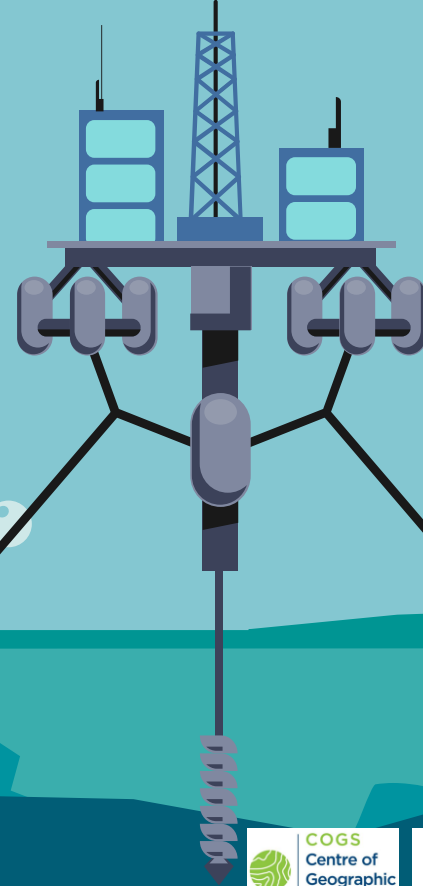


Automating Offshore Pipeline Routing

Tiana Gallo
Alex Moss
Mira Rayson



OVERVIEW

01

BACKGROUND

02

THE DATA

03

OBJECTIVES

04

METHODOLOGY

05

RESULTS

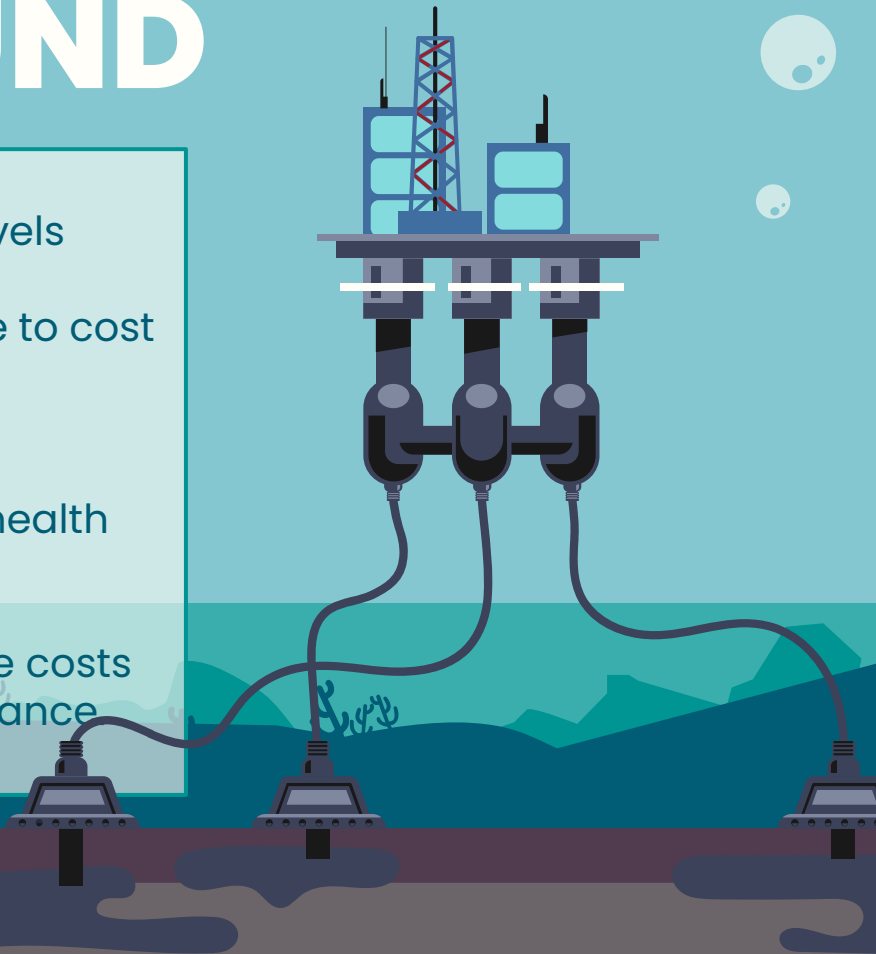
06

DISCUSSION

01

BACKGROUND

- A least-cost path is a planned route that travels from a destination to a source point and is guaranteed to be the cheapest route relative to cost units.
- LCPs can be used in various fields such as transportation, urban planning, emergency health services and in energy management.
- Useful for subsea pipeline routing to minimize costs while considering impact/ regulatory compliance



01

BACKGROUND

- Study area for this project is in the Gulf of Mexico (GoM).
- Covers over 600 000 km² of seafloor.
- This region's oil and gas industry is one of the most developed in the world.
- All data required for the project was provided to us by Geosyntec.

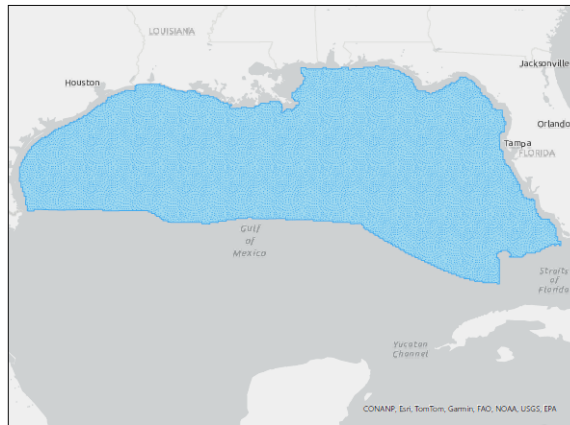


Figure X. Total BOEM Protraction Area



02

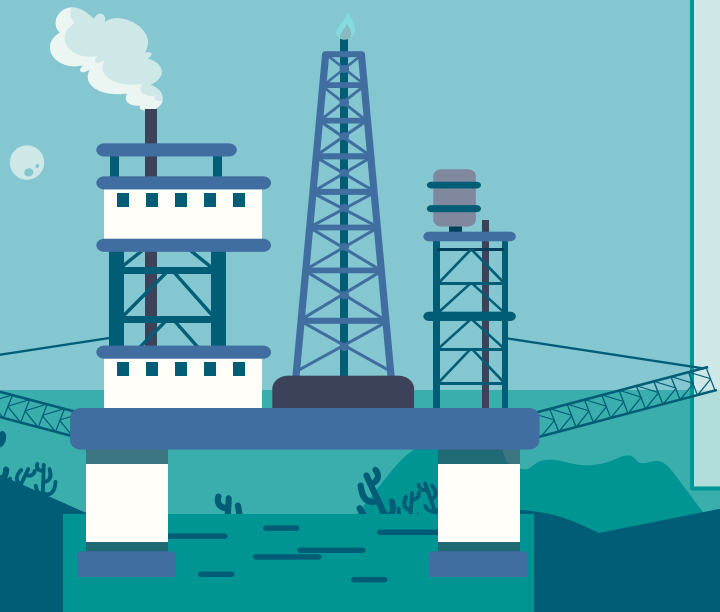
THE DATA

- Geosyntec provided us with crucial data to undertake this project with. The data can be broken down into three main categories:
 1. Bathymetry of the Gulf of Mexico
 2. Seafloor anomalies
 3. Existing subsea pipelines and their associated infrastructure

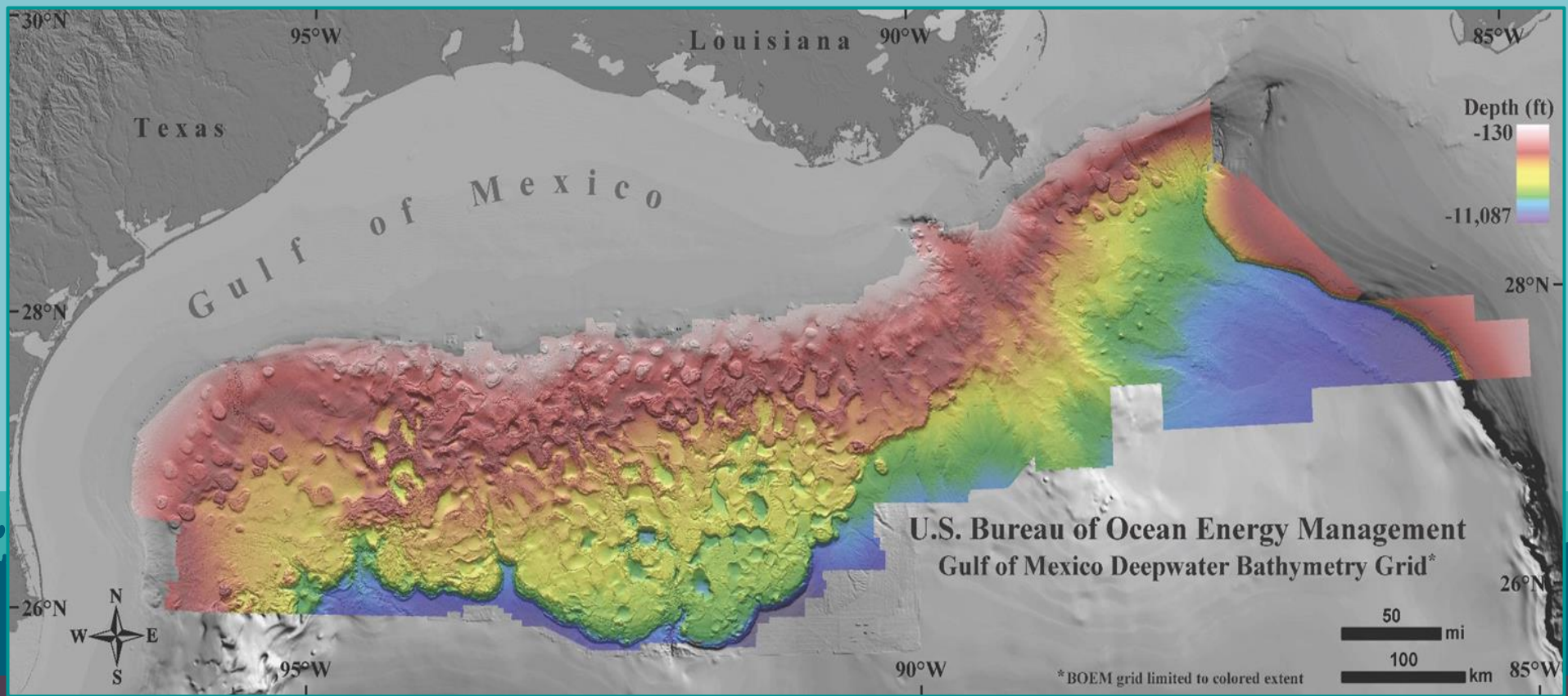


02

THE DATA

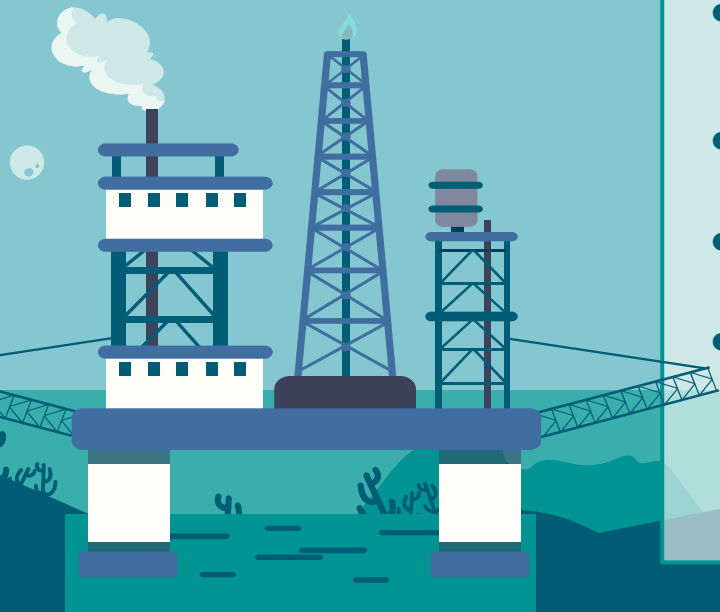


- The bathymetry grid was created by stitching together survey data that covers more than 90,000 square miles.
- The grid is comprised of 1.4 billion 40 by 40 feet cells that range from -130 feet to -11,087 feet (-40 to -3,379m).
- Seafloor slope plays a huge role when setting up infrastructure and planning pipeline routes.

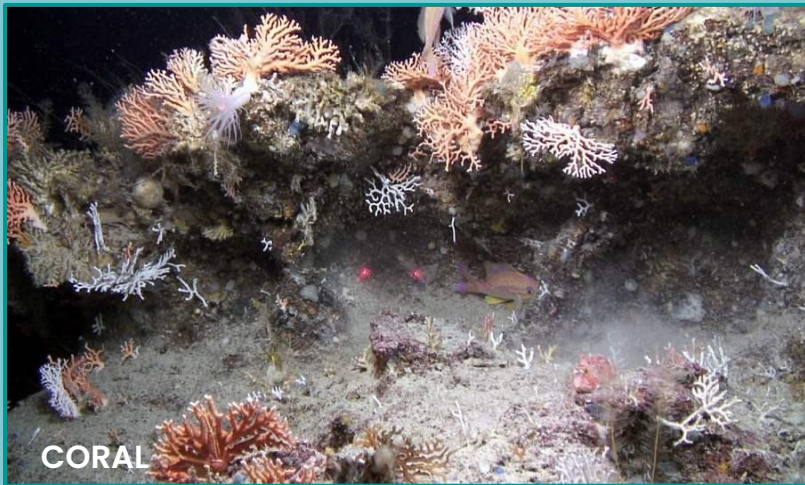


02

THE DATA



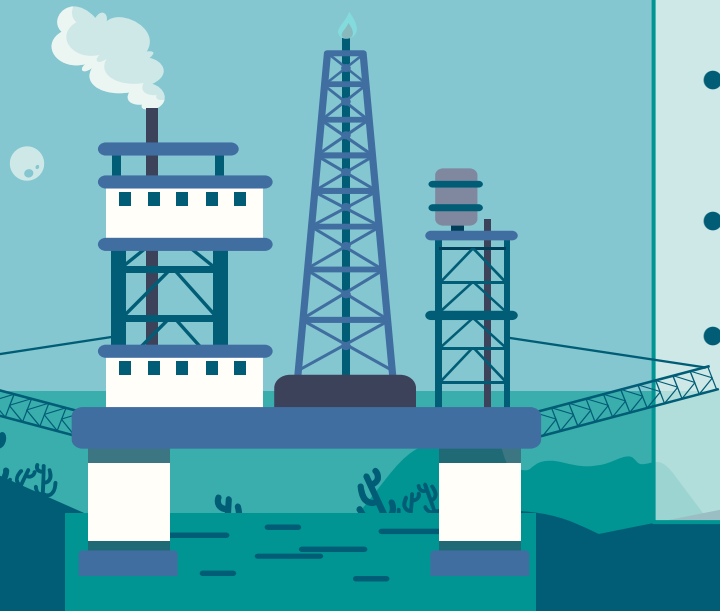
- The seafloor anomalies data consist of natural hydrocarbon seeps and related benthic fauna.
- Avoiding seafloor anomalies is crucial in the offshore oil and gas industry.
- Anomalies can compromise the stability and safety of subsea infrastructure.
- Environmental protection is essential, as seafloor anomalies often host unique and sensitive ecosystems.



02

THE DATA

- Gulf of Mexico's offshore drilling accounts for 14.5% of U.S crude oil production (2022).
- The region is densely populated with existing subsea pipelines and associated infrastructure.
- This extensive infrastructure can make it challenging to route new pipelines.





SUBSEA PIPELINES



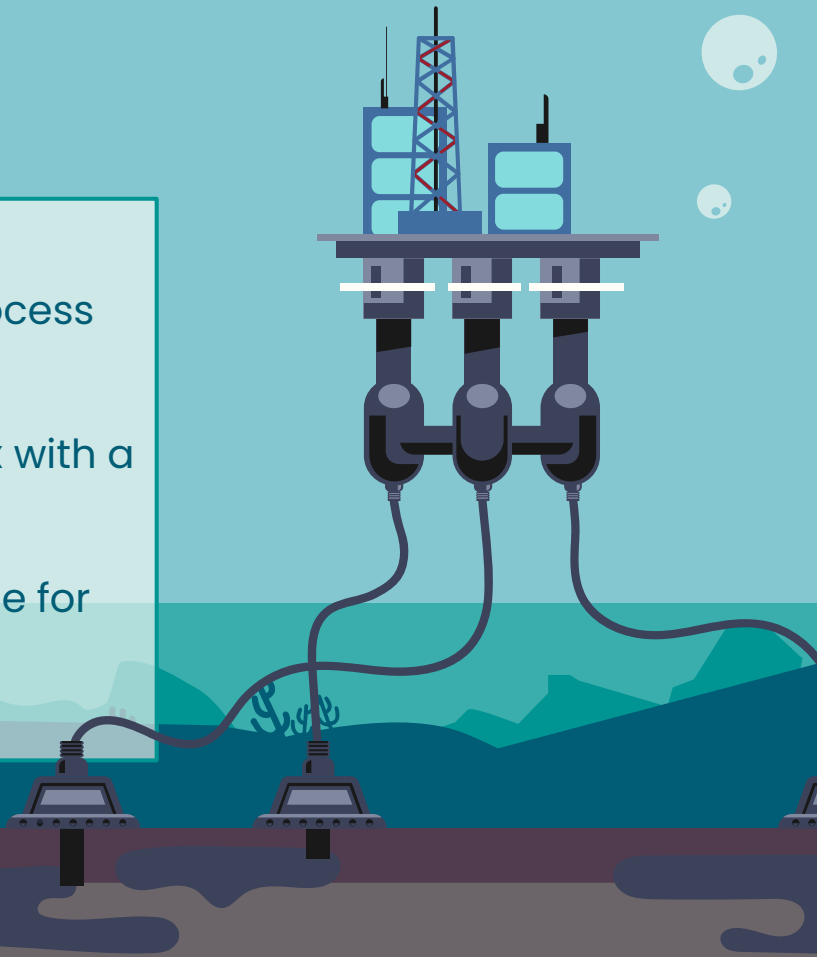
PLATFORMS



PLATFORMS

03 OBJECTIVES

- Develop a GIS-based tool to automate the process of subsea pipeline routing.
- Implement the tools into an ArcGIS Pro toolbox with a Python extension.
- Ensure the tools are re-runnable and shareable for any potential future use.



04

METHODOLOGY



MODEL BUILDER

Create least-cost path
model using various
geoprocessing tools within
ArcGIS Pro



PYTHON TOOLBOX

Automate pipeline routing
process allowing users to
input own parameters

PARAMETERS

- Gradient avoidance degree: 30°
- Anomaly avoidance distance: 500 ft
- Well infrastructure avoidance distance: 500 ft
- Platform infrastructure avoidance distance: 500 m
- Pipeline avoidance distance: 200 ft, if a pipeline must be crossed, it should do so perpendicularly
- Start Point (X / Y): 1,948,443.499 ft / 9,496,072.400 ft
- End Point (X / Y): 2,535,851.500 ft / 10,250,934.201 ft

MODEL BUILDER

- Create slope raster from bathymetry data – reclassify
- Combine all rasters layers to avoid and include buffers
- Raster Calculator

Input raster
Bath_Slope

Reclass field
VALUE

Reclassification

Reverse New Values

Start	End	New
0	6	1
6	12	2
12	18	3
18	24	4
24	30	5
30	90	100
NODATA	NODATA	NODATA

Input raster
Features2Avoid_Raster

Reclass field
VALUE

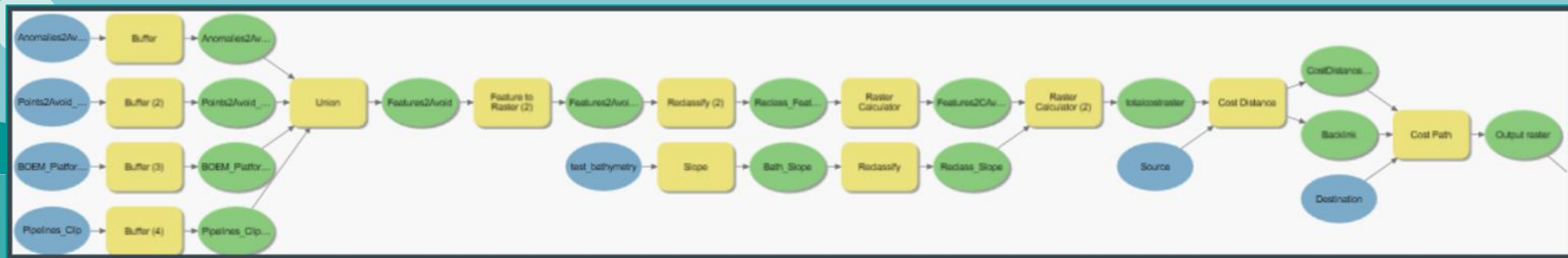
Reclassification

Reverse New Values

Start	End	New
-1	1708	100
1708	3463	100
3463	5018	100
5018	6543	100
6543	8473	100
NODATA	NODATA	NODATA

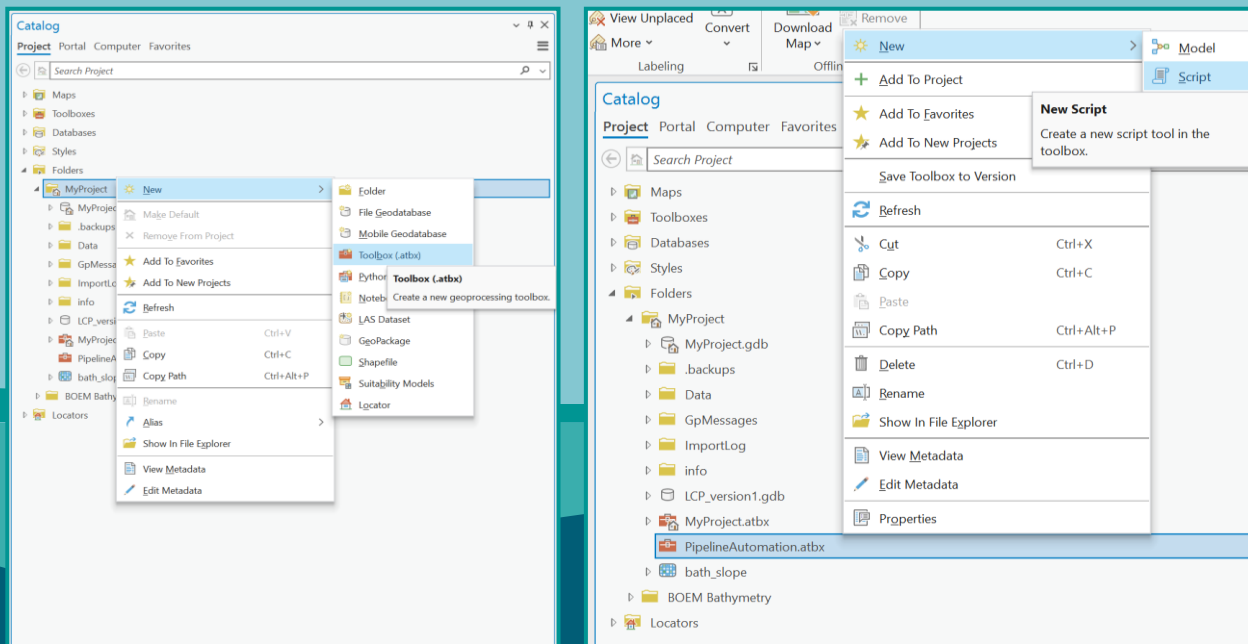
MODEL BUILDER

- Cost Distance
- Cost Path
- Convert to Polyline



PYTHON TOOLBOX

Within Catalog Pane in ArcGIS Pro, add a new toolbox with associated Python script



PYTHON TOOLBOX

1. Slope Raster Reclassification

- Input Raster
- Output Raster
- Reclassification Values

General	Define the script tool parameters					
Parameters						
Execution						
Validation						
		Label	Name	Data Type	Type	Direction
	0	Input Raster	Input_Raster	Raster Layer	Required	Input
	1	Output Slope Raster	Output_Slope_Raster	Raster Dataset	Required	Output
	2	Reclassification Values	Reclassification_Values	ValueTable	Required	Input
	*			String	Required	Input

Geoprocessing

1. Slope Reclassification

Parameters Environments

* Input Bathymetry

* Reclassified Slope

* Reclassification Values

Reverse New Values

Start

End

New

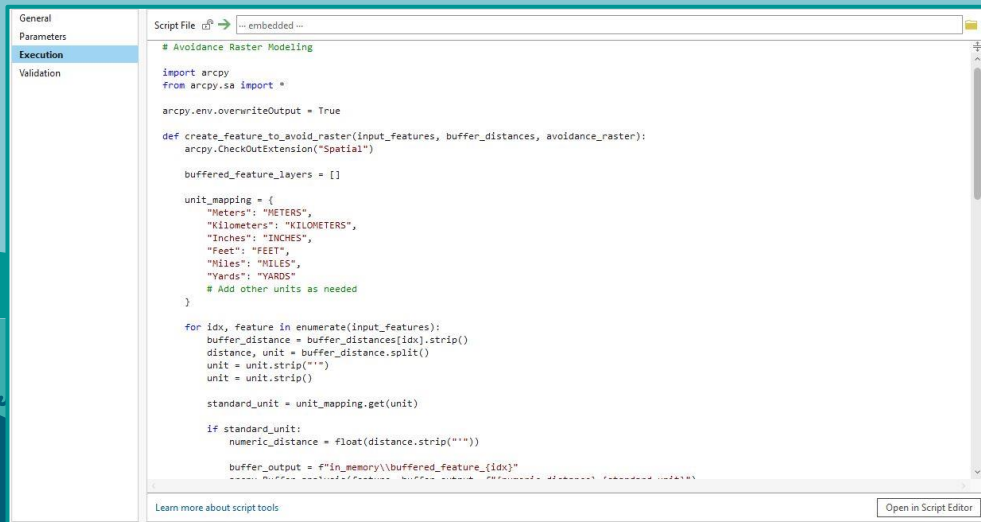
Classify

Unique

PYTHON TOOLBOX

2. Avoidance Raster Reclassification

- Input Raster(s)
- Buffer Distances
- Output Raster



The screenshot shows the 'Python Script Editor' window with the 'Execution' tab selected. The script is titled '# Avoidance Raster Modeling'. It imports 'arcpy' and 'arcpy.sa'. It sets 'arcpy.env.overwriteOutput = True'. It defines a function 'create_feature_to_avoid_raster(input_features, buffer_distances, avoidance_raster)' which checks the extension and creates a list of buffered feature layers. It includes a unit mapping dictionary and a loop to process each feature, converting distances to a standard unit and creating a buffer. The script ends with a line to create the final output raster from the buffered features.

```
# Avoidance Raster Modeling

import arcpy
from arcpy.sa import *

arcpy.env.overwriteOutput = True

def create_feature_to_avoid_raster(input_features, buffer_distances, avoidance_raster):
    arcpy.CheckOutExtension("Spatial")

    buffered_feature_layers = []

    unit_mapping = {
        "Meters": "METERS",
        "Kilometers": "KILOMETERS",
        "Inches": "INCHES",
        "Feet": "FEET",
        "Miles": "MILES",
        "Yards": "YARDS"
    }
    # Add other units as needed

    for idx, feature in enumerate(input_features):
        buffer_distance = buffer_distances[idx].strip()
        distance, unit = buffer_distance.split()
        unit = unit.strip()
        unit = unit.strip()

        standard_unit = unit_mapping.get(unit)

        if standard_unit:
            numeric_distance = float(distance.strip())

            buffer_output = f"in_memory\\buffered_feature_{idx}"
            # Create buffer output feature. Buffer output. Remove dataset. Remove index.
```

Learn more about script tools [Open in Script Editor](#)

Geoprocessing

2. Features to Avoid

Parameters

Environments

* Input Features to Avoid

* Buffer Distance

 Unknown

* Avoidance Raster

PYTHON TOOLBOX

3. Least Cost Path

- Input Raster(s)
- Weights
- Output Cost Raster
- Source Point
- Destination Point
- Least Cost Path
- Smoothing Tolerance

General	Define the script tool parameters					
Parameters	Label	Name	Data Type	Type	Direction	
Execution	0 Input Raster Layers	Input_Raster_Layers	[Raster Layer]	Required	Input	
Validation	1 Weights	Weights	[String]	Optional	Input	
	2 Output Cost Raster	Output_Cost_Raster	Raster Dataset	Optional	Output	
	3 Source Point	Source_Point	Feature Set	Required	Input	
	4 Destination Point	Destination_Point	Feature Set	Required	Input	
	5 Least Cost Path	Least_Cost_Path	Feature Class	Required	Output	
	6 Smoothing Tolerance	Smoothing_Tolerance	Linear Unit	Required	Input	
	*		String	Required	Input	

Geoprocessing

3. Least Cost Path

Parameters Environments

* Input Raster Layers

Weights

Output Cost Raster

* Source Point

* Destination Point

* Least Cost Path

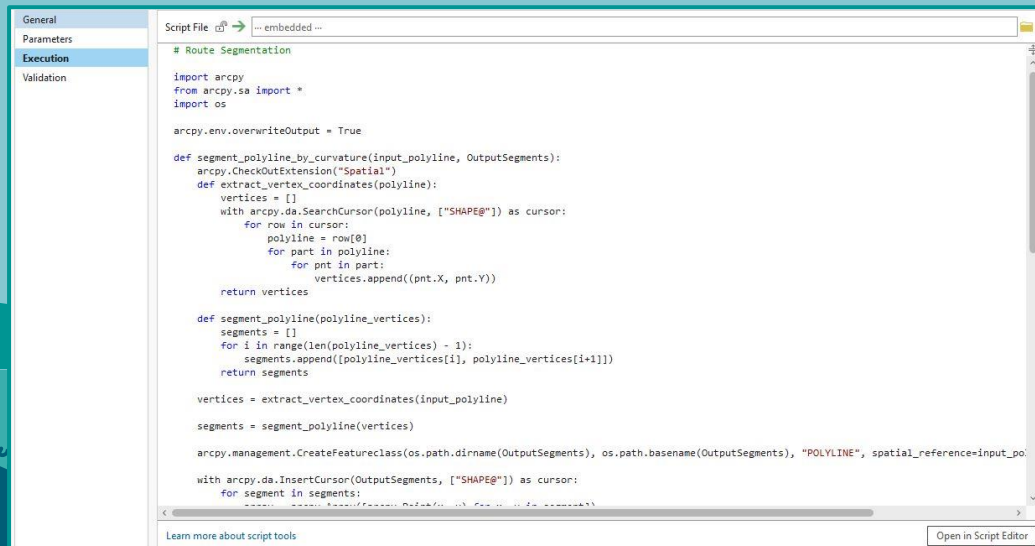
* Smoothing Tolerance

Unknown

PYTHON TOOLBOX

4. Route Segmentation

- Input Route
- Output Route Segments



The screenshot shows the Python Script Editor window in QGIS. The 'Script File' tab is active, displaying a Python script for route segmentation. The script imports arcpy, sets environment variables, and defines functions to extract vertex coordinates and segment a polyline based on curvature. The script is saved as 'embedded.py'.

```
Script File  embedded.py

# Route Segmentation

import arcpy
from arcpy.sa import *
import os

arcpy.env.overwriteOutput = True

def segment_polyline_by_curvature(input_polyline, OutputSegments):
    arcpy.CheckOutExtension("Spatial")
    def extract_vertex_coordinates(polyline):
        vertices = []
        with arcpy.da.SearchCursor(polyline, ["SHAPE@"]) as cursor:
            for row in cursor:
                polyline = row[0]
                for part in polyline:
                    for pnt in part:
                        vertices.append((pnt.X, pnt.Y))
        return vertices

    def segment_polyline(polyline_vertices):
        segments = []
        for i in range(len(polyline_vertices) - 1):
            segments.append([polyline_vertices[i], polyline_vertices[i+1]])
        return segments

    vertices = extract_vertex_coordinates(input_polyline)
    segments = segment_polyline(vertices)

    arcpy.management.CreateFeatureclass(os.path.dirname(OutputSegments), os.path.basename(OutputSegments), "POLYLINE", spatial_reference=Input_po

    with arcpy.da.InsertCursor(OutputSegments, ["SHAPE@"]) as cursor:
        for segment in segments:
            cursor.insertRow([segment])

Learn more about script tools
```

Geoprocessing

4. Route Segmentation

Parameters Environments

* Input Polyline



Output Segments

RouteSegmentation



PYTHON TOOLBOX

4. Route Segmentation – Symbology

- Input Route
- Output Route Segments

Symbology - RouteSegmentation

Primary symbology

Graduated Colors

Field: Shape_Length

Normalization: <None>

Method: Manual Interval

Classes: 2

Color scheme:

Classes Histogram Scales

Symbol	Upper value	Label
	≤ 10560	0.000000 - 10560.000000
	≤ 130800	10560.000001 - 130800.0...

PYTHON TOOLBOX

5. Report Creation

- Input Route
- Output Report

General		Label	Name	Data Type	Type	Direction
Parameters	0	Input Polyline	Input_Polyline	Feature Set	Required	Input
Execution	1	Report Creation	Report_Creation	File	Required	Output
Validation	*			String	Required	Input

Geoprocessing

←

5. Generate Report

+

Parameters

Environments

?

* Input Polyline

📁

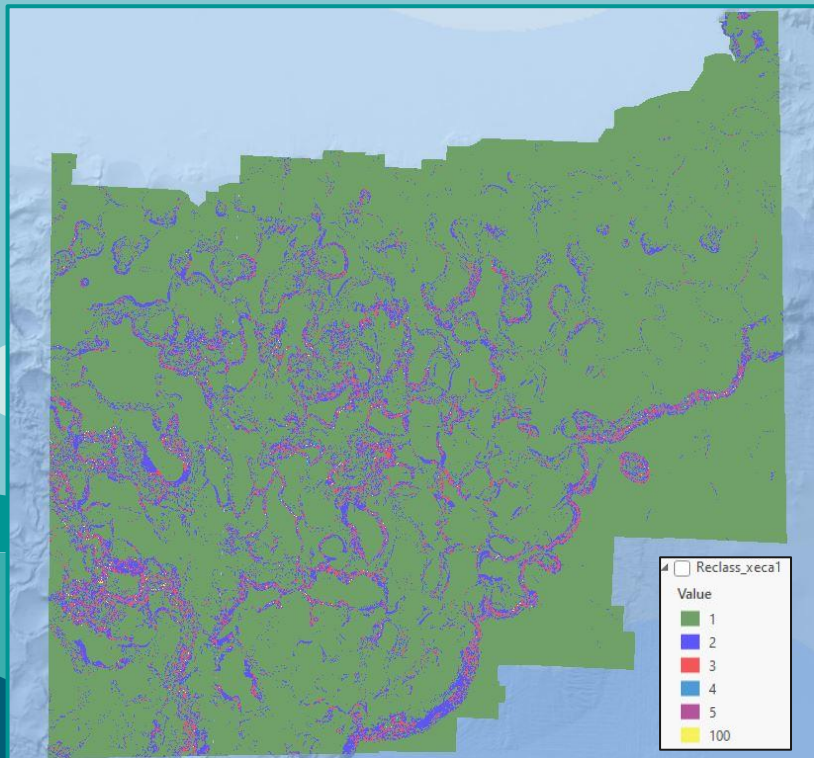
✎

▼

* Report Creation

📁

05 RESULTS

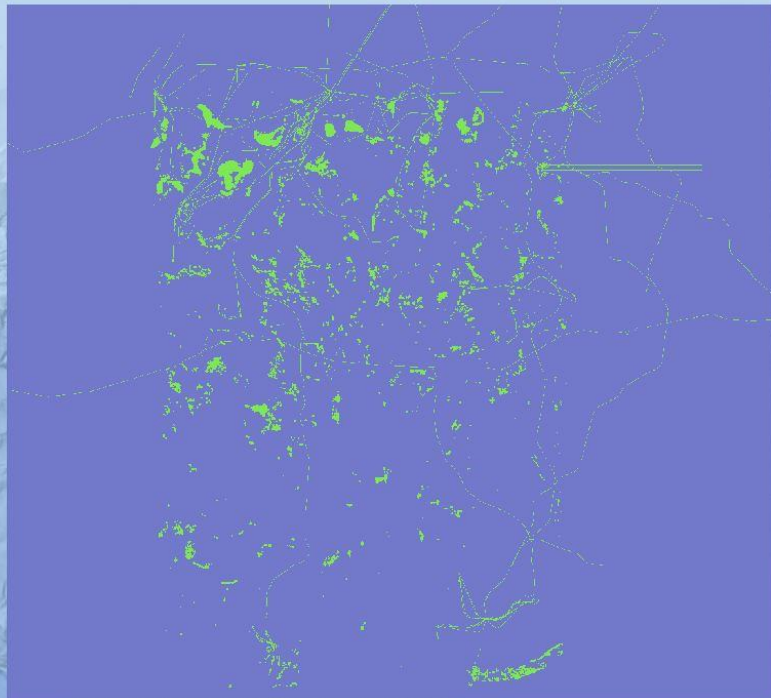


Slope Raster Reclassification

- This raster shows the classifications assigned to the bathymetric raster, with lower values indicating gentler slopes.
- Ensures slopes greater than thirty degrees are classified as impassable (given a value of 100).

05

RESULTS



Avoidance Raster Reclassification

- The reclassified raster showing the features we want to avoid plus their buffers.
- The bright green indicates the features which have a weight of 100 and are impassable (with the exception of crossing pipelines perpendicularly).
- Purple represents surrounding areas and has a weight of 0.

05

RESULTS

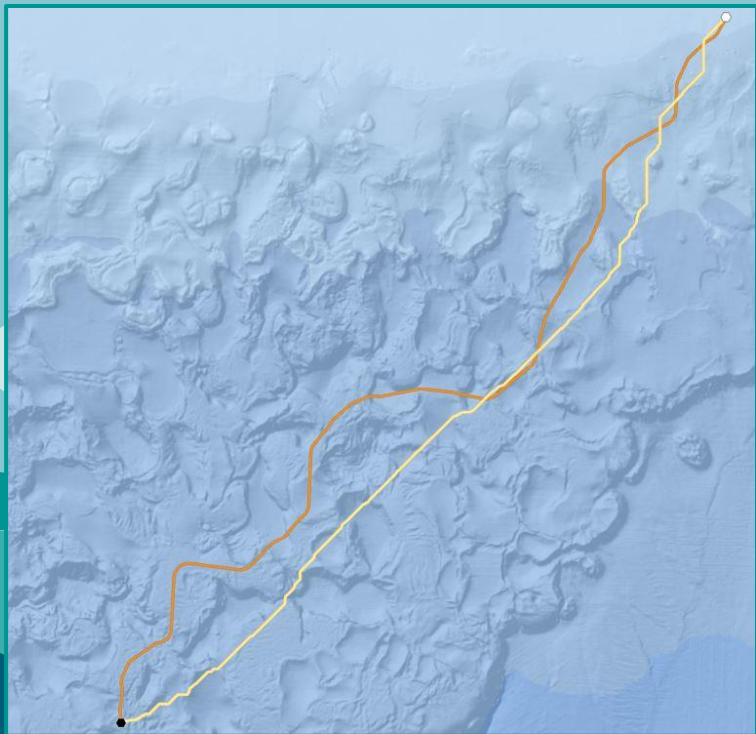


Cost Surface

- The Cost Surface shows the slope and Avoidance rasters combined with appropriate weights assigned to each.
- Black areas indicate a cost of zero to traverse, and white areas indicate that areas are impassable (minus pipeline exception).

05

RESULTS

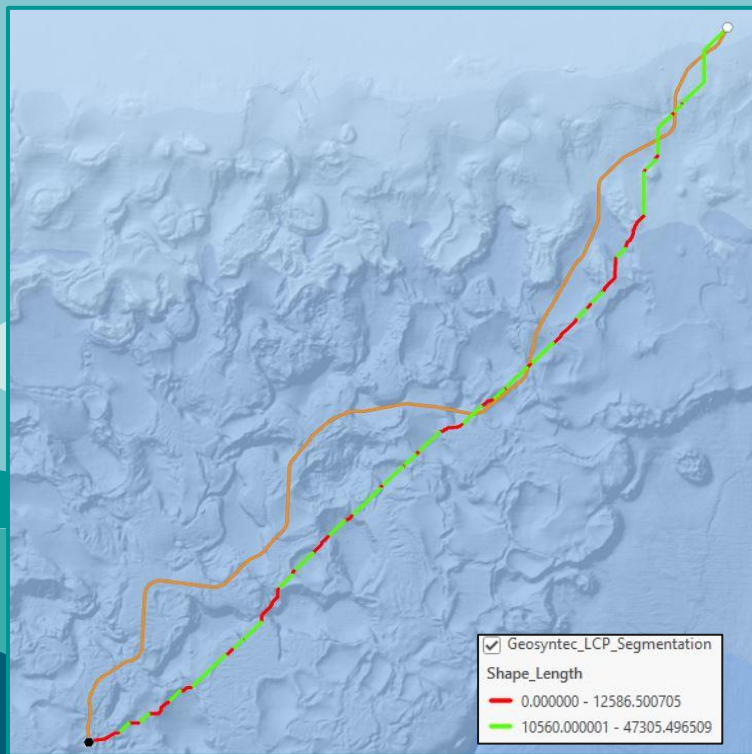


Least Cost Path

- Image displays the output from the Least Cost Path tool.
- Yellow path is our calculated route.
- Orange path is the existing pipeline.
- Shows how our toolbox would route the coordinates of a pre-existing pipeline, for comparison purposes.
- Various reasons why routes are different including costing implementation, data quality, and others.

05

RESULTS



Route Segmentation

- Red + Green line is the LCP after running Segmentation tool.
- Segments that have a straight distance of < 2 miles are **red**.
- Segments that have a straight distance of > 2 miles are **green**.

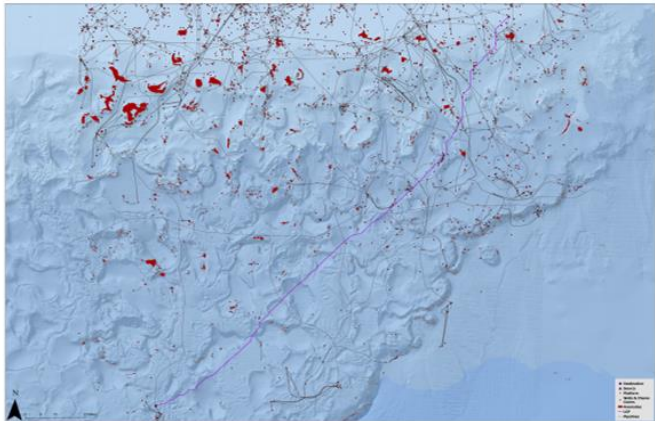
05

RESULTS

Least Cost Path Report

Attribute	Value
Length of Polyline	195.09 miles
Start Coordinates	(603530.0, 10254468.748356499)
End Coordinates	(-20169.999999996275, 9527318.748356499)
Coordinate Reference System	NAD_1927_Transverse_Mercator

Map Layout

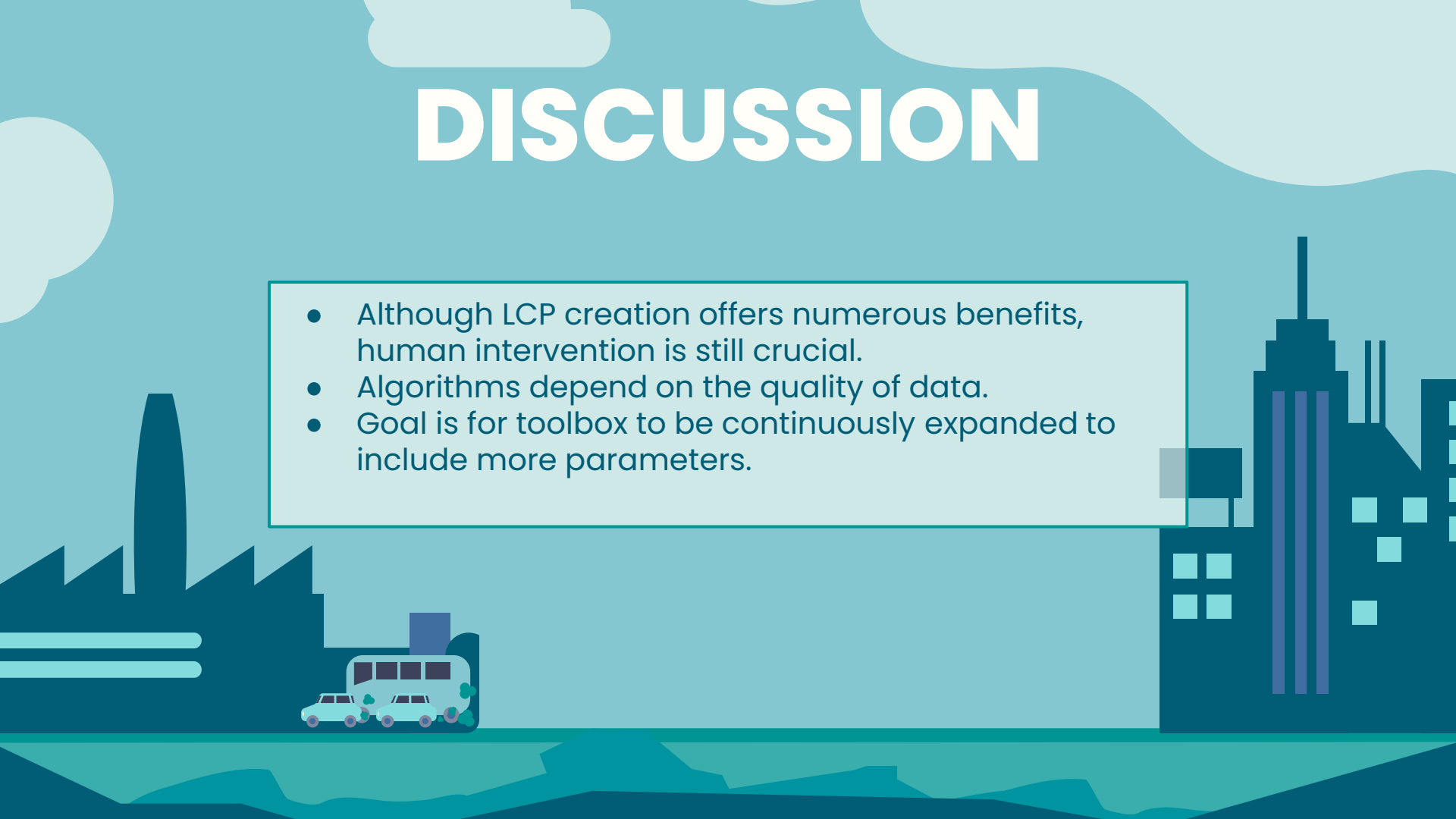


Report Creation

- The pdf output of the reporting tool includes key details about the LCP.
- This tool is optional, and we have created a map frame template to use if desired.

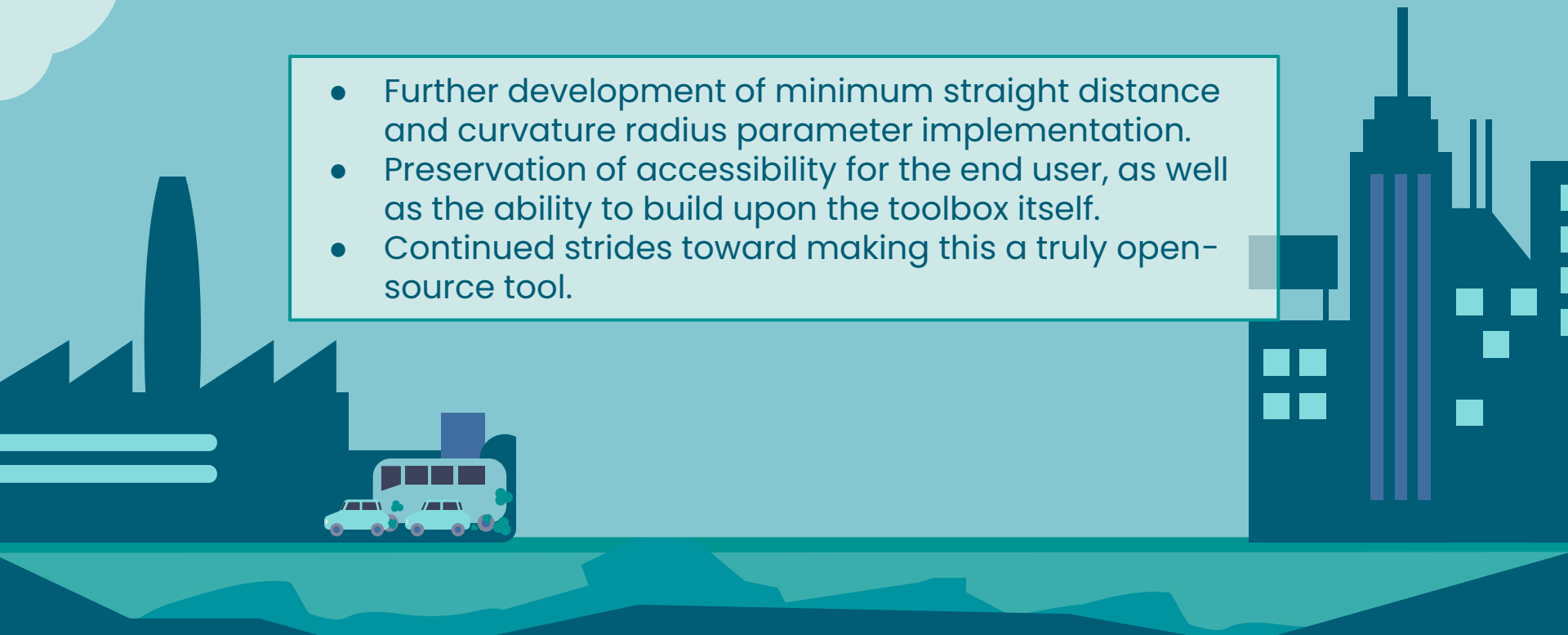
DISCUSSION

- Although LCP creation offers numerous benefits, human intervention is still crucial.
- Algorithms depend on the quality of data.
- Goal is for toolbox to be continuously expanded to include more parameters.



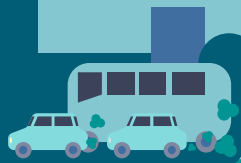
ADVANCEMENTS

- Further development of minimum straight distance and curvature radius parameter implementation.
- Preservation of accessibility for the end user, as well as the ability to build upon the toolbox itself.
- Continued strides toward making this a truly open-source tool.

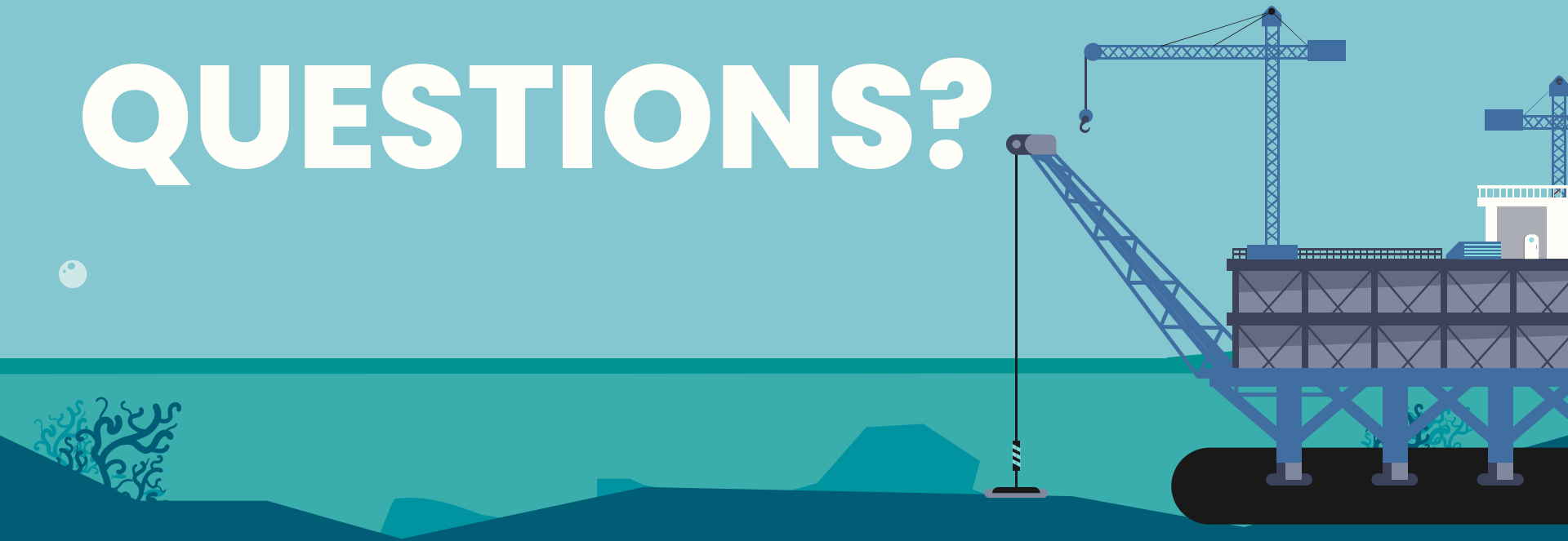


CONCLUSION

- By combining the data obtained from Geosyntec with our own research, we were successful in automating the pipeline routing process.
- Model Builder provided a solid foundation to further our understanding in creating a Python Toolbox.
- The Toolbox can be shared with other users and is very user-friendly.



QUESTIONS?



THANK YOU

