Quickstart Guide

VoxelSpace is a volumetric spatial platform that transforms massive terrain, infrastructure, and sensor datasetsinto living 4D environments. Traditional models capture only surfaces—lines, elevations, or textures. Volumetricmodeling uses
voxels-3D pixels that store data about each cubic volume. This unlocks new capabilities forsimulation, change detection, infrastructure planning, and geoscience because the entire volume becomes thedataset.

VoxelSpace Quickstart Guide

VoxelSpace is a volumetric spatial platform that transforms massive terrain, infrastructure, and sensor datasetsinto living 4D environments. Traditional models capture only surfaces—lines, elevations, or textures. Volumetricmodeling uses
voxels-3D pixels that store data about each cubic volume. This unlocks new capabilities forsimulation, change detection, infrastructure planning, and geoscience because the entire volume becomes thedataset.
VoxelSpace centralizes data in a cloud-hosted repository and offers real-time voxel-level modeling and massive-scale processing using serverless spatial functions called
spatial lambdas.

Prerequisites

Before getting started, ensure you have:

- A VoxelSpace account (the free tier allows up to 5 GB of uploaded data and includes limited compute hours).
- Data in supported file formats as outlined in the upload wizards, including common point-cloud, mesh, block-model, and raster formats.

1. Sign Up & Set Up Your First Project

Follow these steps to create your first project:

1. Navigate to platform.voxelspace.com and log in with your Microsoft, Google identity or your work email.
2. On the catalog page, click **New Project**
3. Provide a project name and description
4. Select your time zone
5. Accept the default coordinate datums and projections:

   - WGS 84 horizontal datum
   - Mean-sea-level vertical datum

6. Set a voxel resolution or leave the default setting
7. Submit the form

Your new project will appear in the project list.

2. Prepare and Upload Your Data

Inside your project, click **Add Object ? Raw Data** and choose the type of dataset you want to import:

Point Cloud

- Upload LiDAR or photogrammetry point clouds
- **Supported formats:** .las, .laz, .txt, .xyz, or compressed .zip files
- Provide an object name and optional description
- Select your file

Mesh

- Upload surface meshes (e.g., .obj files)
- When prompted for a column meta label, supply "Default" or another appropriate label
- Choose your file

Block Model

- Upload block-model tables
- **Supported formats:** .csv, .txt, .xls, .xlsx, or .zip
- Map your columns to Origin X, Origin Y, Origin Z, and Value fields
- Select horizontal and vertical datums and a projection

Ortho-Imagery

- Upload orthorectified images or mosaics
- **Supported formats:** .png, .jpg, .bmp, or .tif
- For PNG/JPEG/BMP files, include the corresponding world file (.jgw, .pgw, .tfw, etc.)
- GeoTIFFs embed this metadata and can be uploaded directly

After choosing the file and completing required fields, click **Create**. A new row appears in the project list showingthe object type, creation time, size, and status icon. Note that raw objects cannot be visualized until they areprocessed.

3. Process Data into Indexed Datasets

To make data queryable and visualizable, convert raw objects into indexed datasets or voxel grids:

Indexed Points

Converts a raw point cloud into a queryable point-cloud dataset. Select your source point cloud and click **Create**.

Indexed Mesh

Similar to Indexed Points but for meshes. Select the raw mesh as source and create an indexed mesh.

Indexed Imagery

Processes ortho-imagery into an indexed raster dataset. **Important:** Only one processing job can run at a time onthe free tier, so subsequent jobs remain queued until the current task finishes.

Voxel Block Model

Converts block-model tables into voxel grids. You can specify:

- Translation offsets
- Scale factors
- Rotation parameters

Launch this process only when no other job is queued.

Additional Processing Options

- **Voxel Terrain**
- **Ortho Voxel Terrain**
- **Voxelised Points**
- **Voxelised Mesh**

Each option is tailored to specific data types. Processing tasks appear on the **Pending Tasks** page. In the free tier,only one task can run concurrently, so plan your workflow accordingly.

4. Visualize Your Data

Once you have indexed datasets, explore them in the interactive 3D viewer:

Creating a View

1. Select **New View** from the project or choose **View** from an indexed object
2. Enter a name and description for your view

Working with the View Editor

1. Click **Add Objects** and select your indexed datasets
2. Datasets appear in the left panel with icons for:Visibility (colored dot)
    - Color settings (palette icon)
    - Bounding-box display
    - Zoom controls
    - Visibility (colored dot)

Navigation and Controls

- **Visibility toggle:** Use the colored dot to show/hide datasets
- **Camera controls:** Click the cross-hair icon to center the camera on the dataset
- **Extent display:** Use the box icon to reveal dataset boundaries
- **Manual navigation:** Zoom or pan manually if objects don't appear automatically

Customizing Appearance

- **Color settings:** Access via the palette icon to choose constant colors or gradient schemes
- **Point size adjustment:** Set point sizes for point clouds
- **Color palettes:** Includes options like Viridis and Plasma
- **Raster overlays:** Overlay raster layers on imagery

Toolbar Features

- Switch between navigation tools (select, pan, zoom)
- Show or hide the grid
- Enter full-screen mode
- Automatic camera re-centering when using zoom icons

5. Explore Advanced Features

VoxelSpace offers powerful capabilities beyond basic visualization:

Spatial Calculations

Run serverless **spatial lambdas** to:

1. Compute volumes across datasets
2. Filter data based on specific criteria
3. Derive metrics across trillions of voxels

Temporal Tracking

Add time as a dimension to your datasets to observe changes over time, including:

- Erosion monitoring
- Construction progress tracking
- Fluid movement analysis

Multi-Source Data Fusion

Seamlessly integrate multiple data types into a single voxel grid:

- LiDAR point clouds
- Drill hole data
- Seismic datasets
- IoT sensor data
- CAD files

This unified approach enables comprehensive analysis across diverse data types.

Collaboration and Integration

- **Project sharing:** Collaborate with team members
- **Unity export:** Export projects for game engines
- **API integration:** Integrate the platform into existing workflows

Conclusion & Next Steps

This quickstart guide has walked you through the core VoxelSpace workflow:

- **Project creation** and initial setup
- **Data upload** for various formats
- **Processing** raw data into indexed datasets
- **Visualization** in the interactive 3D viewer

Volumetric modeling with voxels captures the full volume of space, enabling insights that traditional surfacemodels cannot provide.

Recommended Next Steps

- Explore additional processing types for your specific data
- Experiment with spatial lambdas for advanced analysis
- Consider upgrading your plan to access:
  - Increased storage capacity
  - Additional compute hours
  - Concurrent job processing

For more advanced tutorials and documentation, visit the VoxelSpace knowledge base or contact support throughthe platform.

Spatial Lambdas

VoxelSpace's **spatial lambdas** are serverless functions that execute custom computations on your volumetricdata.

Introduction

VoxelSpace's **spatial lambdas** are serverless functions that execute custom computations on your volumetricdata. They run near your data on highly parallel infrastructure, allowing you to process trillions of voxels inminutes. This guide explains what spatial lambdas are, when to use them, how to write and execute them, and bestpractices for efficient processing.

What are Spatial Lambdas?

Spatial lambdas are stateless cloud functions written in languages such as **Python** or **.NET**. They take voxel gridsas input, perform a computation, and return a result or a modified dataset. Examples include:

- **Volume calculations** and statistical summaries
- **Filtering voxels** by property values
- **Creating derived datasets** from existing data
- **Classifying voxels** with machine learning models

Because the lambdas execute server-side, they scale to trillions of voxels through massive parallelization.

When to Use a Lambda

Use a spatial lambda when you need to:

- Compute metrics such as t**otal volume tonnage average density**, or other summarystatistics
- **Filter** voxels based on property thresholds (e.g., density > 2.5) or remove empty voxels
- **Resample** data to a coarser or finer resolution
- **Classify** voxels with a machine learning model (e.g., ore vs. waste, rock type, or vegetationclass)
- Generate **reports** of computed metrics for compliance or engineering studies

Authoring a Lambda

Spatial lambdas are typically composed of three main components:

1. Input Definition

Specify the dataset(s) and region of interest (bounding box or selection) to process.

2. Compute Function

Write code that iterates through voxels and applies your logic. For example, summing the ore_grade property forvoxels above a grade threshold.

3. Output

Return a summary value (e.g., total_volume) or create a new dataset (e.g., filtered voxels). Lambdas can alsoattach files or emit logs during execution.

Sample Python Lambda

Here's a basic example that sums the volume of high-grade voxels:

```
def lambda_handler (context, voxel_reader, voxel_writer):
  total_volume = 0.0
  for voxel in voxel_reader:
    if voxel.properties.get('grade',0.0) &gt;= 0.5:
      total_volume += voxel.size
  return {'total_volume': total_volume}
```

Code Explanation

voxel_reader**:** Provides voxel objects with property dictionaries and geometry

voxel.properties.get('grade', 0.0)**:** Safely retrieves the grade property with a default valueof 0.0

voxel.size**:** Represents the volume of a voxel (e.g., 1 m^3)

Return value: Dictionary with the computed total volume that will appear in reports

Running a Lambda

Follow these steps to execute a spatial lambda:

Step 1:**Prepare Your Dataset**

Ensure your data is processed into an **indexed dataset** or **voxel grid**. Lambdas operate on indexed datasets ratherthan raw uploads.

Step 2:**Select or Upload a Lambda**

Navigate to your dataset's actions menu

Choose **Run Lambda**

Select a pre-built lambda (e.g., volume calculation, surface extraction) OR upload your owncode file

– – –

Step 3:Configure Parameters

Define the execution parameters:

Region of interest: Full dataset or specific bounding box

Property filters: Set thresholds (e.g., grade ? 0.5)

Output options: Specify whether to generate new datasets

User inputs: Provide any custom parameters your lambda requires

Step 4:Launch the Job

Start the lambda execution:

The task appears in **Pending Tasks**

On the free tier, only one job (processing or lambda) can run at a time

Additional jobs queue until the current job finishes

Step 5:Review Results

When complete, results appear in:

Reports section: Summary values and statistics

Outputs section: New datasets generated by the lambda

Best Practices

Performance Optimization

Limit the region: Restrict the lambda to the smallest area necessary to reduce compute costand time

Efficient algorithms: Use vectorized operations where possible for better performance

Memory management: Be mindful of memory usage when processing large datasets

Development Workflow

Test on a sample: Validate your code on a small subset before running it on the full dataset

Incremental development: Start with simple logic and gradually add complexity

Error handling: Include proper error handling for robust execution

Resource Management

Reuse templates: Start with VoxelSpace's built-in lambdas for common tasks; modify themrather than writing from scratch

Queue awareness: Plan your processing and lambda jobs to avoid long queues, especially onthe free tier where concurrency is limited

Monitor usage: Track your compute usage to stay within plan limits

Debugging and Validation

Use logs: Include log statements in your lambda to help diagnose issues and verifyintermediate results

Validate inputs: Check that your data has the expected properties before processing

Test edge cases: Consider what happens with empty regions, missing properties, or extremevalues

Advanced Lambda Example

Here's a more sophisticated lambda that performs statistical analysis:

```
def statistical_analysis_lambda(context, voxel_reader, voxel_writer):
  grades = []
  volumes = []
  for voxel in voxel_reader:
    grade = voxel.properties.get('grade',0.0)
    if grade &gt; 0: # Only include non-zero grades
      grades.append(grade)
      volumes.append(voxel.size)
  if not grades:
    return {'error':'No valid grade values found'}

  # Calculate statistics
  total_volume = sum(volumes)
  weighted_average_grade = sum(g * v for g, v in zip(grades, volumes)) / total_volume

  return {
    'total_volume': total_volume,
    'voxel_count':len(grades),
    'average_grade':sum(grades) / len(grades),
    'weighted_average_grade': weighted_average_grade,
    'min_grade': min(grades),
    'max_grade': max(grades)
  }
```

Troubleshooting Common Issues

Performance Problems

- **Large memory usage:** Process voxels in batches rather than loading all into memory
- **Slow execution:** Optimize algorithms and reduce unnecessary computations
- **Timeout errors:** Break large jobs into smaller regions

Data Issues

- **Missing properties:** Always use the `.get()` method with default values
- **Unexpected data types:** Validate and convert data types as needed
- **Empty regions:** Check for zero voxel counts before performing calculations

Runtime Errors

- **Import errors:** Ensure all required libraries are available in the runtime environment
- **Syntax errors:** Test your code locally before uploading
- **Logic errors:** Use logging extensively to trace execution flow

Conclusion

Spatial lambdas bring high-performance analytics directly to your data, enabling custom computations at massivescale. By understanding how to author, configure, and run these serverless functions, you can:

- **Extract deeper insights** from voxel models
- **Automate complex spatial analyses**
- **Process trillions of voxels** efficiently
- **Create custom workflows** tailored to your specific needs

The power of spatial lambdas lies in their ability to execute near your data, eliminating the need to transfermassive datasets and enabling real-time analysis at unprecedented scales.

Next Steps

To advance your spatial lambda expertise:

1. **Explore built-in templates** available in the VoxelSpace platform
2. **Experiment with different processing techniques** and algorithms
3. **Study advanced examples** in the developer documentation
4. **Consider upgrading your plan** for increased compute resources and concurrent processing
5. **Join the VoxelSpace community** to share techniques and learn from other users

For comprehensive developer documentation and advanced tutorials, consult the full manuals within VoxelSpace once you have platform access.