TERRAIN MODELING FOR MOBILE ROBOT CONTROL USING CROSS-SECTION CURVES IN NESTED MULTI-RESOLUTION GRIDS

Pryanichnikov Valentin^a, Shipovalov Egor^a

^aKeldysh Institute of Applied Mathematics Russian Academy of Sciences, Miusskaya sq.4, Moscow, 125047 Russia

Abstract

A digital terrain model is a key component of field mobile robot's control system. However, the principles of surface representation on which today's popular models are based suffer from a number of deficiencies. These include excessive memory requirements, computational expensiveness of basic operations, inability to adequately represent certain types of surfaces and lack of semantic information.

In this paper we propose a method of describing terrains based on a fusion of nested multi-resolution elevation grids and cross-sections of common surface elements. The resulting model, while retaining positive characteristics of traditional elevation grids, offers lower memory consumption, faster height lookup times and natural support for vertical surfaces. It also contains basic semantic information and can be defined in a compact human-readable format.

The work is supported by University of Zadar, Croatia, DAAAM International Vienna (Vienna, Austria), International Institute for Educational Technologies of Russian State University for the Humanities, «Sensorika» International Laboratory, Keldysh Institute of Applied Mathematics of Russian Academy of Sciences, CEC IIA (Vienna, Austria) under the agreement of 04.06.2014, and partially with RFBR grants № 13-07-00988a, № 13-07-01032a

Keyword: autonomous mobile robots; field robots; digital terrain model; cross-section; control system; SLAM

This Publication has to be referred as: Pryanichnikov, V[alentin] & Shipovalov, E[gor] (2016). Terrain modeling for mobile robot control using cross-section curves in nested multi-resolution grids, Proceedings of the 26th DAAAM International Symposium, pp.0417-0425, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-07-5, ISSN 1726-9679, Vienna, Austria
DOI:10.2507/26th.daaam.proceedings.056

1. Introduction

1.1. The problem of digital terrain modeling

A digital terrain model (DTM) is a statistical representation of the continuous ground surface by a large number of selected points with known coordinates so the height of any point can be found from the DTM. [1] A digital terrain model (DTM) is a key component of every field robot's control system. It provides the framework in which the robot's core tasks of navigation, collision avoidance and path planning are solved. Hence, the quality and efficiency of this model has a critical impact on overall performance of the mobile robot.

The problem of terrain representation has been extensively researched in the past, both in the context of mobile robotics and in a much wider field of Geo-Information Systems (GIS). Many of the approaches that emerged in these separate fields share the same principles, which fall into four main categories: point-based, triangle-based, grid-based and a hybrid approach combining two of these.

None of these approaches attempt to model terrains in a full 3D space due to high memory and computational requirements. Instead they reduce the model to the so-called 2.5D dimensionality: replacing the Z coordinate with elevation value. This greatly simplifies the model and reduces its computational requirements at the expense of support for multi-level structures.

1.2. Mobile robot control system as the context of application

To better understand the role of a DTM in mobile robot control and the requirements for it, it is necessary to consider the context in which it operates.

Component name	Measurements $X_s \rightarrow L_s$	Navigation $X_s \rightarrow X_n$	Control $U \rightarrow X$	Interface $M_i \rightarrow X_p$
A Online algorithms, software	(6) A_s input: $L_s(t, tdt, \dots, tkdt)$ output: $X_s(t)$	(7) A_n input: $X_s(t)$ output: $X_n(t)$	(2) A_u input: $X_p(t) X_n(t)$ output: $U(t)$	(1) V_i supervisor input: M_i, X_n output: $X_p(t)$ $M_i \square$ human operator Service interface: access to virtual model settings, drivers and internal data models
V Virtual reality algorithms or drivers for real devices	(5) V_s sensors input: $V_e, X(t)$ output: $L_s(t)$	(4) V_e external environment input: M_i output: $Q(t)$ data stream $\langle t, X_s, X_n, U, X_p \rangle$	(3) V_r robot input: $V_e, U(t)$ output: $X(t)$	
M Models and data structures in mobile robot control system	M_s internal model of the sensing system	M_e internal map representing V_e for the control system	M_r internal model of the robot's dynamics	

Table 1. Components of sensing and control system for a real robot, mechatronics device or its software model

Onboard software implements control algorithms A governing the actions of the mobile robot. Goals X_p are set by a human operator M_i via the interface V_i or autonomously by the robot itself. The mobile robot, its sensors and environment in which it operates are parameterised and described in the respective structures and data models $M(M_r, M_s, M_e)$ on which control actions U are based. Note that M_e , like V_e , can be constructed from real sensor data as well as virtual.

Raw sensor data L , control signals, robot parameters, results of filtering the sensor measurements X_s and their interpretation by the navigation subsystem X_n , make up the operational environment Q . The structure of such a system implemented in a distributed computational architecture is shown in Table 1.

Research software components V_r, V_s, V_e (complete models of sensors, the robot and external environment) are conceived during development and debugging of the robot control system or while testing the whole complex in emulation mode. In the latter case they represent virtual (programmatically constructed) devices or objects. Whether real or virtual, they are accessed via the same driver interfaces V [10].

Here t is current time measured with the step dt ; $tk = tk dt, k=0, 1, \dots$, - time series of system observation events or issued control commands. Component algorithms are synchronised with helper agents or in a direct cycle with

possible time discrepancy between supervisor data and sensed data taken into account. The main control cycle is the sequence: (1),..., (7), (2),..., (7),...

Alternatively, this same workflow can be written as a sequence of expressions evaluated in every work cycle:

$X(t) = Vr (Ve, U(t))$, $Ls(t) = Vs (Ve, X(t))$, $Xs(t) = As (Ms, Ls(t, t-dt, .. tk dt))$, $Xn(t) = An (Me, Xs(t))$,
 $Xp(t) = Vi (Mi, Xn)$, $U(t) = Au (Mr, Xn(t) Xp(t))$, moving on to $X(t)$. [8]

Adaptive behaviours of the mobile robot are constructed based on expert rules. Such rules are used to select algorithms maximising overall efficiency based on pre-defined criteria and fill sensor measurement gaps using logical filters that abstract away time discrepancies in time-series data.

This allows the mobile robot to remain operational regardless of inconsistencies in sensor data and estimate its own location and trajectory with accuracy sufficient for locomotion tasks. The following approach is utilised to construct expert rule schemes. Let $\Psi_m(.)$ be a system of rules for threshold sampling of parameter $(.)$ on a number of intervals $0, ..., m$; $Resm(.)$ be the distance to the closest smaller threshold value.

Now we can construct discrete operational environment $Q = Me(i, j)$ defined as a vector field, where $i = \Psi_m(Xn|x)$, $j = \Psi_m(Yn|y)$, X, Y, Xn, Yn robot's planar coordinates and their navigational estimations, $Q = \langle Z, \Psi_m(Xs), \Psi_m(U) \rangle$ - two-dimensional data stream (environment map and array Z storing history of control actions U and measurements Xs, Xn). This forms the basic data model for simultaneous localisation and mapping (SLAM) and similar tasks. In addition, recursive application of rules $\Psi_m(Resm(.))$ allows for multi-level threshold sampling.

2. Existing solutions

2.1. Point-based DTMs

In a point-based DTM, elevation of an arbitrary point is taken from such of some individual sampling point. Most often the geometrically closest point is used, determined with Voronoi diagrams. Alternatively, a regular grid (height map) is utilised sometimes. A major problem with this approach is the large discontinuities in the surface resulting from abrupt height "jumps" between areas of neighbouring sample points. This does not allow for adequate representation of tilted surfaces limiting point-based DTMs to a few niche implementations.

2.2. Triangle-based DTMs

This problem is overcome in triangle-based DTMs. In these, every point of the surface belongs to a triangle formed by three sampling points. Its elevation can then be calculated using equation of the plane defined by that triangle. Because the triangle is the basic geometric shape into which all other angular shapes can be decomposed.

Fig. 1. Representation of a slope in a point-based regular DTM. Note the staircase-like structure representing a smooth surface.

Fig. 2. Representation of the same slope in a triangle-based regular DTM. Note the angled wall on the left representing a vertical surface

This model is easily adaptable to any pattern of sampling points (in case of no pattern the triangles are formed using Delaunay triangulation).

Thanks to their flexibility and ability to represent any surface with adequate precision triangle-based DTMs became the most popular in general use. However, they do not lend themselves well to intensive data manipulation. Referencing arbitrary triangles in the mesh or quickly finding neighbouring triangles becomes expensive in large models which makes them less suitable for robotics applications.

2.3. Grid-based DTMs

However efficient the DTMs described above are, each of them has its issues when it comes to robotics applications. For point-based DTMs it is the inability to adequately represent smooth tilted surfaces. They have to be approximated with stairway-like structures that may only work adequately with at very high resolutions making them difficult to work with.

Triangle-based DTMs can represent such surfaces easily, but they cannot handle vertical ones. A vertical surface requires two elevation values for the same points in the X-Y plane whereas the 2.5D model can keep only one. This, of course, applies to grid-based DTMs as well. Another problem with triangle-based DTMs which was already pointed out is their computational inefficiency. Finding neighbouring triangles, referring to triangles, extending and adjusting a triangular mesh are relatively non-trivial and expensive operations.

Last but not least, in all aforementioned DTMs the data is largely unstructured. In actual terrain there are usually recurrent patterns and primitives such as slopes, hills, pits, which are very helpful for assessing traversability and path planning.

Fig. 3. Representation of the same slope as in a grid-based regular DTM. The grid replaces the triangular mesh but the angled wall is still in place..

Fig. 4. Example terrain represented in the proposed model

Accounting for these in a terrain model would make it much more compact and expressive, but little such work has been done with a notable exception of [14].

3. The proposed solution

3.1. Introduction

The model we are proposing is free from these deficiencies. It supports vertical as well as tilted surfaces, contains much less redundant information and makes extensive use of higher-level features when describing terrains. This is achieved by extending the grid model with two additional techniques: nesting and cross-sectioning.

Another extension we add is the micro-terrain: a descriptor of physical qualities of terrain surfaces that is particularly useful in realistic modelling of robot sensing and locomotion. In our model the micro-terrain describes surface acoustic qualities and its custom grip coefficient with respect to the robot's wheel. Optical qualities can be added as well as can be added as well.

3.2. Multi-level terrain representation

The nesting of grids allows modeling of terrain on several levels of detail. Every grid cell (a square formed by four grid points) can contain another grid, and so on. Nested grids do not have to have the same number of cells. The

model places no limit on the number of levels of terrain representation. We will consider a minimally sufficient example of two regular levels with a third special level of micro-terrain.

The top level which we call the macro-terrain is represented by the coarsest grid covering an entire area described by the model. The largest terrain features such as slopes, ridges or hills are represented as macro-terrain. The next level that we call the meso-terrain represents terrain features of sizes comparable to the size of the robot. These can include common terrain features as well as human-made objects or large vegetation.

The third and the last level - the micro-terrain, is special and is handled in a much simpler way than the normal ones described above. It describes not terrain features, but terrain surfaces such as sand, pebble or grass. Its job is to help in modelling of robot sensing as well as locomotion by describing relevant physical qualities of the surface.

3.3. Description of terrain primitives with cross-section curves

The nesting of elevation grids helps to reduce memory footprint by applying finer resolutions to the more articulate parts of the terrain (such meso-terrain in our example model) and using a coarser grid where it is sufficient (macro-terrain). However, it leaves in a lot of redundant information as real-world terrains are highly repetitive, especially in inhabited areas.

In our model this problem is addressed by maintaining a dictionary of common terrain elements which we call terrain primitives. Each dictionary item defines a primitive by storing two cross-section curves: along X and Y axes. The surface of the primitive is then formed by moving one cross-section curve along the other.

Any cross-section curve can be construed as a continuous single-value single-variable function. Such functions can be defined in analytical form as well as explicitly. For example, we can define a hill primitive by specifying the cosine function for both cross-section curves.

This method works well for curved surfaces. Note that for consistency we confine all curve functions to the interval from 0 to 1. Alternatively, cross-section curves can be defined as numeric vectors. This works best for angular shapes such as walls or other man-made objects. In this case, values in the supplied vector are considered to be curve function values distributed evenly in the 0...1 range. Intermediate values are found by interpolation [2].

3.4. Creating terrain elements from cross-section primitives

Instances of terrain primitives defined in the dictionary with the above method can be placed in terrain cells. By terrain cells we mean the squares formed by sets of four elevation points of the grid. In our example model, one primitive comprises the macro-terrain which is in turn populated by more primitives forming the meso-terrain. An instance of a primitive that is placed in the terrain we will call a terrain element. When a terrain element is created, a number of modifications can be applied to the terrain primitive to increase flexibility of the model. First is vertical scaling. The scaling factor can be explicitly specified with a numeric parameter in the terrain element's definition or calculated by default to keep the element's height proportional to the level of terrain it is placed into.

In our example model, if no vertical scale is specified for an element of meso-terrain it will be scaled down to one tenth because the side of the cell it is placed in is one tenth of the overall length of the terrain. The scaling factor can be negative. In this case, the primitive is turned upside down forming a recess in the terrain rather than an elevation. A hill primitive can be reused in this way to make a pit and so on.

Elements can also be raised or lowered without scaling, by specifying a vertical offset. Its value is used simply as a constant addition to the curve functions. An element can be turned around vertical axis going through its mid point. The turn angle is specified in the element's definition. If the element extends beyond cell boundaries as a result of the turn, it is trimmed to fit. In most cases, grid vertices comprising the cell into which the element is placed will have different heights resulting in non-horizontal base surface. In our model, elements are never tilted. Instead, the would-be tilt value is added to the element's height producing preserving its vertical orientation.

3.5. Terrain definition language

The compact nature of the model allows defining terrains in a human-readable text-based format such as JSON. Consider the following example that defines a square terrain area 100x100 units in size (the "size" attribute.) The model is unit-agnostic, but meters can be assumed for simplicity:

```
{
  "descr": "100x100 units sloped example terrain"
  "size": 100,
  "surfTypes": {
    "pebble": {
      "descr": "Regular gray pebble",
      "cohesion": 0.99, "fricAngle": 41, "sinkage": 0.22,
      "color": [ 200,200,200 ], "sonarRef1": 0.33
    }
  }
}
```

```
    },
    "grass": {
      "descr": "Green summer grass",
      "cohesion": 0.99, "fricAngle": 41, "sinkage": 0.61,
      "color": [ 000,150,000 ], "sonarRefl": 0.15
    },
    "sand": {
      "descr": "Quartz sand",
      "sinkage": 0.7, "cohesion": 0.99, "fricAngle": 41,
      "color": [ 000,150,000 ], "sonarRefl": 0.24
    },
    "brick": {
      "descr": "Red brick",
      "sinkage": 0, "cohesion": 0.98, "fricAngle": 41,
      "color": [ 080,000,000 ], "sonarRefl": 0.8
    }
  },
  "elemTypes": {
    "box": {
      "descr": "Simple rectangular box",
      "x": [1,1]
    },
    "hill": {
      "descr": "Round hill with a sine-wave surface",
      "x": 'cos', "y": []
    },
    "pyramid": {
      "descr": "Triangular pyramid",
      "x": [0,1,0], "y": []
    },
    "slope": {
      "descr": "Irregularly angled slope",
      "x": [0,0.02,0.04,0.06,0.07,0.11,0.13,0.15,0.17,0.2]
    }
  },
  "elemTree": {
    "type": "pebble",
    "surface": "sand",
    "vScale": 0.7,
    "children": [
      {
        "descr": "A box rotated 40 degrees clockwise",
        "type": "box", "surface": "brick",
        "cellX": 1, "cellY": 1, "vScale": 1, "vOffset": 0,
        "turnDeg": 40
      },
      {
        "descr": "A hill scaled down explicitly to 1/10",
        "type": "hill", "surface": "grass",
        "cellX": 2, "cellY": 1, "vScale": 0.1, "vOffset": 0
      },
      {
        "descr": "Pit constructed from hill with negative multiplier",
        "type": "hill", "surface": "sand",
        "cellX": 2, "cellY": 1, "vScale": -0.5, "vOffset": 0
      },
      {
        "descr": "A pyramid raised with vertical offset value",
        "type": "pyramid", "surface": "brick",
        "cellX": 7, "cellY": 1, "vScale": 0.3, "vOffset": 2
      }
    ]
  }
}
```

```

    }
  ]
}
}
}

```

Square shape is required in our implementation so the size can be expressed with a single value, but any rectangle can be accommodated by specifying a value for each side. "surfTypes" element specifies the dictionary of micro-terrain types. The variables "sinkage", "fricAngle", "cohesion", "color" and "sonarRefl" are for sinkage, friction angle and cohesion coefficient, visualisation colour and sonic wave reflection capability respectively. "elemTypes" element defines the dictionary of terrain primitives: box, hill, pyramid and slope.

Each primitive is formed by two cross-sections defined in two vector variables: "x" and "y". Since it is common for one cross-section to be either a horizontal line or the same as the other, it is possible to omit definition in these cases. If no vector variable is defined, it is filled with values of 1 to the length of the other vector. If a variable is defined but empty ("[]" literal in JSON), its value is copied from the other cross-section.

Now consider the "elemTree" element. It describes the placement and adjustment parameters of the elements comprising the terrain. The "type" parameter refers to one of the primitives defined under "elemTypes" above. "surface" does the same for "surfType" "vScale" and "vOffset" parameters allow to fine-tune the instance of the primitive by vertically scaling or raising it. "turnDeg" parameter contains the turning angle. "cellX" and "cellY" variables specify the grid cell in which the resulting element is placed.

3.6. Traversability and deformation modelling with micro-terrain

Surfaces that mobile robots must navigate vary in their physical qualities: hard floors, sand, rubble, grass, bog, etc. The type of locomotion that mobile robots use also varies: legged, caterpillar, wheeled or mecanum-wheeled are some of the popular choices. Since the combination of these features is critical for assessing navigability of an area of terrain, the relevant surface characteristics must be adequately represented in a realistic terrain model. For some types of robots and terrains, the shape and physical qualities of the terrain change as a result of locomotion, either temporarily or permanently, which also needs to be accounted for.

The above calls for an extension to the terrain representation. In our example model, we represent terrain deformation on the micro-terrain level, leaving damage to meso-terrain a potential subject for future work.

We use an approximate deformation model because a complete physics model will be too complex and intractable for real-time simulation (see [3] for what it would involve).

There are many way to assess terrain traversability [17] by a wheeled robot [9]. We'll consider a classification algorithm described in [4]. It is based on simplified form of classical terramechanical equations and has two input terrain parameters: cohesion c and internal friction angle ϕ . Cohesion is the force binding soil particles together; it makes the most difference for clay soils. Friction angle ϕ is measured between the normal force and resultant force that is attained when failure occurs in response to a shearing stress. Its tangent is the coefficient of sliding friction. Both of these values are specific to the type of soil and can be looked up in geological reference literature or determined experimentally in a relatively simple lab setting.

Fig.5. Forces acting on a wheel moving over deformable terrain

With c and ϕ it is possible to compute terrain shear strength and thus give an estimate of traversability, from Coulomb's equation [4]: $\tau_{\max} = (c + \sigma_{\max} \tan \phi)$ The derivation of normal stress value σ can be found in [4]. For our purposes it is enough to note the variables it is derived from: vertical load W , linear speed V , torque T , wheel angular

speed ω and sinkage z . The sinkage generally depends on the properties of the soil which cannot be derived from cohesion or friction angle. So we add another parameter to the model: the sinkage coefficient k so that $z = W \cdot k$.

This is an approximation and the derivation of k for various types of surfaces is outside the scope of this paper. For an example sinkage model that can be used see [5]. Since these parameters are specified for types of micro-terrain, terramechanical properties of terrain elements can be described in a compact and logical way. While our research focuses on wheeled robots, the same soil description parameters can support basic modelling for tracked [7] and legged [6] locomotion.

3.7. Visualisation and interoperability

Effective means of visualisation is a prerequisite for any terrain model. While rendering of terrains defined in our model is relatively easy to code from scratch, interoperability with an open 3D format is still very helpful. It is possible to export terrains from our model into DEM or VRML formats. It is also possible to import files in these formats wholly or partially to create terrain primitives.

VRML allows to use mature and feature-rich visualisation tools, interoperate with software such as MathWorks Simulink 3D and import external 3D objects that can be converted to cross-section-based primitives. The DEM (Digital Elevation Model) is a family of formats representing terrains with elevation points. These can be visualised with most GIS tools and, after conversion, used by robot simulation software such as Gazebo [13]. DEM [19] is also a popular format for publishing real terrain datasets, many of which are available free of charge [16].

4. Conclusions and future work

In this paper we described a terrain model with a number of unique characteristics. Future work will focus on increasing flexibility and completeness of this model. Allowing completely free positioning of child elements in parent's cells without seriously degrading performance of point height queries is one challenge. This may call for multiple sub-models with different balance between performance and realism. Another direction worth exploring is improving micro-terrain model by introducing permanent deformability and layered soils. We are also looking into ways of supporting overhanging objects and caves without the complexity of full three dimensions (building on the approach in [15]), and a terrain synthesis algorithm for building artificial terrain based on real patches (an idea explored in [12]).

5. References

- [1] Li Z., Zhu Q., Gold C. Digital terrain modeling: principles and methodology. New York: CRC Press, 2005. ISBN: 978- 0-415-32462-5.
- [2] Kidner D., Dorey M., Smith D. What's the point? Interpolation and extrapolation with a regular grid DEM // Fourth International Conference on GeoComputation, Fredericksburg, VA, USA. 1999. URL: http://www.geocomputation.org/1999/082/gc_0_82.htm (: 2015/ 09/ 02).
- [3] Wong, Jo Yung. Terramechanics and off-road vehicles. Elsevier, 1989.
- [4] Iagnemma, Karl, Hassan Shibly, and Steven Dubowsky. "On-line terrain parameter estimation for planetary rovers." In Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on, vol. 3, pp. 3142-3147. IEEE, 2002.
- [5] Meirion-Griffith, Gareth, and Matthew Spenko. "Comprehensive pressure-sinkage model for small-wheeled unmanned ground vehicles on dilative, deformable terrain." In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 4052-4057. IEEE, 2012.
- [6] Manko, David J. A general model of legged locomotion on natural terrain, Vol. 179. Springer Science & Business Media, 2012.
- [7] Le, Anh Tuan, D.C. Rye, and H.F. Durrant-Whyte, Estimation of track-soil interactions for autonomous tracked vehicles, in: 1997 IEEE International Conference on Robotics and Automation, 1997. Proceedings, 2:1388–93 vol.2, 1997. doi:10.1109/ROBOT.1997.614331.
- [8] Pyanichnikov V.E. Algorithmic base for remote sensors of mobile robots, Mechatronics, Automation, Control. 2008. № 10(91). P. 10-21.
- [9] Filaretov V., Pryanichnikov V., Autonomous mobile university robots AMUR: Technology and applications to extreme robotics, Procedia Engineering. 2015. Vol.100. P. 269-277.
- [10] Pryanichnikov V., Andreev V., Kirsanov K., et al., Technologies of modernization and sensing of special mobile robots, Annals of Southern Federal University (Techn. sc.): Taganrog. 2011. № 3. P. 166–171.
- [11] Katalinic B., Pryanichnikov V., Ueda K., Cesarec P., Kettler R., Bionic Assembly System: hybrid control structure, working scenario and scheduling, in: Proceedings of 9th National Congress on Theoretical and Applied Mechanics. Brussels, 2012. P. 101–108.
- [12] Zhou, H., Jie Sun, G. Turk, and J.M. Rehg, Terrain synthesis from digital elevation models, IEEE Transactions on Visualization and Computer Graphics 13, no. 4 (July 2007): 834–48. doi:10.1109/TVCG.2007.1027.

- [13] Koenig, Nathan, and Andrew Howard, Design and use Paradigms for Gazebo, an open-source multi-robot simulator,in: Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, 3:2149–54. IEEE, 2004. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1389727.
- [14] Smelik, Ruben M., Tim Tutenel, Klaas Jan de Kraker, and Rafael Bidarra, Declarative terrain modelling for military training games, International Journal of Computer Games Technology 2010 (2010): 1–11. doi:10.1155/2010/360458.
- [15] Triebel, Rudolph, Patrick Pfaff, and Wolfram Burgard,Multi-level surface maps for outdoor terrain mapping and loop closing, in: Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, 2276–82. IEEE, 2006. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4058725.
- [16] Tong, Chi Hay, David Gingras, Kevin Larose, Tim Barfoot, and Érick Dupuis. The Canadian Planetary Emulation Terrain 3D Mapping Dataset. The International Journal of Robotics Research, 2013, 0278364913478897.
- [17] Papadakis, Panagiotis. Terrain Traversability Analysis Methods for Unmanned Ground Vehicles: A Survey. Engineering Applications of Artificial Intelligence 26, no. 4 (2013): 1373–85.
- [18] Génévaux, Jean-David, Éric Galin, Eric Guérin, Adrien Peytavie, and Bedřich Beneš. Terrain Generation Using Procedural Models Based on Hydrology. ACM Transactions on Graphics (TOG) 32, no. 4 (2013): 143.