Creative and Coordinated Computation

Inaugural address delivered by

Marc van Kreveld

Professor in Computational Geometry and its Application

Department of Information and Computing Sciences
Faculty of Science
Utrecht University

on Friday, January 17, 2014
Utrecht, The Netherlands

Rector magnificus, colleagues, family, friends, ...

I am honored to stand before you today, and have the opportunity to tell you all about my work, my research, and some other things that I will get to later.

It is Friday afternoon, after 4 o'clock, and I realize that most people would call this the beginning of the weekend. However, I am going to be talking about scientific work for a while, so bear with me. I promise that I will not show any formulas, but I will show many illustrations. I truly hope that everyone in the audience will hear something worthwhile in this lecture.

My research is related to *geometry*, a branch of mathematics that deals with objects, like points, lines, triangles and circles, and addresses relations between these objects, like distances, intersections, and containment, or properties like lengths, areas, and angles. My research is also about *algorithms*, the use of automated methods to execute tasks, and the formal study of these methods. The third aspect of my research is *applications*, because points, triangles and circles can have a concrete meaning in applications only, and the application dictates which tasks we want to perform on the geometry.

The title of this speech, "Creative and Coordinated Computation", may seem contradictory: a creative process is often not coordinated, and moreover, coordination may obstruct creativity. We can resolve this contradiction in two ways. Firstly, "creative" also refers to the process of creating, making things. Secondly, "coordinated" can refer to the situation that coordinates have been assigned to things. So we can rephrase the title as "Construction and Geometry in Algorithms". At the same time, I want to emphasize that my research contains a mix of *creativity*, the need for bright ideas based on insight, to solve problems and advance knowledge, and *coordination*, the structured part of research that is required for provability, experimental verification, and reproducibility.

Let me get back to geometry and computers. By the way, I use the term 'computer' throughout to mean the hardware and software together: Not just the machine, also the programs.

It is not so interesting to study how a computer should store points, circles or triangles. Since a computer can store numbers well, it can also store a sequence of three numbers that can be interpreted as the coordinates of a point in the 3-dimensional world. Similarly, a computer can easily store triangles: any triangle is fully determined by three points, namely the

corners, each of which has three coordinates, so with nine numbers we can store a triangle. While a computer can *represent* geometric objects easily, it is important to realize that a computer does not *understand* geometry: it does not see it and it has no intuition for it. In the end, it only has numbers, and can perform operations on these numbers only.

One could also say that there are three levels of abstraction when discussing computers and the representation of real objects: no abstraction at all, which is the detailed real world itself; partial abstraction, which is how people like me represent the real world in computers by using many simple shapes; and full abstraction, or how computers see geometry, purely as sequences of numbers, or bits, if you like.

I just claimed that *representing* simple geometric shapes in computers is not so interesting. What is interesting is how to let computers perform tasks that involve geometry. When you have a geometric task, so, something to be computed on geometric data, how do you get to a sequence of steps that performs it automatically? We refer to geometric tasks as geometric *problems*, and well-designed geometric algorithms are *solutions* to these problems. Since the computer has no understanding of geometry, the intelligence of the solution must come from us, the algorithms designers.

What is also interesting is how to *define* a geometric task for an application. This is the part of algorithms research that precedes the search for an algorithm, a solution. I will spend the better half of this lecture on four specific applications where a geometric task needs to be defined first, and then solved. I will talk about rivers, about flocks of birds, about building facades, and about new types of drawing puzzles. These four examples illustrate the types of problems that show up in computational geometry research, and demonstrate the diversity of its applications.

Rivers flowing uphill

[Research done together with Rodrigo Silveira.]

Let me start with the first example. Suppose that data has been collected on the elevation, height above sea level, in some mountainous terrain. The data consists of measurement points, and every point has a latitude, longitude, and elevation, its three coordinates. With such data, it is possible to reconstruct the elevation everywhere by spatial interpolation, for which triangles can be used.

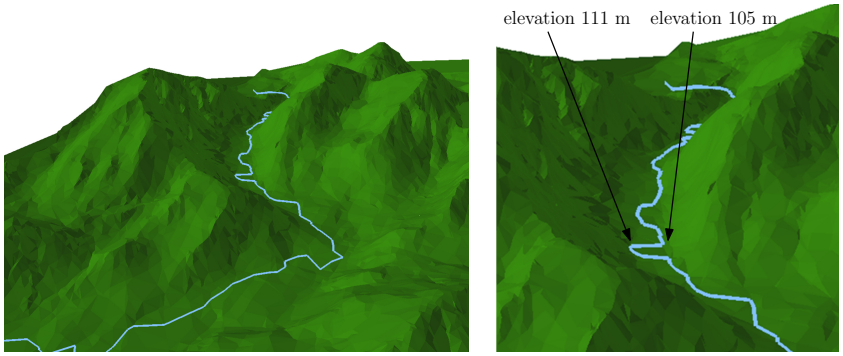

Figure 1: A terrain and a river where, according to the integrated data, the river flows uphill.

Suppose that in another data set, a river shape has been collected, for example by land surveying, and this river is in the same mountainous terrain. Assume the river is a small stream, and its representation is simply a sequence of coordinates that describes the shape.

When we try to integrate these two data sets, we will notice for data sets in practice that they are inconsistent. When we place the river at the measured locations in the terrain, it may be that the river at some places flows from lower to higher elevations, uphill!

While there is a Dutch graphical artist who lets water flow upwards, as shown in his lithograph called "Waterfall", it is generally believed that water follows the laws of gravity.

Another problem that occurs in the integrated data set is that some rivers may be running halfway up a hill slope instead of on the valley floor. If the river is not in a ditch-like local structure, the water would slide down, because it will follow the path of steepest descent.

In any case, we have two data sets and we do not know which one is incorrect, but still we would like to have a single data set that integrates the elevation and the river in a way that makes sense, meaning that the river flows from higher to lower elevations only, and it is situated at a valley floor.

To integrate the data we may for instance change the elevations of points in the terrain. They were measured to begin with, so they already had

some imprecision. Changing elevations a little does not necessarily make the terrain less correct, and we can hope that the two data sets become consistent.

Research papers in the scientific area of geomorphology describe a rather crude solution to this problem called stream burning: simply reduce the elevations of all points that lie on the river until it lies in the just created canyon-like thing, and also take care that the flow is from higher to lower elevations. When you lower the elevations of two points with a river edge in between, this edge will always become a ditch or valley-floor type edge. When you lower the one point even more, the flow will also be in the direction that corresponds to the river flow direction.

The effect of this approach is perhaps technically a terrain with a river that lies at a place where a river would be, but it is certainly not natural: the approach may create a crevice, or fracture in the terrain, just to ensure that the water flows downhill in the river and cannot escape from its river bed. The initially measured elevations of some points of the terrain must sometimes be changed drastically to get the river embedded. The terrain that results contains artifacts that are created by the method, but which do not occur in reality.

While it is simple and efficient to lower the elevations where the river should be only, there is no good reason to limit ourselves this way. We could also raise or lower the terrain around the river. This means that we have many more data points that we can manipulate, so potentially we could do more damage to the originally measured data. To avoid this we will minimize the total change in elevation of all points that are lowered or raised.

This is our definition of the geometric problem: given a terrain and a river, change the terrain elevations to embed the river properly, but with minimal change in elevation of the terrain points. It turns out that this optimization problem can be solved with a very general algorithmic technique called linear programming.

We performed a quantitative analysis to test how many points had their elevation changed, and how much these elevations changed in total, for this basic version of the problem and for various extensions. We also used visual inspection to see what the changed terrain looks like, for example to look for artifacts. The result: better terrain models.

Animals traveling in groups

[Research done together with Kevin and Maike Buchin, Bettina Speckmann, and Frank Staals.]

Let us proceed to the second example.

Many species of animals have been tracked using GPS, Global Positioning Systems, and nowadays there are large collections of trajectories, or movement paths, of animals. These trajectories are sequences of locations with a time at which the animal was at that location. If locations are measured frequently enough, then we can assume that the animal moved in a straight line from the one measured location to the next one. With a data set consisting of trajectories of many animals of a type, say, birds, we could ask ourselves whether we can define and compute groupings of birds. For example, we should be able to determine which birds travel in groups, which groups merge with other groups, or split up into smaller groups, how long a group of birds typically stays together, and so on and so forth. These types of questions are of interest to ecologists.

Of course it is also possible to track people and analyze their trajectories, which has applications in the social sciences and in identifying security threats.

Before we can start to compute anything, we must first define what a group really is. Simply put: a group is a subset of the birds that are together during some time. A group must have sufficiently many birds to call it a group, the distance between the birds must be small enough to call it a group, and they must be together long enough to call it a group. So we use a distance parameter, a time parameter, and a count parameter to define a group. It appears that this is the minimum we need to have a workable definition.

In the small animated example, assume that a group must have 3 birds to really be a group, and stay together for at least one second. We see a group of red birds and a group of blue birds. Our definition ensures that we do not have a group of six red and blue birds, because they are not together long enough. So there is also no merge and split of two groups, which is according to our intuition.

In the second animation, we see two groups that merge and then split up again. We see a group of blue birds that are together the whole time, a group of red birds that are together the whole time, and a group of six red

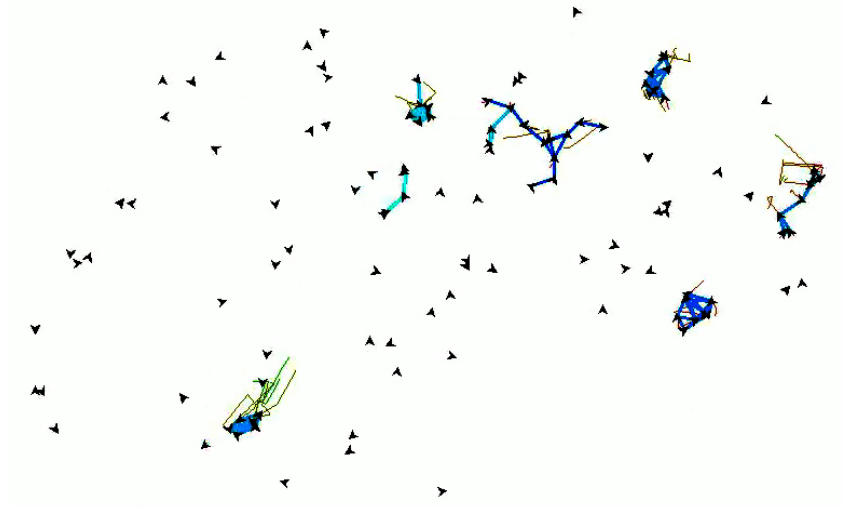

Figure 2: Snapshot of deer movement data and the computed grouping. Since it is only a snapshot, grouping cannot be interpreted as being close in the image.

and blue birds that are together briefly.

In this last animation, the six flight paths are exactly the same, but four of the birds have been colored green. The current definition states that the green birds also are a group, and i fact, every subset of three, four or five birds, is a group, because any such subset of the six birds also stays together long enough and has size at least three. The animation would give 42 different groups, to be precise, but this is not at all according to our intuition. Therefore, we will restrict the groups to the ones that are maximal in group size or duration, so, the ones whose existence is not directly implied by other groups. We can prove that this reduces the total number of groups from exponentially many to polynomially many.

With the proper definition we have solved half of our problem of obtaining groups, their merges and their splits. For the computing itself we use various techniques from algorithm design and from mathematical topology, but I would get too technical if I would elaborate on that.

Based on our definition and algorithm, we can make an implementation and

test it on movement data. We chose to use simulated bird flights, where the flight path of each bird is influenced by the flight of the birds close by and within sight. In the video we see the simulated bird movement data, and birds are connected by lines when they form groups according to our definition. When they form a group, they also leave a blue trail. We do not have a ground truth, there is no knowledge of what the real groups would be, and biologists have not provided any exact definition of a group as far as we know, but our results are plausible: we do not see birds that should have been deemed a group, nor do we see groups that should not have been deemed a group.

Furthermore, the three parameters that define when a group is really a group appear to have the expected effect. For example, requiring a group to stay together longer means that fewer groups will be found; the short-duration groups are now excluded.

Reflections, I

Before I continue with the next two examples, let me reflect on the story so far. When a specific task needs to be automated, this task may not be specified very precisely, so the first objective is to define exactly what must be done. This is a general concern in computing science, and holds just as well in software technology and artificial intelligence. After defining, we still need to solve the problem, that is, design an algorithm that performs the specified task. Here we need deep knowledge of algorithmic techniques.

After designing an algorithm, we are not done. The *foundational* next step is to mathematically analyze the algorithm. We must prove that it is correct, and therefore always solves the problem on valid data. We must analyze its efficiency, where we establish how well the solution scales to larger and larger data sets within a mathematical model of computation.

The *applied* next step is to implement the algorithm and test it on suitable data. We can then see whether the solution produces results that we expected to get.

In the two steps of defining and solving, obtaining geometric intuition plays a large role. This means that my research involves making sketches, seeing patterns, exploiting general truths about the specific geometry of the problem, and understanding the consequences of a definition. And it means that I often play with pens and pencils, and draw situations to acquire that

intuition. When I look back my whiteboard or sketch pads I used earlier, I often don't remember anymore what the little sketches were supposed to mean. They look mostly like the drawings of a toddler.

From points to building models

[Research done together with Thijs van Lankveld and Remco Veltkamp.]

The third example concerns the construction of 3-dimensional models of buildings in the computer from data measured in the real world.

Those of you familiar with the 19th century painting style called pointillism know that when you see enough points with certain colors, you can reconstruct a real-world image in your mind, consisting of planes and other shapes. One dimension up, this same task is an important task for computers. Laser scanning is a technology where the return time of a tiny laser beam is measured, giving one point with three coordinates that is known to lie on the surface of some object. Laser scanning produces many thousands, up to millions of points per second, leading to huge sets of points that all lie on the surface of the objects in our world.

In the picture you can see such a huge point set, but from a distance so it is not so easy to see separate points. We can identify bigger and smaller buildings, and trees and boats in the foreground. When we zoom in on the buildings, we start to see that we are only looking at many separate points.

From these points, we can try to reconstruct building facades, roofs, gutters, street signs, traffic lights, and everything else. We need to go from points to planes and sides of planes. A proper 3-dimensional model of a simple house, for instance, consists of a representation of the four sides and two roof planes, fifteen corner edges and ten corner points. Such a representation is much more compact and useable than storing the millions of points the lie on house.

Reconstructed virtual cities are important for the training of emergency response teams, which can run disaster scenarios and try to respond in the best possible way. This type of application is also called serious gaming.

The process of building reconstruction typically has the following steps. First, find many points that lie in the same plane, which could possibly be a roof, a facade, or the plane of the street level. Do this many times, to get many planes with many points in each.

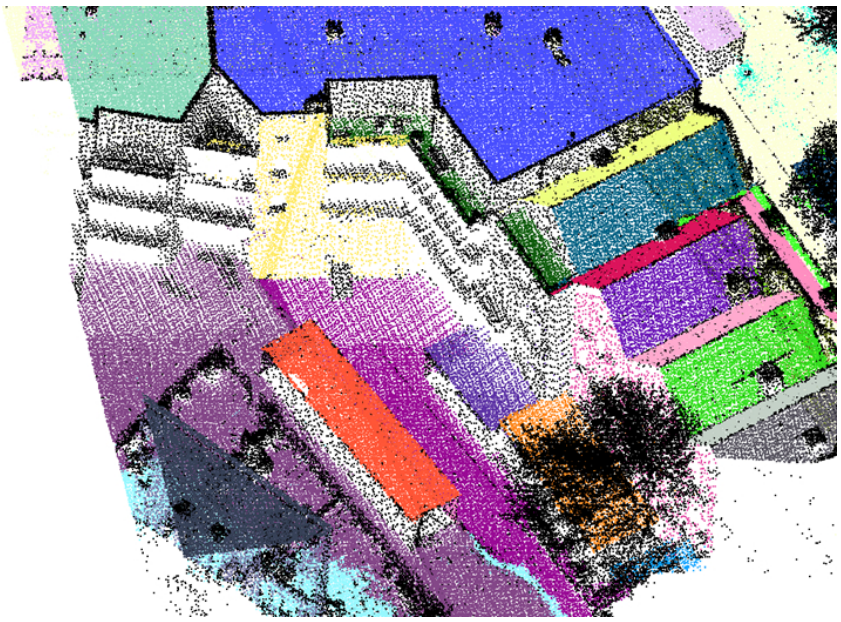

Figure 3: Downward view on a laser-scanned point set where points in the same plane have been given the same color. The black points near the lower right are from a tree. Flat roofs (top) and sloped roofs (right) can be seen.

In the image of the point set, the points lying in the same plane have been given the same color. The black color is used for points that do not lie in a plane with many other points. In the foreground we see the black points forming a tree.

Second, consider these planes with many points separately, and try to find a reasonable boundary of the points in each plane. This way we want to get the outline of a roof plane or facade, for example. This step is often done using alpha-shapes [5], which are most easily imagined by letting the computer roll a circle along the outside of the points, and connecting pairs of points that are on the circle at the same time. Of course the computer cannot roll a circle, but the process can be described by an algorithm that refers to the coordinates of the points only.

This shape reconstruction method was not my idea, and I don't want to

be the next in line after Stapel, Smeesters, Poldermans, Bax, and Nijkamp, so let me say clearly that alpha-shapes were invented by Edelsbrunner, Kirkpatrick and Seidel in the early eighties.

After the points in each plane have some boundary, we have a collection of facets that are all placed somewhere in the 3-dimensional space, but they are not nicely connected into a closed object. There will be gaps between the facets everywhere. In reality, the situation looks something like what is shown simplified in the figure.

Instead of trying to fill the gaps between the facades, we will try to find the lines that are a shared side of two facades. These lines come from the intersection lines of all the planes that contain many points. It is, however, not clear beforehand which of the intersection lines are useful and which ones are not.

When we consider the construction of a single facade boundary in a plane again, we now have additional lines that are these intersection lines. They are likely boundaries of the facade, but there will also be intersection lines that cannot be used at all to construct this facade boundary. They come from planes whose facades are not adjacent to the one we are now handling.

In our research we developed a reconstruction method for a boundary when both points and lines are given, but the lines are only suggested boundaries. They can be used, partially used, or ignored, depending on the points nearby. The effect is that adjacent boundaries usually match up nicely along a seam, an edge of a building.

Let us now go to the 3-dimensional scene again. Here we see the result of the alpha shape boundaries, and here we see the result of our shapes bounded by points and lines. We can see the desired effect at the numbers 1 and 2 and many other places in the figure. At number 3 we see that the alpha-shape is still used when there are no useful intersection lines to bound a point set. At number 4 we see an artifact where points were measured behind the facade of the dark green points, on the floor inside the building. This is caused by windows, which are sometimes transparent for laser scanning, and sometimes not.

In this last example, we never defined what the desired 3-dimensional model is for a 3-dimensional point set. Instead, we described a process that will give a 3-dimensional model. If we don't define the geometric problem precisely, we cannot prove correctness of a solution anymore.

In fact, we don't know how to define the reconstruction problem in such a

way that solutions would produce the desired results in practice. The reason is that the data suffers from imprecision, varying point density, and missing parts due to occlusion or transparent surfaces. An effective reconstruction method must deal with all of these complications as well. Since this is very hard, no fully automated building reconstruction method exists for buildings based on laser scanning data. All existing models of sufficient quality are made at least partially by human modelers.

Connect-the-dots

[Research done together with Mira Kaiser, Tim van Kapel, Gerwin Klappe, Maarten Löffler, and Frank Staals.]

The fourth and final example is perhaps the easiest. We look at a geometric task that can be imagined on a single sheet of paper, and is in fact usually given as a puzzle on a single sheet of paper. The task for humans is: connect the dots in the given order to produce a drawing. Connect-the-dots puzzles are also called follow-the-dots, and they exist in widely varying complexities.

Obviously we do not want to let the computer *solve* connect-the-dots puzzles, but we want to let it *generate* such puzzles from a drawing. So, given some representation of a drawing, how do we generate points and associated numbers so that the solution of the puzzle looks very much like the initial drawing?

We can set up a problem definition, for example by allowing the solution of the puzzle to be at most 4 millimeters away from the original drawing, and under this restriction, minimize the number of points used in the puzzle. For example, the red points would not be allowed to be consecutive in a puzzle, but the green points would be acceptable.

As it turns out, this formulation of the problem can be solved with a variation of an old algorithm developed by Imai and Iri [6], more than 25 years ago. Their motivation was to reduce the number of control points to describe a shape, because memory space was expensive back in those days. Today, we can use the same method for a completely different purpose.

While an old algorithm solves the well-known connect-the-dots generation task, we were not very happy with the type of puzzle itself. For one thing, just connecting points in a given sequence is not very interesting for a puzzler. In principle the puzzle must be drawable by a single line, otherwise we have to pre-draw pieces or use other tricks. Finally, in the solved puzzle,

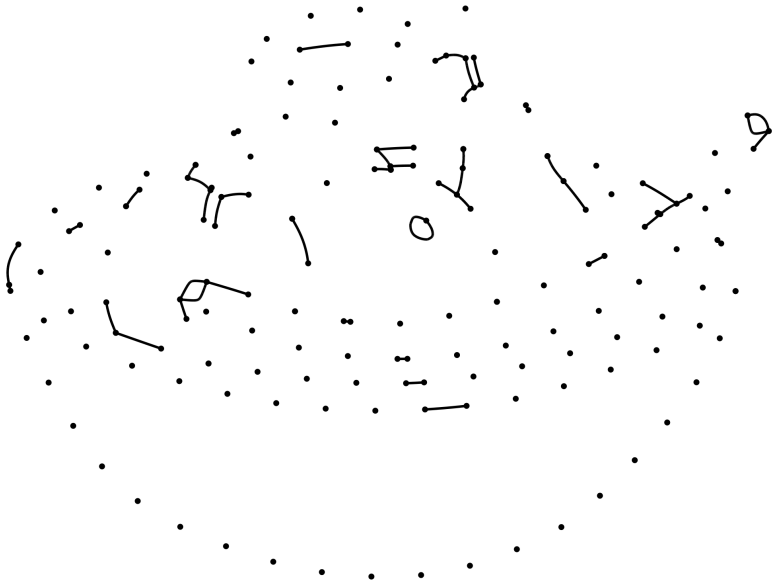

Figure 4: Connect-the-Dots puzzle where point pairs at one specified distance must be connected.

the finished drawing, the numbers are still present, and make the drawing somewhat ugly.

Now we have to go beyond *defining* the task: we have to *invent* a new drawing puzzle type. Suppose there exists a puzzle type with points but no numbers to specify how points are to be connected, what rules could a puzzler follow to connect them and produce the original drawing from which the points were taken? Such a rule must be simple, well-defined, and easy to check, so that the puzzler knows when to draw a line between two points and when not.

For example, we could use the rule: connect two points if and only if their distance is exactly equal to the diameter of a 10 cents coin. Any two points that are not at this diameter distance, should be clearly at a different distance, otherwise the puzzle will be confusing. A puzzler can then slide a 10 cents coin over a piece of paper to check distances and if right, draw the line.

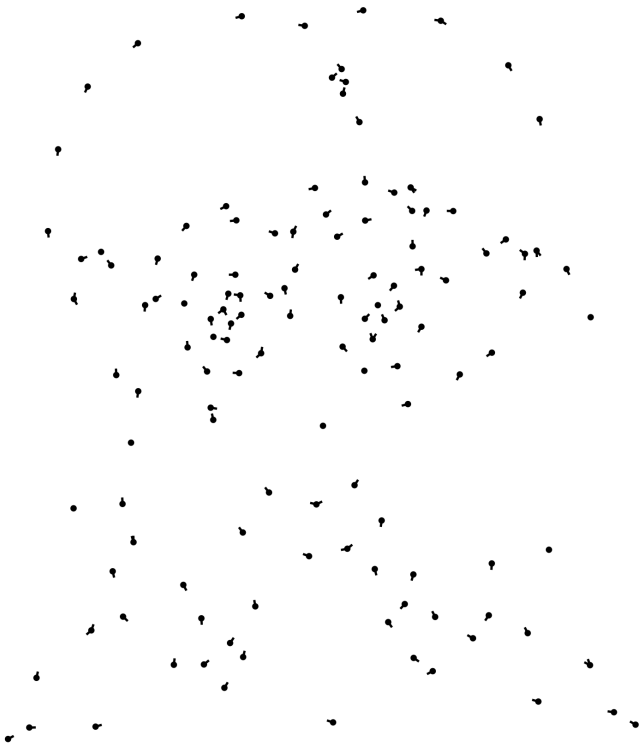

Figure 5: Connect-the-Dots puzzle where points are connected to other points in the indicated direction.

As another example, the points could have little direction indicators and we could use the rule: connect a point in the direction of its indicator to the closest point in that direction. To avoid confusion of the puzzler, there should be only one clear candidate point for each indicator; other points must be clearly in a different direction, or clearly further away, otherwise the puzzle is confusing. Here you can see the solve puzzle.

A third example is a puzzle with colored points, and each point must be connected to the nearest point of the same color. Again we must define and use a notion of ambiguity.

After inventing these new puzzles, we define the geometric tasks involved,

where an exact definition of ambiguity is needed. Then we develop the algorithms and analyze them formally, and implement and test them.

In this example, we saw that the two stages of defining and computing can sometimes be preceded by a stage of invention. The connect-the-dots puzzles without numbers are a new type of puzzle. The ideas are not difficult, once you realize that such puzzles can exist at all.

Reflections, II

We have seen four quite different situations where geometric tasks need to be solved by computers. The research area of computational geometry helps to provide ways to define the problem suitably, and then solve it using a variety of algorithmic techniques. The applications we saw involved terrain and river data integration, animal motion analysis, city reconstruction from point sets, and connect-the-dots puzzles.

As said before, the word "creative" in the title of this speech refers to the process of creating: computational geometry researchers create algorithms, and algorithms create 3-dimensional models, puzzles, or knowledge from data.

There are many other applications where computational geometry plays a role that are worth mentioning, like computer graphics, robotics, manipulation and grasping, spatial data analysis, and automated cartography.

Virtual worlds

While the more fundamental, algorithmic way of dealing with geometry is my personal focus, I am part of a group of colleagues who together form the division Virtual Worlds. These colleagues do just as exciting things as I in their research, possibly even more exciting.

For example, in computer graphics, one can do automated object construction based on procedural rules instead of actual data. When the red shape is given, many variations and compositions can be made automatically, as shown in grey.

For the realistic animation of virtual characters, we may need to model muscles and joints, and run simulations to get realistic motion.

We can also capture motion data in the motion capture lab, and use this to

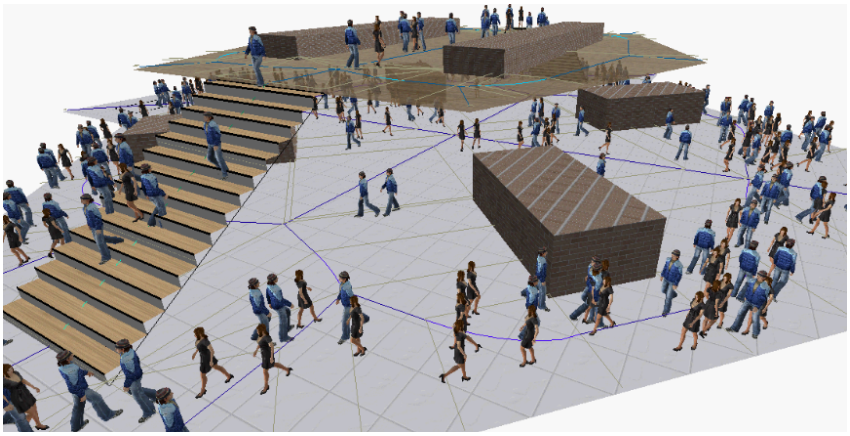

Figure 6: Virtual world with crowds moving through it (from [10]).

understand motions of real people and generate motions of virtual characters.

We study the movements of crowds in complex 3-dimensional environments, a topic of major importance in safety and disaster management.

These are great themes of research: it is often based on solid theory, it is challenging to us researchers, it can be illustrated well with images and movies, and it lays a foundation that can be used by organizations and companies that wish to advance the possibilities of their software. Here we are thinking primarily of serious gaming and entertainment gaming companies.

Not completely in jest, I'd like to say that together, we will make the virtual world a better place.

Ius promovendi

Before I close with words of thanks I want to make a political statement, just because this is my opportunity to speak in public to many important people from Utrecht and other universities. It concerns the Ius Promovendi, the right to promote a PhD candidate, or in more fancy English, the right to bestow doctorates. By a Dutch law, this right is reserved to full professors only.

Consequently, a senior researcher who successfully acquired research funding, trained and supervised the PhD student for four years, and helped with set-up and correction of the thesis, is not allowed to have the honour of being the promotor if he or she is just an assistant professor or associate professor. Instead, a full professor who did not acquire the funding and often was not involved in the supervision, will have this honour, which is exactly for this reason a questionable honour. Full professors put the names of these PhD holders on their CVs as people they promoted. Which is correct, but it isn't right.

Before I became full professor, I supervised seven PhD students and acquired the research funding for most of them. There was no intervention of a full professor, basically because the full professor trusted me as a supervisor and researcher. I take pride in supervising PhD students, and I wanted the corresponding honour of being promotor.

Now I am full professor and I can be promotor of my own PhD students. The problem seems solved for me, but this is not true. Now I will have to act as the promotor of PhD candidates that really should have someone else as their promotor. I will be embarrassed each time this is going to happen. The Netherlands should change its law on higher education and correct this injustice.

While this is just a personal plea, The Young Academy has summarized all of the real reasons why the *Ius Promovendi* must be extended to professors who are not full professor. I gladly refer to their report and column.

I do not suggest that the Ius Promovendi should be given to everyone. A researcher should be qualified to bestow doctorates. Fortunately, Utrecht University already as a suitable replacement, namely the Senior Qualification Research. It is generally better when honorary rights that include a responsibility are coupled to a qualification, and not to a job.

It is true that universities face bigger problems these days, for example severely reduced research funding, and loss of trust in universities due to the lack of scientific integrity of certain professors. However, this cannot be an argument to keep a flawed system in place.

I will be proud of my university if it will take a leading role in realizing the change of the law regarding the *Ius Promovendi* in the Netherlands.

Words of thanks

I wish to close with words of thanks to many people.

First of all, and starting with the early years of my life, I thank my father, mother and brother. My father stood here in my place, just over 30 years ago, with an inaugural speech entitled: "Social Psychology, what is it good for?". He passed away nearly two years ago.

Continuing with the early years of my career, I thank Mark Overmars and Mark de Berg. Mark Overmars was my promotor. He certainly deserved the honour, and I am honoured to have him as my promotor. Mark de Berg was a great colleague during my PhD time, but also long after that.

I thank my former and current PhD students René van Oostrum, Tycho Strijk, Sergio Cabello, Iris Reinbacher, Esther Moet, Rodrigo Silveira, Maarten Löffler, Anne Driemel, Thijs van Lankveld, and Frank Staals. I loved working with all of you. I also thank all other collaborators in joint research projects and papers.

I thank Utrecht University for providing me with a career and the opportunity to pursue the research I love and have been talking about. Thank you, Bert van der Zwaan, rector of this University, and Gerrit van Meer, dean of the Faculty of Sciences, for having faith in me and supporting the latest career step to full professor. I also thank Remco Veltkamp and Mark Overmars for their essential role in this process.

I thank all the colleagues of the department of Information and Computing Sciences for the great working environment it provides, and the division Virtual Worlds for sharing a passion for geometry-related computing research.

I thank the beadles Helga Hoiting and Paulien van der Veer, as well as Rita Jansen, for helping me organize this special day.

The most important people in my life are my wife Bettina and my children Ivo and Lucas, and I am grateful that we can share so many things in our lives.

Last but not least, I thank you, the audience, for being here today and lending me your ears.

Ik heb gezegd.

References

[1] De Jonge Akademie. Rendement van Talent - Aanbevelingen voor motiverend en stimulerend loopbaanbeleid. Advies, 2010. `http://www.dejongeakademie.nl/dejongeakademie/Content/documents/AdviesRendementvanTalent.pdf`.

[2] Arianna Betti, Peter-Paul Verbeek, and Ingrid Robeyns. Bezwaren rond het ius promovendi, promotierecht aan herziening toe. Column Wetenschapsbeleid, De Jonge Akademie, 2011. `http://www.dejongeakademie.nl/Pages/DJA/31/578.bGFuZz1OTA.html`.

[3] Kevin Buchin, Maike Buchin, Marc van Kreveld, Bettina Speckmann, and Frank Staals. Trajectory grouping structure. In *Algorithms and Data Structures - 13th International Symposium, WADS 2013. Proceedings*, volume 8037 of *Lecture Notes in Computer Science*, pages 219–230. Springer, 2013.

[4] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry – Algorithms and Aplications*. Springer-Verlag, Berlin, 3rd edition, 2008.

[5] Herbert Edelsbrunner, David G. Kirkpatrick, and Raimund Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–558, 1983.

[6] Hiroshi Imai and Masao Iri. Polygonal approximations of a curve formulations and algorithms. In *Computational Morphology*, pages 71–86. Elsevier Science, 1988.

[7] Maarten Löffler, Mira Kaiser, Tim van Kapel, Gerwin Klappe, Marc van Kreveld, and Frank Staals. The connect-the-dots family of puzzles: design and automatic generation. Manuscript, 2014.

[8] Marc van Kreveld and Rodrigo I. Silveira. Embedding rivers in triangulated irregular networks with linear programming. *International Journal of Geographical Information Science*, 25(4):615–631, 2011.

[9] Marc van Kreveld, Thijs van Lankveld, and Remco C. Veltkamp. On the shape of a set of points and lines in the plane. *Computer Graphics Forum*, 30(5):1553–1562, 2011.

[10] Wouter van Toll, Atlas F. Cook IV, and Roland Geraerts. Navigation meshes for realistic multi-layered environments. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 3526–3532. IEEE, 2011.