GPU-Accelerated Spatial Interpolation Rendering for Web-Based Environmental Monitoring

Christophe Lienert*, Hans Rudolf Bär, Lorenz Hurni****

* Department Construction, Traffic and Environment, Canton of Aargau
** Institute of Cartography and Geoinformation, ETH Zurich

Abstract. Hydrological processes are dynamic, highly interlinked and have to be examined on various time and spatial scales. Precipitation constitutes the main input into the hydrological system and largely determines runoff disposition and runoff amounts in a given catchment area. In order to better evaluate and reanalyze extreme precipitation, novel data processing and animated visualization methods are presented which are based on interpolated ground measuring precipitation gauges.

Graphical rendering of animations is associated with high computational costs and memory consumption. To overcome these limitations and instead of using rather static movies, we propose to delegate animated interpolations to the client-side graphical processing unit (GPU). GPUs are able to instantaneously interpolate large sets of scattered point data. Interpolated surfaces can then be visualized by classifying, color-coding and shading. Even the handling of large time series with hundreds or thousands of data points is possible. Using the Web Graphics Library (WebGL) as a client-side technology, animations can efficiently be distributed over the Internet and also offer user interactions. After discussing the needs in hydrology, the GPU-based technology is introduced and a use case in hydrology is presented.

Keywords: Interpolation, Animation, WebGL, Precipitation, Hydrology

1. Data Processing in Hydrology

1.1. State-of-the-art in hydrological data processing

Hydrological time series data are needed to quantify a wide range of interlinked processes and phenomena in the water cycle. Understanding these

requires high-quality hydrological data and modeling. Latter is usually data-intensive and the overall quality is strongly dependent on data availability and resolution. With the fast-paced advancement of Web-based and other technologies in hydrology, possibilities for new research fields and practice-oriented applications have widened remarkably. Technologies in data measurement engineering, transmission devices, data model standardization, data management tools, publication and visualization services are being improved at a staggering speed. But not only have amounts and temporal resolutions of collected data risen tremendously, so have data format heterogeneity and vocabularies. In contrast, user expectations to high-quality and timely hydrological data products and model outputs have risen, which are met by employing larger data processing infrastructures and novel (semi-) automated processing algorithms (Aquatic 2012, Solomatine & Wagner 2011, Lecca *et al.* 2011).

Data interoperability for real-time data exchange and merging between various organizations still pose significant challenges, mainly because of inherent semantic and structural heterogeneities of unprecedented amounts of data and non-uniform autonomous data sources (Ravindran *et al.* 2010). Various research initiatives in the hydrology domain have tackled this problem, resulting in attempts to create standardized markup languages, such as HydroML, or graphical user interfaces such as HydroSeek (Beran & Piasecki 2009, Horsburgh *et al.* 2009).

Hydrological databases - like many other environmental databases - are nowadays usually linked to GIS in order to better capture, manage, analyze and visualize data. GIS may be coupled more or less tightly to the hydrological model procedures or may assist users in the interpretation of model results (WMO 2011). Database design is a combination of various elements such as existing systems and data and user needs. Usability of the modeling and visualization tools is directly associated with the user acceptance, as users generally don't intend to deal with technicalities and underlying logical processes related to the database (Al-Sabhan *et al.* 2003, Hughes & Forsyth 2006, Lienert *et al.* 2011).

1.2. Available Data

In this paper, the discussion on available data is focused on precipitation data products. Data such as streamflow, water level, groundwater, evaporation, soil moisture or any other data related to hydrology is omitted. For the present study, 10 minutes precipitation data has been used from over 70 automated ground gauges of Swiss Federal Office for Meteorology and Climatology (Fig. 1). The precipitation data is collected on the ground with

different types of perennial, static or seasonal, temporal sensors. Data is transmitted every 10 minutes with additional real-time quality checks.

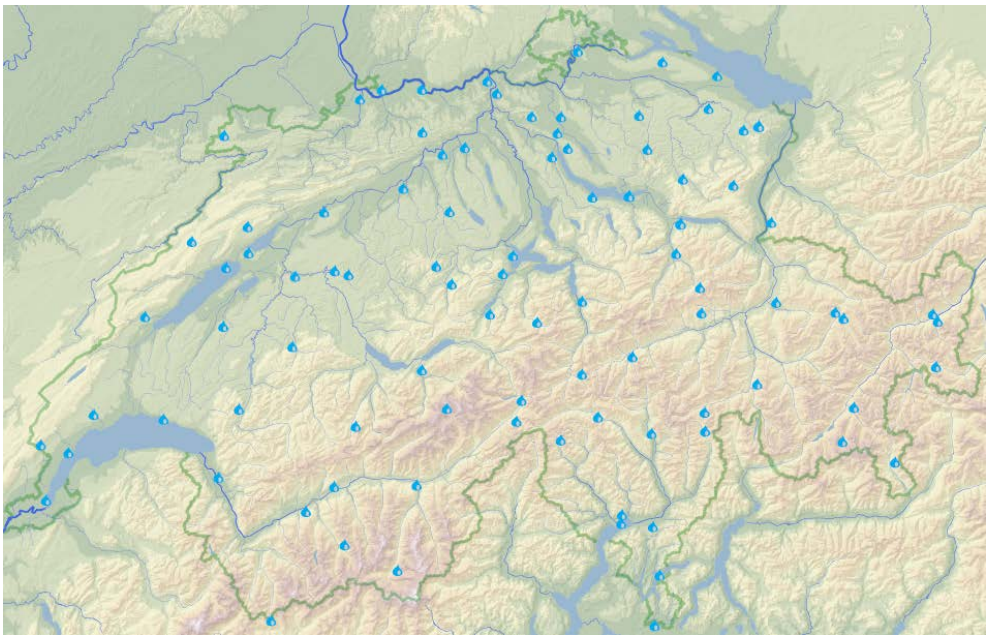

Figure 1. Overview of used precipitation data. The raindrops locate the automatic real-time ground gauging stations in Switzerland used for this study.

1.3. Point measurements vs. area-covering precipitation radar

Precipitation is characterized by its intermittency, high spatial and temporal variability, and sensitivity to environmental conditions such as wind. Precipitation is therefore–even today–quite difficult to measure accurately. Most commonly, point measurements are carried out using can-type gauges (tipping-buckets or weighing system) with a collecting area ranging from 125–500 cm^2 (Savina *et al.* 2012, Sevruk *et al.* 2009).

Besides being collected on the ground, precipitation is also qualitatively and quantitatively measured by precipitation radars. The strength of the radar is–still–that it captures well the spatial extent of entire precipitation cells, but less so the quantities, due to technical limitations in rain-rate conversions (cloud cover, ground clutter). Also, complex topography generally impedes accuracy, due to shielding effects (Joss & Waldvogel 1990, Panziera *et al.* 2011). Ground gauges are nowadays used for calibration of, and combination with, these radar measurements to eventually enhance the quality of real-time precipitation data products in terms of accuracy, temporal and spatial resolution (Wueest *et al.* 2010, Abdella & Alfredsen 2010).

Methodological frameworks for real-time combination of radar and gauges encompass objective analysis, interpolation with deterministic weights, and geostatistics, particularly the stochastic interpolation method Kriging (Erdin *et al*. 2012, Haberlandt 2007).

1.4. Intended purpose of precipitation data

Access to real-time precipitation data is especially important in order to encounter problems related to flood management, early warning, crisis management, environmental monitoring, or any other kind of ad-hoc decision-making (Matthies *et al.* 2007). Online, Web-based real-time data from sensors and precipitation radar are needed to centrally revaluate on-going current conditions. They help identify trends related other available measurement data, assess antecedent soil moisture conditions, and estimate possible response times of precipitation to produce superficial runoff within catchments (Kornelsen & Coulibaly 2013, Nikolopoulos *et al.* 2011, Zehe *et al.* 2005).

After flood events, reanalysis of the precipitation situations are needed by authorities and other stakeholders (Lienert *et al.* 2010). Compiling post-flood reports are usually costly, and data processing for visualization are time-consuming. In contrast, such information is often needed by crisis managements for communication purposes, during or shortly after the flood event. Thus, beside data quality, access to processed data plays another crucial role.

1.5. The importance of spatial interpolation

Spatial interpolation allow for quantifying area-covering precipitation amounts over a given time period and spatial unit, e.g. a catchment. Most physically or semi-physically based rainfall-runoff models require these spatially distributed quantities as input. Interpolation quality strongly depends on the location of the ground gauge as it has to be representative for the surrounding, unmeasured area. The representativeness of a gauge depends on topography, the climatological region and altitudinal belt (Grebner & Roesch 1998).

Interpolation involves weighted measured values from reference gauges, which surround the interpolated point. The main difference between current interpolation methods is how the weights are determined. The weights may be deduced from a) geometric-deterministic, e.g., nearest neighbors, arithmetic mean, inverse distance weighting, or b) statistical methods, e.g., kriging (Creutin & Obled 1982, Lebel *et al.* 1987). Compared to geometric methods, statistical interpolations are more complex, but do not necessarily lead to better results (Grebner & Roesch 1998, p.49 ff). Therefore, in this

paper, the geometric inverse distance weighting interpolation method is used for animated visualization (see Chapter 3).

1.6. The need for temporal animation

Spatially referenced visualization is key when evaluating, analyzing, understanding and communicating hydrological flood events, complementing non-spatial information such as tables and graphs. The term animation is used when several maps, representing a point in time, are temporally ordered, put in sequence and visualized–preferably without jolting (Ogao & Kraak 2002). Concise visualizations, particularly animations, are needed to address technical personal in crisis management teams who are usually under time pressure. But also non-technical stakeholders, such as journalists or rescue service-men may be much better addressed (Cutter 2003).

2. WebGL: OpenGL for Browsers

WebGL is a new standard for the well-established technology OpenGL (WebGL 2011). It brings the computing power of the graphics processing unit (GPU) to the Web browser by providing a JavaScript interface to OpenGL. WebGL is based on OpenGL for Embedded Systems (OpenGL ES 2010) and can be considered as a subset of OpenGL (since version 4.1).

The introduction of the OpenGL Shading Language (GLSL) starting with OpenGL version 2.0 in 2004 detached the processor from graphics-only tasks and opened it for general "General-Purpose Computing on the GPU" (Owens *et al.* 2007). WebGL and OpenGL ES did not only include GLSL in their implementation, but made its use even mandatory (OpenGL ES Shading Language 2009). WebGL programming involves a) the JavaScript programming interface which is responsible for the communication between the browser and the GPU, and b) GLSL interface which controls processes running entirely on the GPU.

When turning graphics data into image representations, the GPU follows a structured way of passing and processing data, denoted as the graphics pipeline (Owens *et al.* 2008). It is basically split into a modeling step, processing the data for the following rendering step. Essentially, there are two nodes in the graphics pipeline (see Fig. 2) where specialized programs can intervene: 1) where vertices are processed and 2) where pixels are assembled. This is where the Shading Language comes into play. The programmable units written in GLSL are also called shaders.

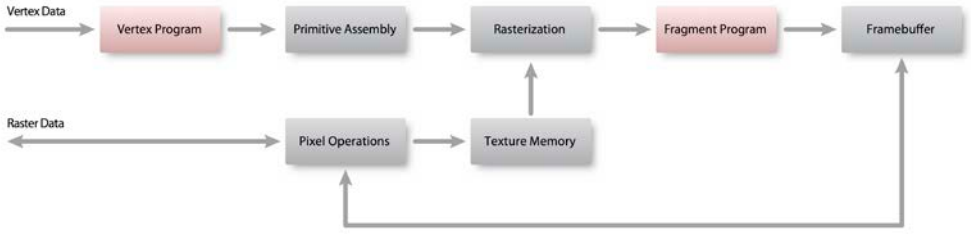

Figure 2. Schematic diagram of the OpenGL ES graphics system (WebGL 2011, slightly modified).

The GPU provides a data interface for vector and raster data, and compiled shader programs. As shown in Fig. 2, the vertices of the vector data are first processed by the vertex shader, assembled to graphics primitives, which are then rasterized. Raster data follows a different path, being first transformed to textures (a mapping-optimized raster format). Rasterized textures are then passed to the fragment program, which determines the color value of each pixel. The frame buffer finally stores all resulting pixels and is either connected to system window in order to display its content, served as a source for further textures, or is passed back to the host program.

WebGL does not only offer a way to free the CPU from computational task, it is also able to accelerate this process, potentially by an order of magnitude. This is made possible by the parallel computation design of the GPU. The GPU is a collection of many small processors running in parallel (Single Instruction - Multiple Data SIMD approach, see Owens *et al.* 2007).

Although the WebGL standard has only been released in 2011, WebGL is implemented in the major browsers and is ready to be used without extra installation. However, some features and most mobile browsers are not (yet) supported. Still in its infancy, a new standard is emerging called Web Computing Language (WebCL). It abstracts from the graphically motivated computation model and provides a more generic interface for parallel computing (WebCL Working Draft 2013).

3. Interpolation, Animation and Visualization Using WebGL

As mentioned above, the focus is on the inverse distance weighting interpolation method (IDW), which dates back to Shepard (1968). IDW is easy to implement, but is a computationally intensive interpolation method, as it operates on the entire (global) data set where every single data point is contributing to the interpolated value. The IDW method has drawn attention

just recently, as it is ideally suited for fast parallel execution (Henneböhl *et al.* 2011, Srinivasan *et al.* 2010).

Although OpenGL (and WebGL) is primarily known for astonishing 3D graphics, it may likewise be used efficiently for computation-intensive 2D raster task such as spatial interpolation. For this study, IDW was implemented with WebGL since it is installed on major browsers and therefore enables spatial interpolation in Web applications. In order to ensure that the graphical user interface remains responsive and that animations are smooth, most of the computational workload is left to the graphics processor and the necessary data are transferred to the graphics memory. Two techniques have been considered, both having their pros and cons.

The first technique uses multiple rendering passes, one for each data point that contributes to the final interpolation. Starting with an empty framebuffer, weighted values as well as the weights are added and written to a second framebuffer using two color channels. The framebuffers are then exchanged and the procedure is repeated until all data points have been processed. An additional rendering pass is then required to adjust the interpolated values to the sum of weights equal to one. Since this method continuously exchanges data between two framebuffers, it is called ping-pong technique (Göddeke 2007).

The second technique tries to avoid the computational overhead introduced by the ping-pong technique. This requires transferring the previously CPU-based loop over all data points to the shader program. Unfortunately, shader programs are not allowed to have variable-sized loops. As a consequence, switching to another data sets requires the recompilation of the shader programs. Given rather small shader programs and a relatively short compilation time, this restriction is more of an aesthetical nature.

Interpolation and visualization parameters are transferred to the GPU by so-called uniform values, either as scalar values or small arrays. Point data sets may also be passed as arrays of uniform values but they are restricted to rather small sizes (the actual value depends on the graphics hardware or the graphics driver). To avoid this limitation, we considered packing the point data into textures and access them from the shader program by texture look-up functions. Texture sizes are limited only by the maximum texture size, which is much higher than the maximum size of uniform values. In this case it is important to ensure that the WebGL implementation supports the floating-point texture extension which otherwise would severely limit the precision of the interpolated values. Apart from that, textures are particularly interesting since they allow the simultaneous loading of data for all animation phases. Also, using textures enables data pre-processing delegation to the GPU, such as the calculation of running sums.

GPU-based computation does not necessarily end with number-crunching interpolation. To achieve optimal performance, it is meaningful to also run subsequent visualization steps on the GPU. In the presented implementation, classification of the interpolated values and color-coding is packed into an additional rendering pass using texture look-up functions. By interpreting interpolated values as texture coordinates, interpolated values can directly be mapped to color values. To further improve the visual perception of the interpolated surface, shading similar to the one used for digital elevation models is applied.

The very last rendering step involves the composite with a base map. In order to serve different map scales, a vector data map is used. It is rasterized and converted to a texture only when the resolution of the map is known. The composite will involve image blending or multiplication, depending on the needs.

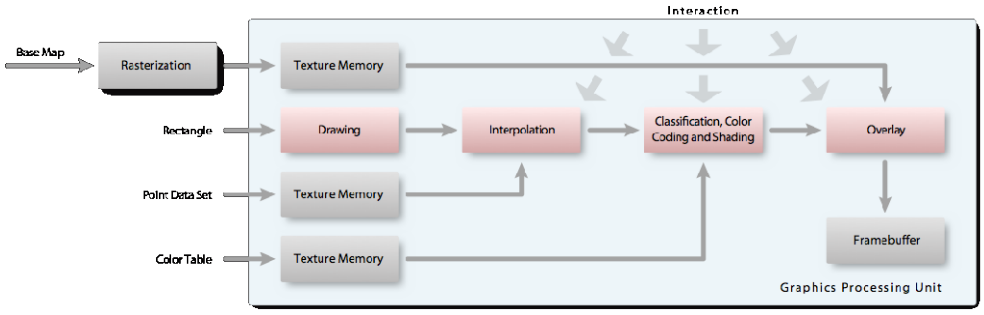

Figure 3. Data flow and data processing for GPU-based spatial point data interpolation.

Figure 3 schematically presents the data flow and the necessary stages of computations. It shows that all computation steps from the raw data to the final map are performed on the GPU. User interactions control the data and parameters to be used for interpolation, the classification, color schemes and shading to be applied and finally the composite of the resulting map image.

Assuming that point data sets are available at regular but rather long time intervals, temporal interpolation becomes an issue, especially if smooth animations are desired. Given a continuous temporal interpolation function, data values may be estimated for any point in time, within the available time period. For best performance and in order to keep each stage of data processing entirely on the GPU, the WebGL's built-in linear interpolation function is used. This requires the point data set to be packed into a texture. The interpolated values are then calculated on-the-fly for a given point in time.

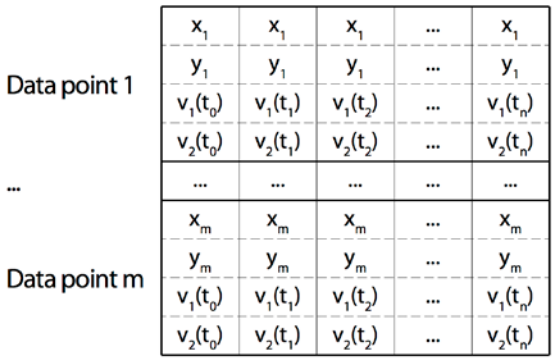

Figure 4. Data layout for temporal interpolation.

Figure 4 shows the data layout for the point data sets that is well suited for fast data access and linear interpolation along the time axis. The texture is organized so, that its coordinates refer to a specific data point, and a given point in time, respectively. The four components of every texture element are supposed to carry the data point's coordinates in the plane and the data value. The forth component is reserved as an additional value, such as a running sum for a given interval. Data point coordinates are stored redundantly on purpose. Enabling WebGL's linear interpolation option along the time axis, a single texture access function call is able to retrieve the data point's coordinates and the interpolated value simultaneously.

Given a maximum texture size of typically 8192 by 8192 texture elements, more than 8000 measurements may be interpolated for more than 8000 points in time. Thus, measurements at e.g. hourly intervals may be interpolated and animated over a period of nearly up to a year. The frame rate mainly depends on the number of stations and the image size of the final map. On modern laptops, smooth animation for the interpolation of around 100 data points and an image size of about one megapixels were achieved.

4. Prototype System for Hydrologic Applications

A prototype system for animated visualization of interpolated point data sets is shown in Fig. 5. The map shows a frame from a flood event in Switzerland of August 2005, based on 24 hours sums over 10 minutes measurement intervals. The user interface is realized with Ext JS, panels are resizable and the slider fit the window size.

The left side panel is used to choose the data set for the animation. The bottom panel contains the interface elements, which control interpolation and animation.

A Time slider is used to choose the measurements at a specific point in time. In order to able to access all measurements, the width of the slider should be larger than the number of measurements.

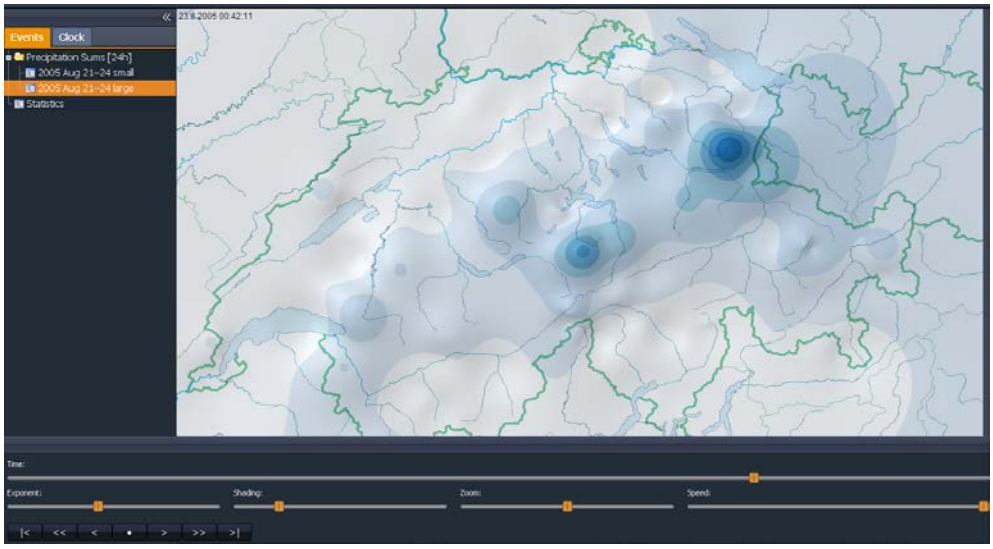

Figure 5. Prototype system for the animated visualization of interpolated point data sets, showing the graphical user interface and resulting map.

An exponent is applied to the inverse distance when interpolating the data values. Small values tend to narrow, larger values tend to widen the region of influence.

A shading slider controls the percentage the shading of the calculated surface contributes to the map. A value of zero avoids surface shading.

A zoom slider controls the scale of the map.

A speed slider most efficiently controls the animation. It determines the time elapsed between two consecutive animation frames. Linear time interpolation is applied to all data measurements. Negative speed values animate backward in time.

Animation control buttons are used to jump to the first measurement of the period, to start animation backward in time, to move one step to the previous measurement, to stop running animation, to start animation forward in time, to move one step to the next measurement and to jump to the last measurement of the period.

5. Discussion of Results, Conclusions and Outlook

We have been able to show that WebGL allows to interpolating and smoothly animating large time-series of data measurements for hundred and more gauging stations. Without further installations, a modern Web browser will be sufficient to dynamically visualize raw measurement data. WebGL is accelerating data interpolation by a factor of 100 to 500, compared to an implementation written in JavaScript.

The presented prototype may serve as a valuable tool for visually inspecting, controlling, and analyzing large time series of point measurements. It will enable users to efficiently discover spatial patterns, retrace flood events, understand hydrological processes, particularly in retrospect, and communicate to involved stakeholders in early warning and crisis management.

A number of potentially useful features have not yet found its way in the prototype implementation. We assume that the GPU-based calculation is not expected to raise particular difficulties for tasks such as

- The calculation of running sums for arbitrary time intervals
- The estimation and overlapping of precipitation amounts over a given catchment area
- The inclusion of additional precipitation interpolation improvements (e. g. including digital elevation models)

Temporally animated interpolations will not be restricted to hydrology and precipitation visualization. Other environmental data from point measurements will potentially be suited for a similar processing.

References

Abdella Y., Alfredsen K. (2010) A GIS toolset for automated processing and analysis of radar precipitation data. *Computers & Geosciences* 36(4), pp 422–429

Al-Sabhan W., Mulligan M., Blackburn G.A. (2003) A real-time hydrological model for flood prediction using GIS and the WWW, Computers, *Environment and Urban Systems*, 27(1), pp 9–32

Aquatic (2012) Global Hydrological Monitoring Industry Trends. Vancouver B.C., Canada: Aquatic Informatics Inc.

Beran B., Piasecki M. (2009) Engineering new paths to water data. *Computers & Geosciences* 35(4), pp 753–760

Creutin J.D., Obled C. (1982) Objective Analysis and Mapping Techniques for Rainfall Field: An Objective Comparison. *Water Resources Research*, 18(2), pp 413–431

Cutter S. (2003) GI Science, Disasters, and Emergency Management. *Transactions in GIS* 7(4), pp 439–446

Erdin R., Frei C., Künsch H.R. (2012) Data Transformation and Uncertainty in Geostatistical Combination of Radar and Rain Gauges. *Journal of Hydrometeorology*, 13, pp 1332–1346

Göddeke, D. (2007) GPGPU: Basic Math Tutorial. http://www.mathematik.uni-dortmund.de/~goeddeke/gpgpu/tutorial.html. Accessed 6 April 2013

Grebner D., Roesch T. (1998) Flächen-Mengen-Dauer-Beziehungen und mögliche Niederschlagsgrenzwerte in der Schweiz. Zürich: v/d/f Hochschulverlag der ETH

Haberlandt, U. (2007) Geostatistical interpolation of hourly precipitation from rain gauges and radar for a large-scale extreme rainfall event. *Journal of Hydrology*, 332, pp 144–157

Henneböhl, K., Appel M., Pebesma E. (2011) Spatial Interpolation in Massively Parallel Computing Environments. AGILE 2011, April 18–22.

Horsburgh J.S., Tarboton D.G., Piasecki M., Maidment D.R., Zaslavsky I., Valentine D., Whitenack T. (2009) An integrated system for publishing environmental observations data. *Environmental Modelling & Software*, 24(8), pp. 879–888

Hughes D.A., Forsyth D.A. (2006) A generic database and spatial interface for the application of hydrological and water resource models. *Computers & Geosciences* 32(9), pp 1389–1402

Joss J, Waldvogel A. (1990) Precipitation measurements and hydrology. In: *Radar in Meteorology*, Atlas D (Ed). Boston MA, USA: American Meteorological Society, pp 577–606

Kornelsen K.C., Coulibaly P. (2013) Advances in soil moisture retrieval from synthetic aperture radar and hydrological applications. *Journal of Hydrology* 476(7), pp 460–489

Lebel T., Bastin G., Obled C., Creutin J.D. (1987) On the Accuracy of Areal Rainfall Estimation: a Case Study. *Water Resources Research* 23(11), 2123–2134

Lecca G., Petitdidier M., Hluchy L., Ivanovic M., Kussul N., Ray N., Thieron V. (2011) Grid computing technology for hydrological applications. *Journal of Hydrology* 403(1-2), pp 186–199

Lienert C., Weingartner R., Hurni L. (2011) An interactive, web-based, real-time hydrological map information system. *Hydrological Sciences Journal* 56(1), pp 1–16

Lienert C., Weingartner R., Hurni L. (2010) Post-Event Flood Documentation and Communication using a Hydrological Map Information System. *Proceedings of AutoCarto 2010*. Orlando, FL USA

Matthies M., Guipponi C., Ostendorf B. (2007) Environmental decision support systems: Current issues, methods and tools. *Environmental Modelling & Software* 22(2), pp 123–127

Ravindran N., Yao L., Xu L. (2010) A labeled-tree approach to semantic and structural data interoperability applied in hydrology domain, *Information Sciences* 180(24), pp. 5008–5028

Nikolopoulos E.I., Anagnostou E.N., Borga M., Vivoni E.R., Papadopoulos A. (2011) Sensitivity of a mountain basin flash flood to initial wetness condition and rainfall variability, *Journal of Hydrology* 402(3–4), pp 165–178

Ogao P.J., Kraak M.-J. (2002) Defining visualization operations for temporal cartographic animation design, *International Journal of Applied Earth Observation and Geoinformation* 4(1), pp 23–31

OpenGL ® ES Common Profile Specification (2010) Version 2.0.25 (Full Specification). Editors: Aaftab Munshi, Jon Leech http://www.khronos.org/registry/gles/specs/2.0/es_full_spec_2.0.25.pdf. Accessed 4 April 2013

OpenGL ® ES Shading Language (2009) Language version 1.0, document revision 17, Editor: Robert J. Simpson http://www.khronos.org/registry/gles/specs/2.0/GLSL_ES_Specification_1.0.17.pdf. Accessed 4 April 2013

Owens J.D., Houston M., Luebke D., Green S., Stone J.E., Phillips J.C. (2008) GPU Computing. *Proceedings of the IEEE*, Vol. 96, No. 5, pp. 879–899

Owens J.D., Luebke D., Govindaraju N., Harris M., Krüger J., Lefohn A.E., Purcell T. (2007) A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum*, Vol. 26, pp. 80–113

Panziera L., Germann U., Gabella M., Mandapaka P.V. (2011) NORA–Nowcasting of Orographic Rainfall by means of Analogues. *Quarterly Journal of the Royal Meteorological Society* 137, pp 2106–2123

Savina M., Schäppi B., Molnar P., Burlando P., Sevruk B. (2012) Comparison of a tipping-bucket and electronic weighing precipitation gage for snowfall, *Atmospheric Research*, 103, pp 45–51

Sevruk B., Ondrás M., Chvíla B. (2009) The WMO precipitation measurement intercomparisons, *Atmospheric Research*, 92(3), pp 376–380

Solomatine D.P., Wagner T. (2011) Hydrological Modelling. In: Wilderer P. (2011): Treatise on Water Science. Elsevier, Amsterdam, pp. 435–457

Shepard, D. (1968) A Two-Dimensional Interpolation Function for Irregularly-Spaced Data. Proceedings of the ACM National Conference, pp. 517–524

Srinivasan, B.V., Qi H., Duraiswami, R. (2010) GPUML: Graphical processors for speeding up kernel machines. Siam Conference on Data Mining, April 2010.

WebCL Working Draft (2013) https://cvs.khronos.org/svn/repos/registry/trunk/public/webcl/spec/latest/index.html. Accessed 4 April 2013

WebGL (2011) https://www.khronos.org/registry/webgl/specs/1.0/. Specification Version 1.0. Accessed 4 April 2013

WMO World Meteorological Organization (2011) Hydrological Data Management: Present State and Trends. *Operationaly Hydrology Report No. 48*. Geneva

Wueest M., Frei C., Altenhoff C., Hagen M., Litschi M., Schaer C. (2010) A gridded hourly precipitation dataset for Switzerland using rain-gauge analysis and radar-based disaggregation. *International Journal of Climatology*, 30(12), pp 1764–1775

Zehe E., Becker R., Bárdossy A., Plate E. (2005) Uncertainty of simulated catchment runoff response in the presence of threshold processes: Role of initial soil moisture and precipitation, *Journal of Hydrology*, 315(1–4), pp 183–202