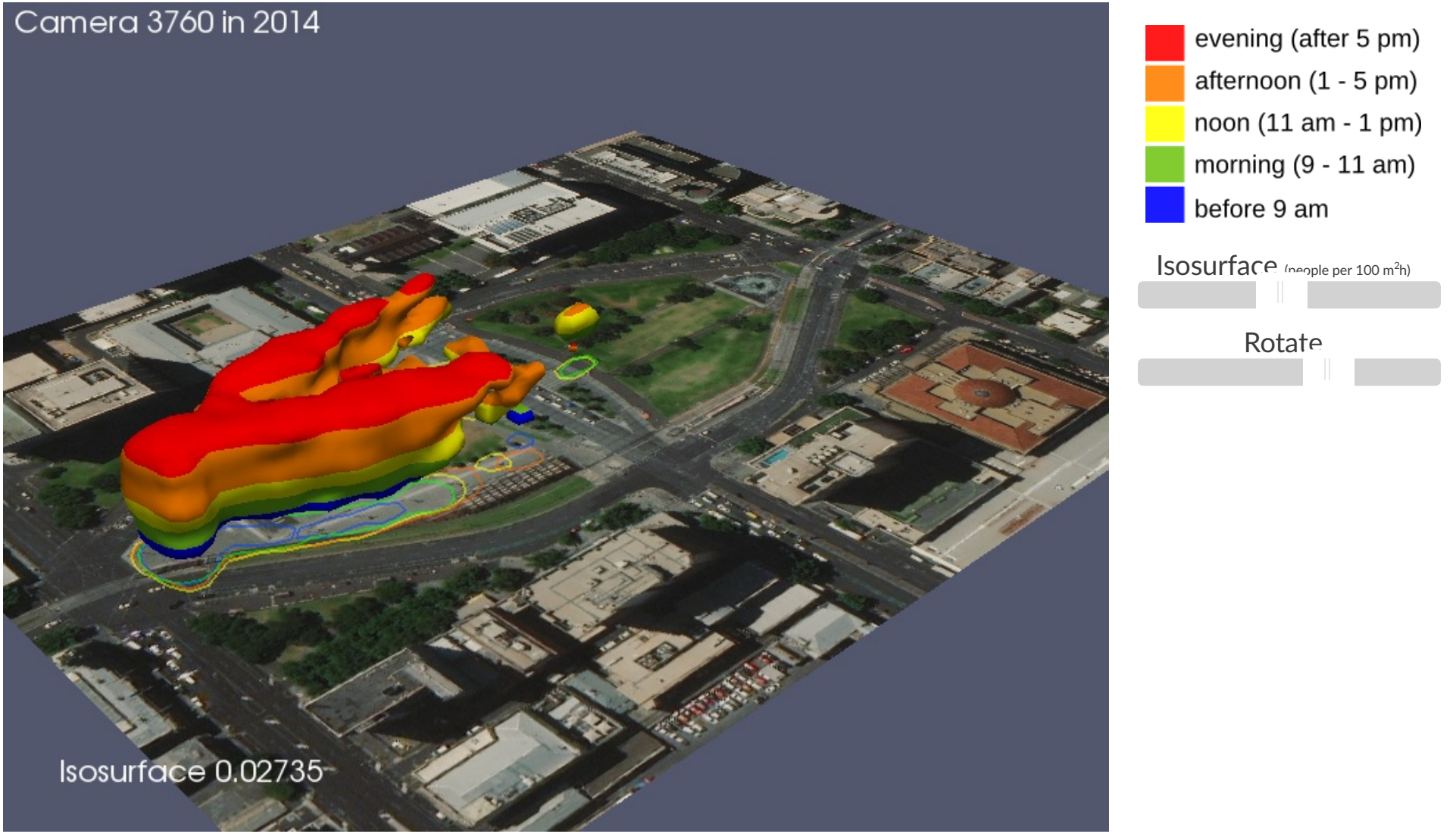
Effects of plaza reconstruction

webcam 3760 in 2012 (Jul - Sep), Victoria Square, Adelaide, Australia



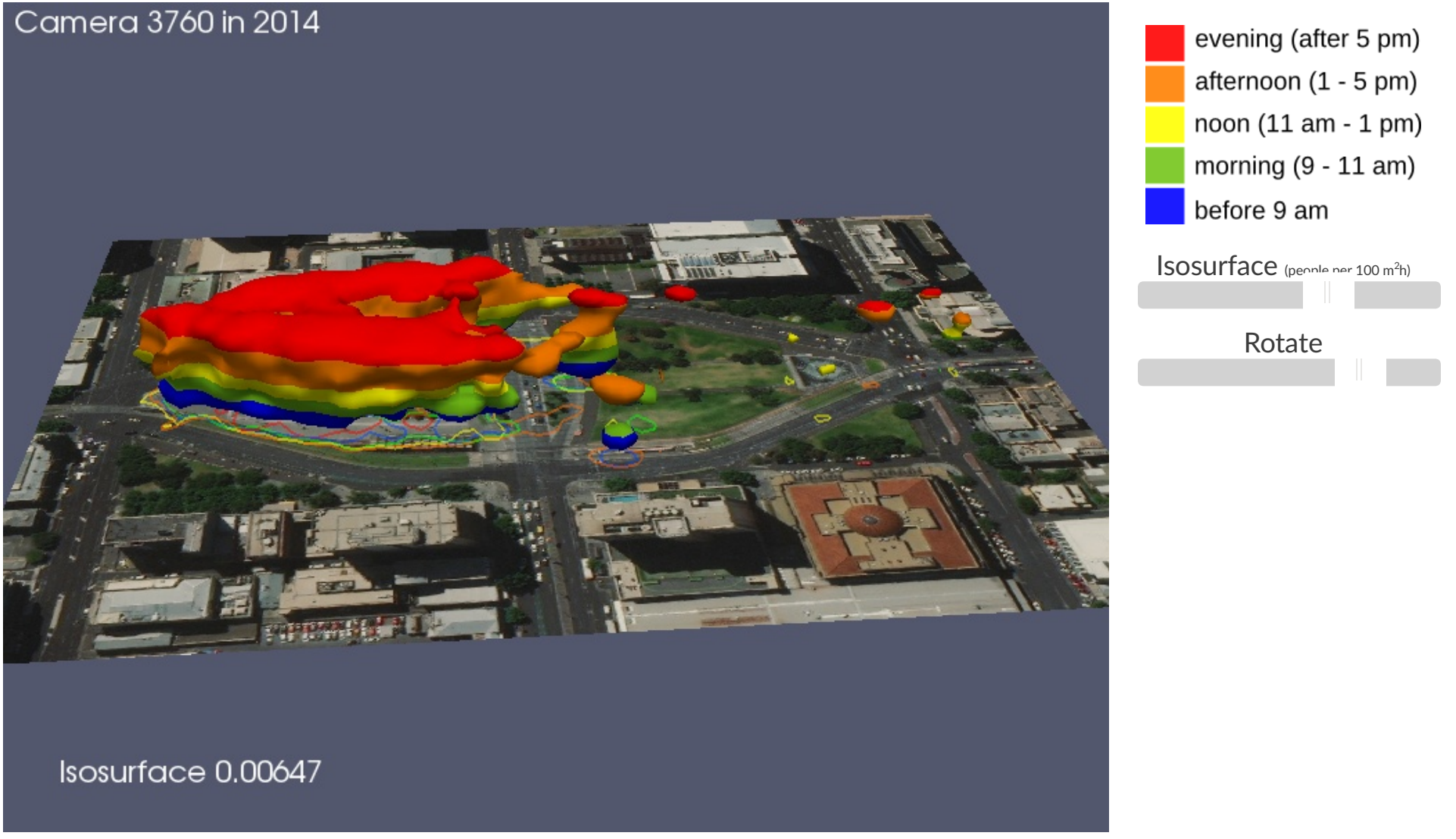
Effects of plaza reconstruction

webcam 3760 in 2014 (Jul - Sep), Victoria Square, Adelaide, Australia



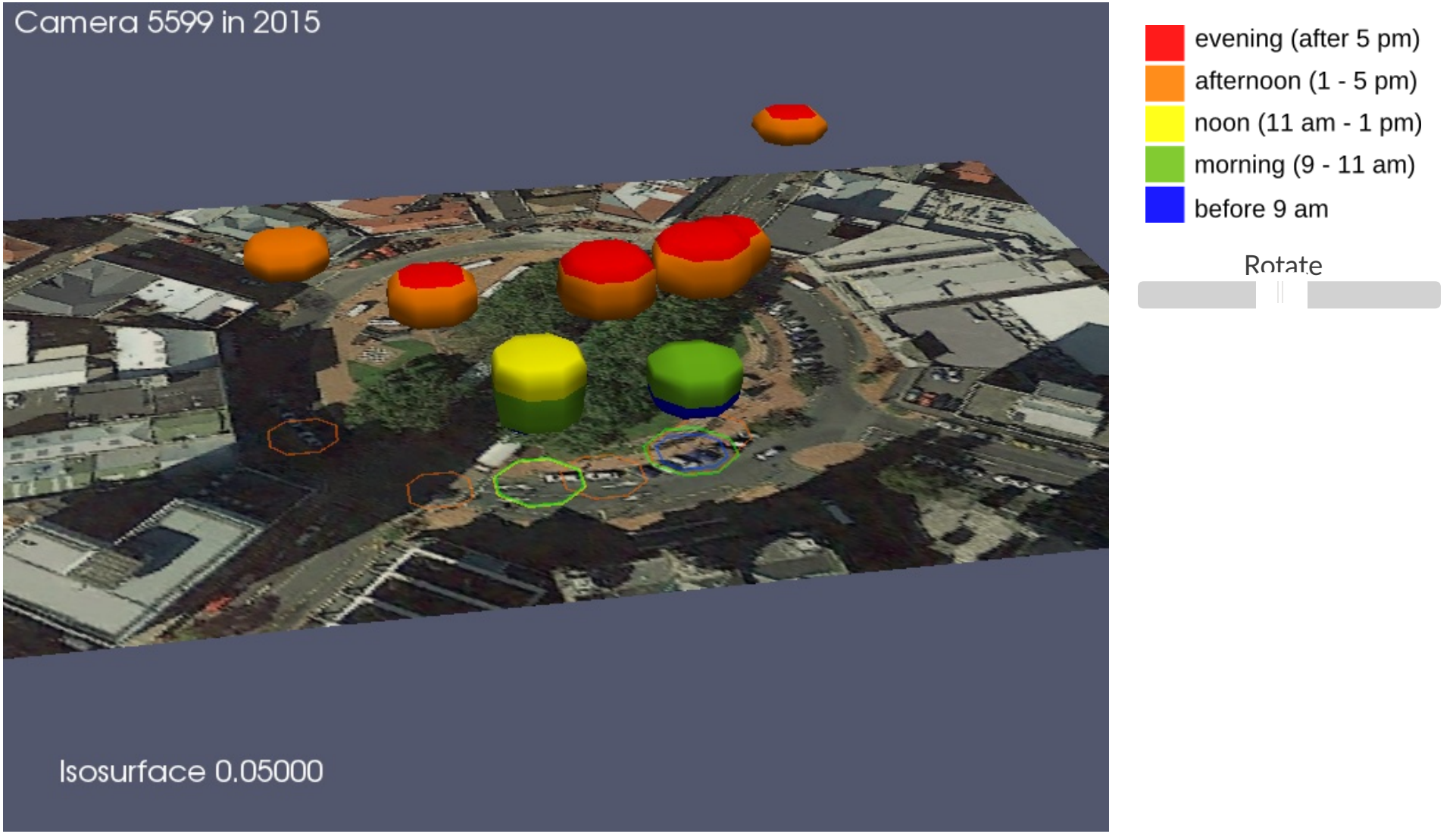
Change in pedestrian density (2014 minus 2012)

Positive values ~ increase in density in 2014
Negative values ~ decrease in density in 2014



High density of pedestrians and vehicles (webcam 5599)

$$\texttt{if (P > percentile}^{P}_{99}\texttt{ AND V > percentile}^{V}_{99}\texttt{, V + P)}$$

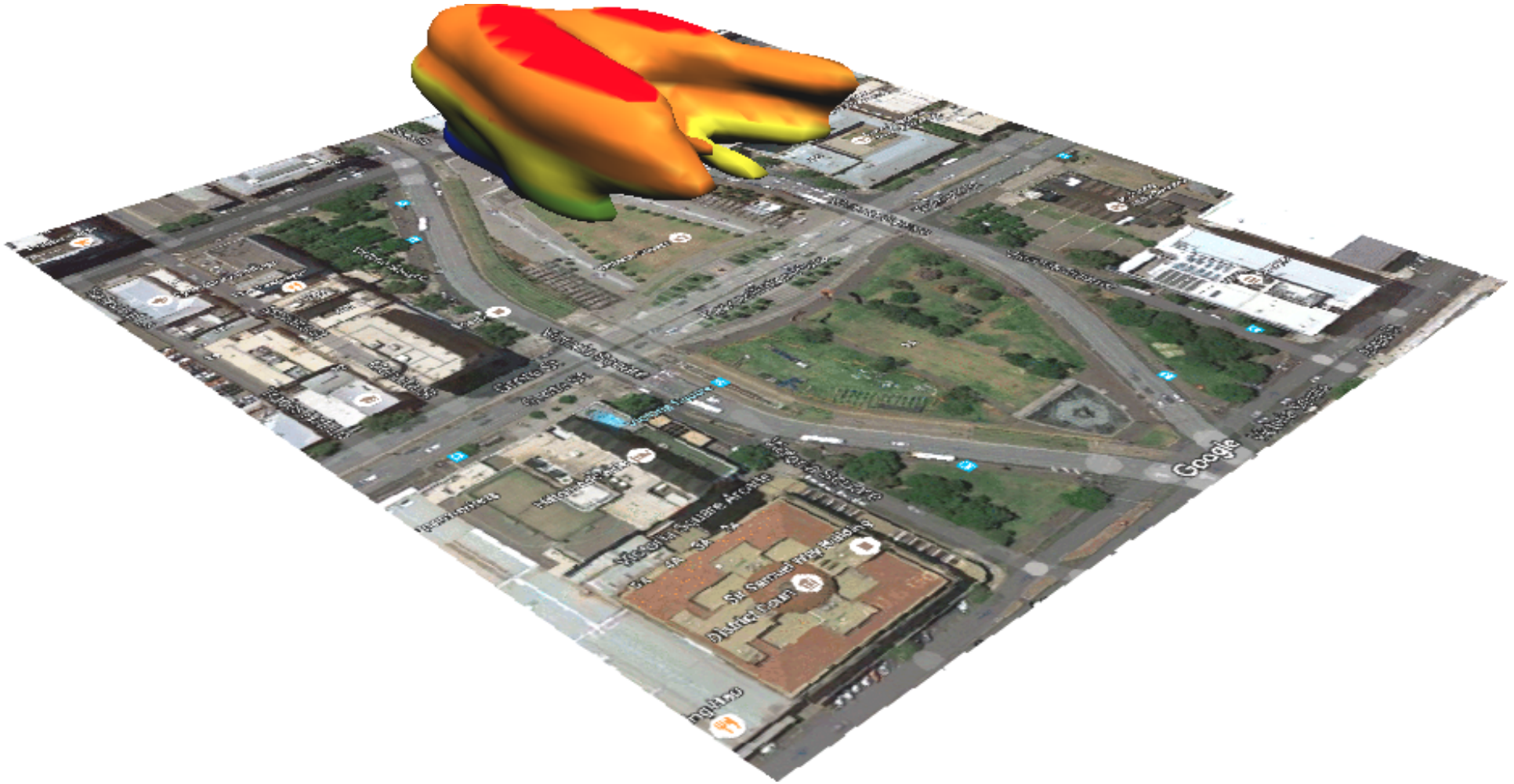


Software

- Python libraries
- Jupyter Notebook for data exploration
- Georeferencing: scikit-image, GRASS GIS
- KDE: SciPy, Statsmodels
- Rendering: GRASS GIS, ParaView, Blender

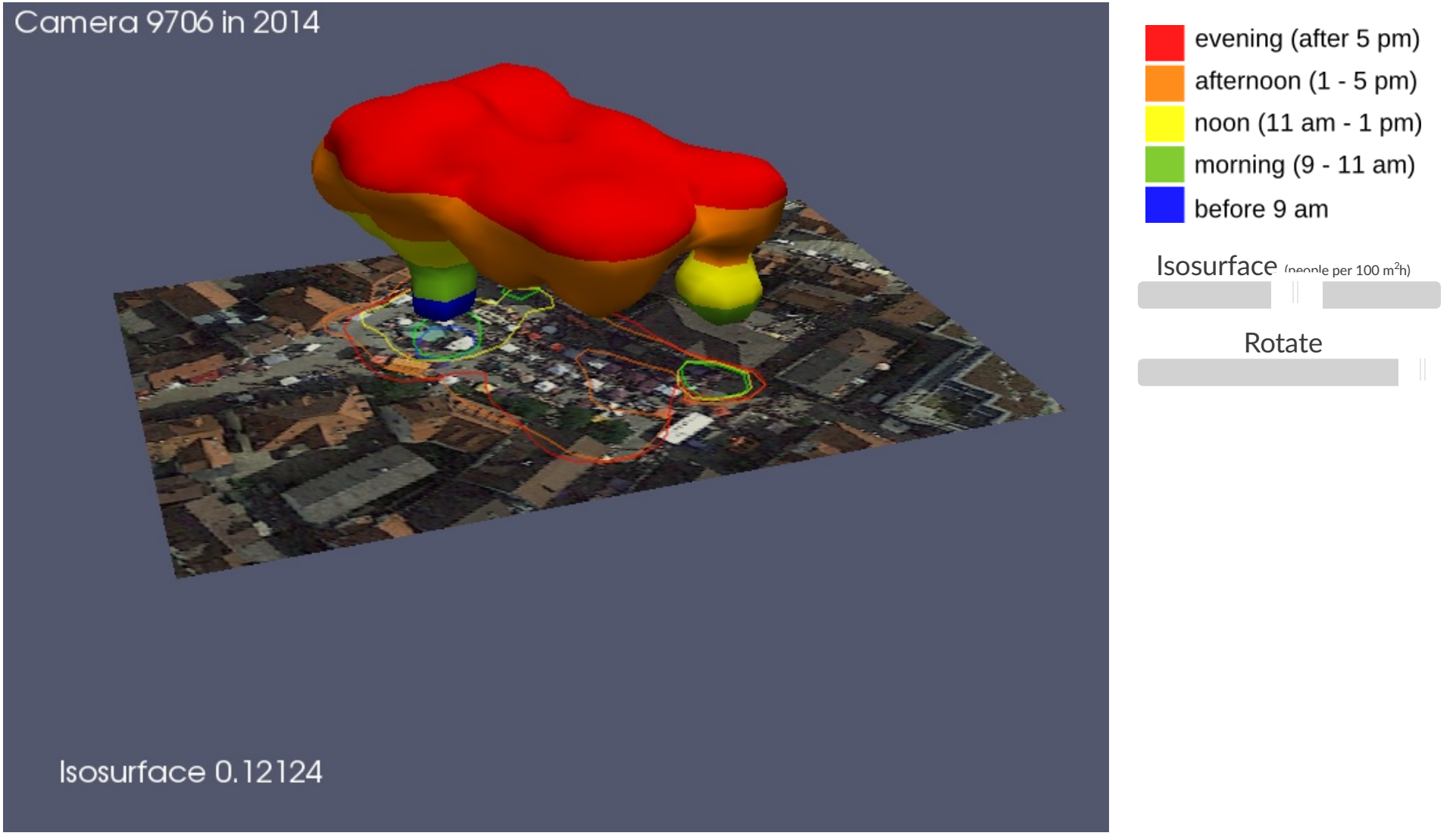
github.com/petrasovaa/amos-visualization

Visualization: GRASS GIS

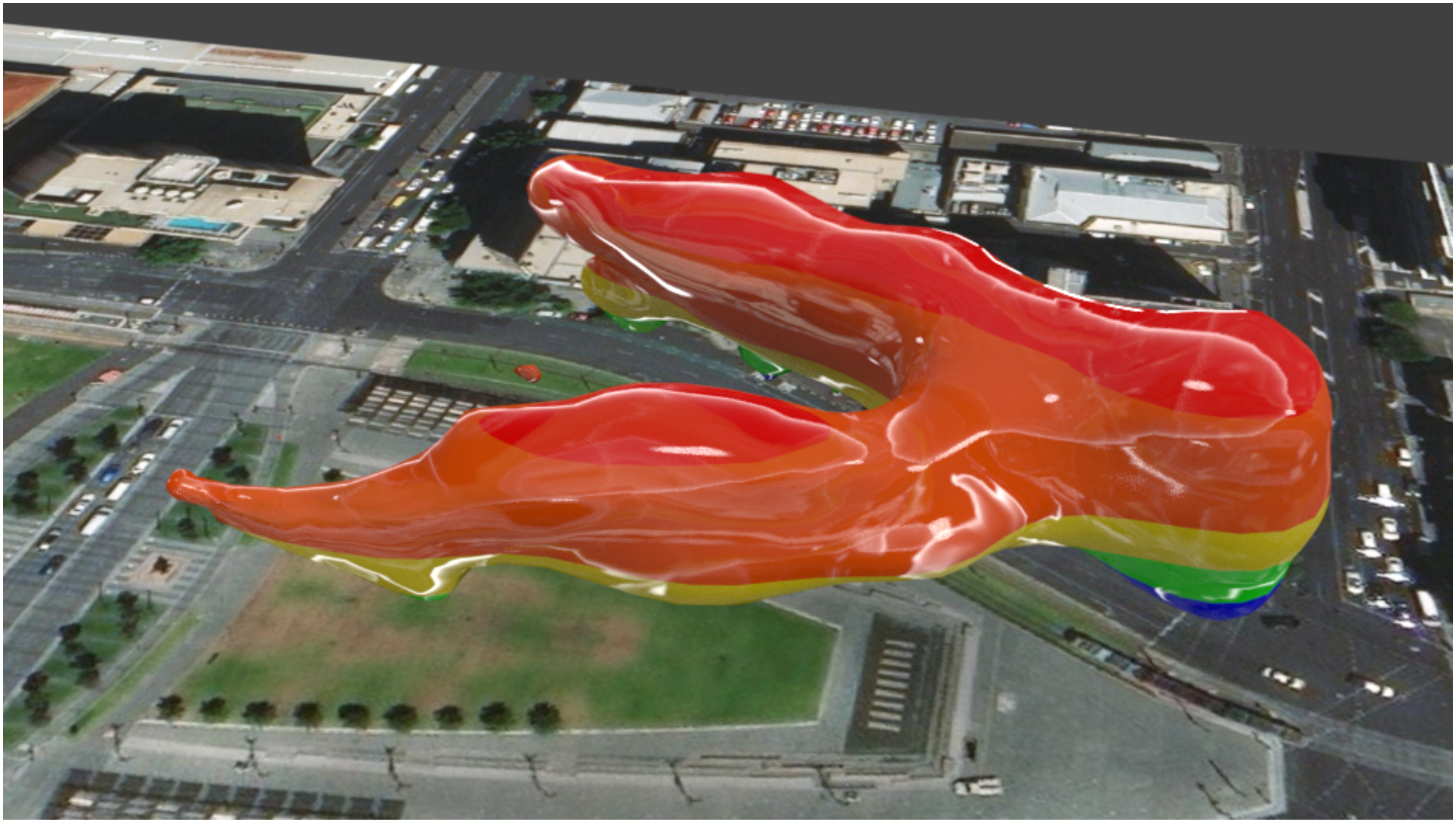


Visualization: ParaView

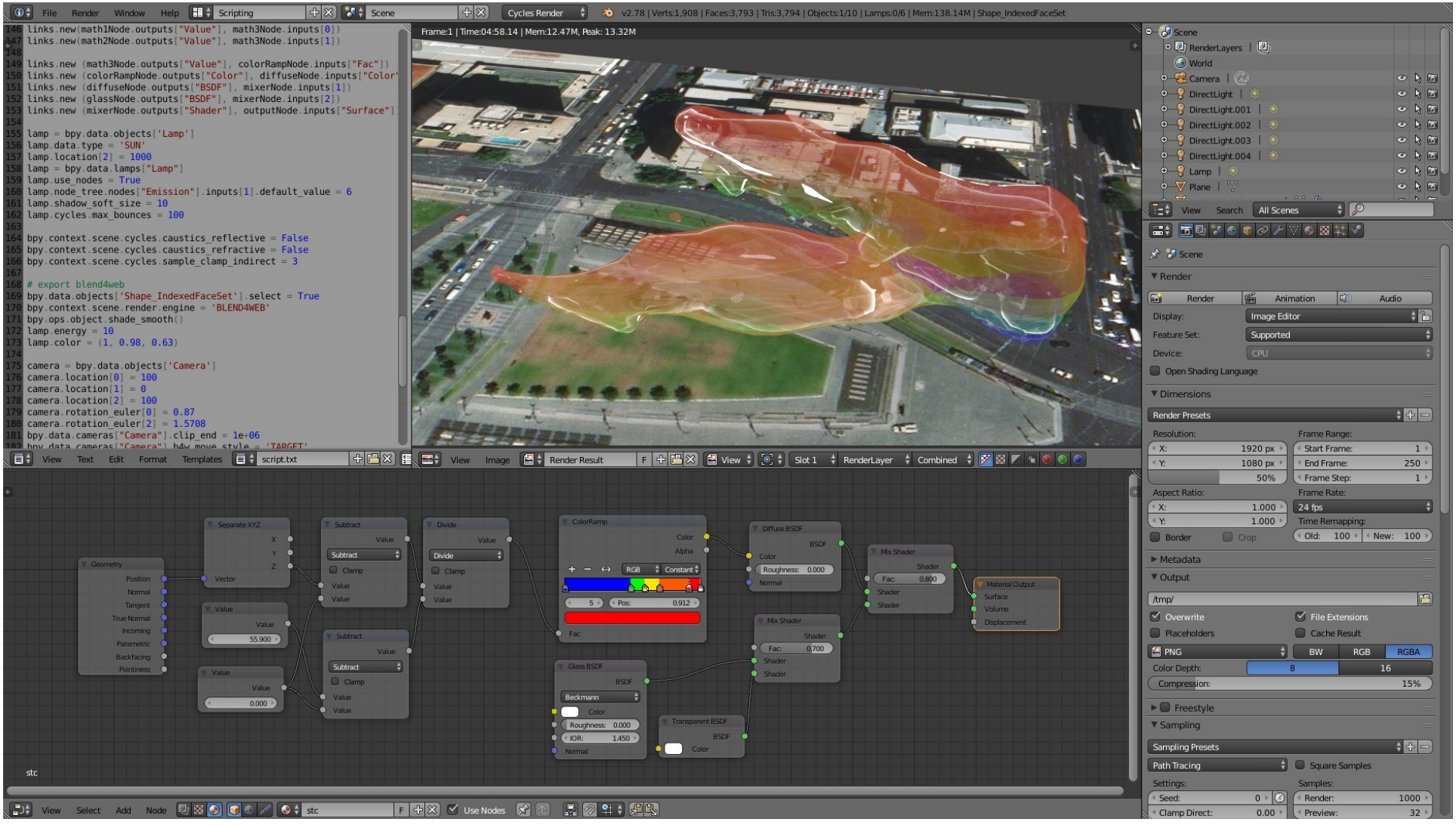
webcam 9706 (July), Ehingen, Germany



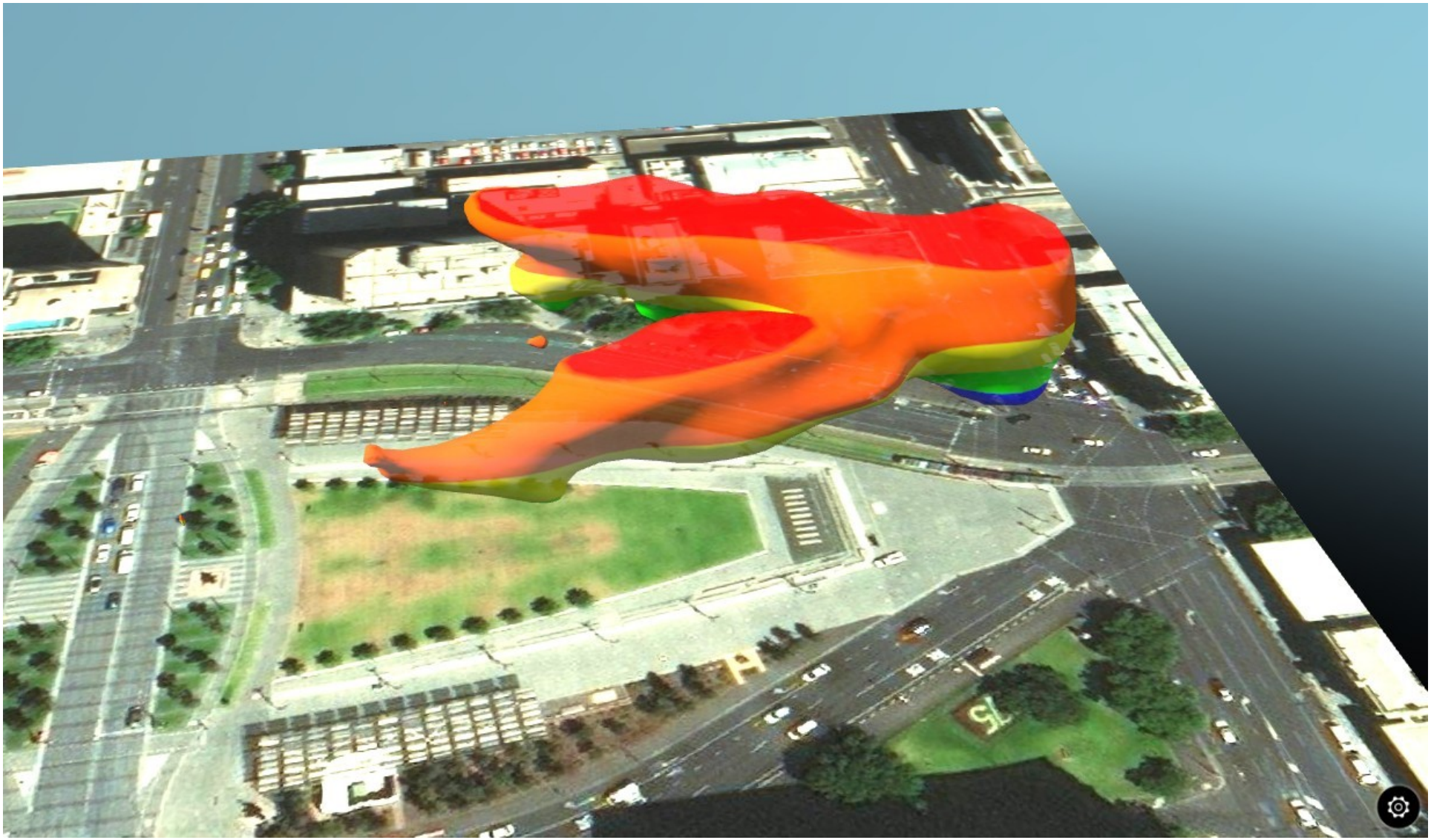
Visualization: Blender



Blender


File Render Window Help Scripting Scene Cycles Render v2.78 | Verts:1,908 | Faces:3,793 | Tris:3,794 | Objects:1/10 | Lamps:0/6 | Mem:138.14M | Shape_IndexedFaceSet
links.new(math1Node.outputs["Value"], math3Node.inputs[0])
links.new(math2Node.outputs["Value"], math3Node.inputs[1])
links.new (math3Node.outputs["Value"], colorRampNode.inputs["Fac"])
links.new (colorRampNode.outputs["Color"], diffuseNode.inputs["Color"
links.new (diffuseNode.outputs["BSDF"], mixerNode.inputs[1])
links.new (glassNode.outputs["BSDF"], mixerNode.inputs[2])
links.new (mixerNode.outputs["Shader"], outputNode.inputs["Surface"]
lamp = bpy.data.objects['Lamp']
lamp.data.type = 'SUN'
lamp.location[2] = 1000
lamp = bpy.data.lamps["Lamp"]
lamp.use_nodes = True
lamp.node_tree.nodes["Emission"].inputs[1].default_value = 6
lamp.shadow_soft_size = 10
lamp.cycles.max_bounces = 100
bpy.context.scene.cycles.caustics_reflective = False
bpy.context.scene.cycles.caustics_refractive = False
bpy.context.scene.cycles.sample_clamp_indirect = 3
# export blend4web
bpy.data.objects['Shape_IndexedFaceSet'].select = True
bpy.context.scene.render.engine = 'BLEND4WEB'
bpy.ops.object.shade_smooth()
lamp.energy = 10
lamp.color = (1, 0.98, 0.63)
camera = bpy.data.objects['Camera']
camera.location[0] = 100
camera.location[1] = 0
camera.location[2] = 100
camera.rotation_euler[0] = 0.87
camera.rotation_euler[2] = 1.5708
bpy.data.cameras["Camera"].clip_end = 1e+06
Frame:1 | Time:04:58.14 | Mem:12.47M, Peak: 13.32M
Scene
RenderLayers
World
Camera
DirectLight
DirectLight.001
DirectLight.002
DirectLight.003
DirectLight.004
Lamp
Plane
View Search All Scenes
Render
Render Animation Audio
Display: Image Editor
Feature Set: Supported
Device: CPU
Open Shading Language
Dimensions
Render Presets
Resolution:
X: 1920 px
Y: 1080 px
50%
Frame Range:
Start Frame: 1
End Frame: 250
Frame Step: 1
Aspect Ratio:
X: 1.000
Y: 1.000
Frame Rate:
24 fps
Time Remapping:
Old: 100 New: 100
Border Crop
Metadata
Output
/tmp/
Overwrite File Extensions
Placeholders Cache Result
PNG BW RGB RGBA
Color Depth: 8 16
Compression: 15%
Freestyle
Sampling
Sampling Presets
Path Tracing Square Samples
Settings:
Seed: 0
Clamp Direct: 0.00
Samples:
Render: 1000
Preview: 32
View Text Edit Format Templates script.txt
View Image Render Result View Slot 1 RenderLayer Combined
Geometry
Separate XYZ
Subtract
Divide
ColorRamp
Diffuse BSDF
Mix Shader
Material Output
Value
Glass BSDF
Transparent BSDF
stc
View Select Add Node stc Use Nodes

Visualization: Blend4Web



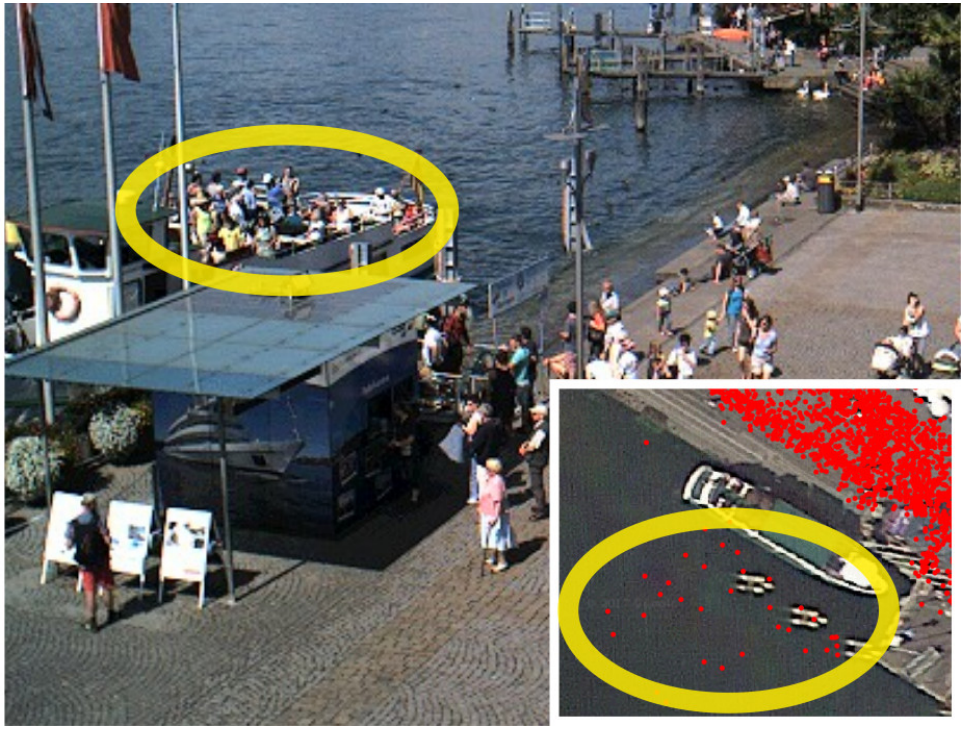
Conclusion & Future work

- new method for **harvesting and visualization** of spatio-temporal information about active transportation
- new way for cities to **detect and analyze changes** in active transportation behavior in an unintrusive way
- georeferenced data give us the ability to **incorporate other geospatial data and methods** (e.g., solar radiation modeling)
- possible thanks to the **synergy between crowdsourcing** technologies (AMOS, mTurk, open source software)
- **machine learning** techniques trained by mTurk data will enable us to analyze much **larger data** volume in real-time, possibly leading to the discovery of more patterns

Appendix

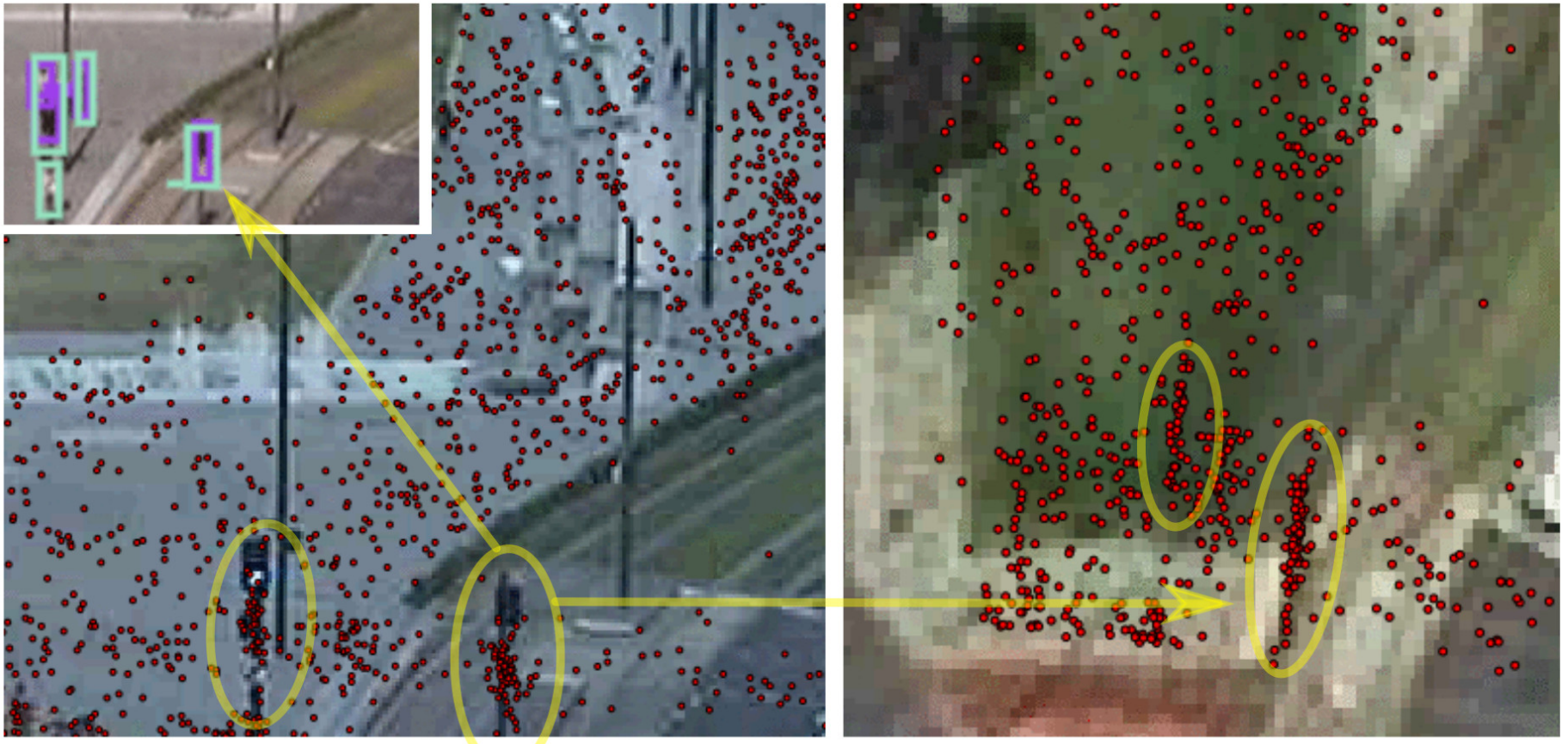
Challenges: webcam geometry and view

- areas hidden behind trees or other objects
- assumes pedestrians and vehicles on a horizontal plane, otherwise we get large spatial errors



Challenges: mTurk reliability

Traffic lights, statues mistakenly marked as pedestrians, machine learning approaches would avoid this type of error



Webcam image

Georeferenced

References:

- Hipp, J. A., Adlakha, D., Gernes, R., Kargol, A., Pless, R., Drive, O. B., Louis, S. (2013). Do You See What I See: Crowdsource Annotation of Captured Scenes, 24–25. http://doi.org/10.1145/2526667.2526671
- Hipp, J. A., Manteiga, A., Burgess, A., Stylianou, A., Pless, R. (2016). Webcams, Crowdsourcing, and Enhanced Crosswalks: Developing a Novel Method to Analyze Active Transportation. Front. Public Health, 4(97). http://doi.org/10.3389/fpubh.2016.00097
- Jacobs, N., Roman, N., Pless, R. (2007). Consistent temporal variations in many outdoor scenes. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. http://doi.org/10.1109/CVPR.2007.383258