Ice-Floe Simulation Viewer Tool

Justin Adams
Computer Engineering
Memorial University
St. John's, Newfoundland
jadams@mun.ca

Justin Sheppard
Computer Engineering
Memorial University
St. John's, Newfoundland
justin.sheppard@mun.ca

Shadi Alawneh
Electrical & Computer Engineering
Memorial University
St. John's, Newfoundland
shadi.alawneh@mun.ca

Dennis Peters
Electrical & Computer Engineering
Memorial University
St. John's, Newfoundland
dpeters@mun.ca

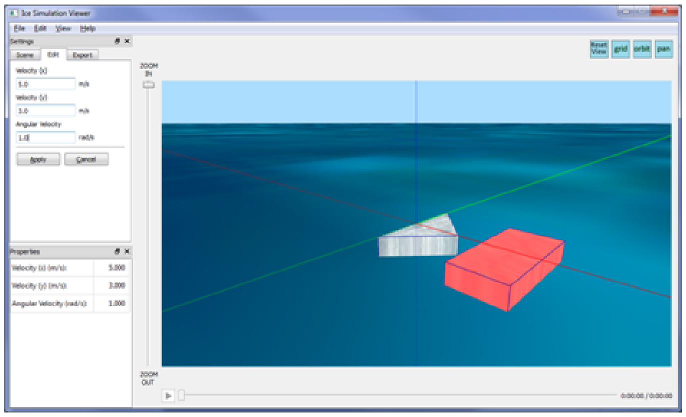
***Abstract*—We are developing software to simulate the behaviour of sea ice in a floe on the ocean surface, including interactions with solid objects (e.g., ships). In order to interact with this simulation and to visualize the results, we are developing a tool that gives the user the ability to set up the initial conditions, run the simulation, and display the simulated ice-floe data in sequence. Initial data can be extracted from images (e.g., from a satellite), manually manipulated, and then passed to the simulator. The output can then be viewed and interrogated interactively. By using this tool to display the simulation of the ice-floes, the users will have the ability to analyze ice-floe and ice-structure interactions.**

I. INTRODUCTION

Ice interactions is an area of research that is gaining popularity amongst researchers and investors alike. The Sustainable Technology for Polar Ships and Structures (STePS^2) project is developing a numerical model for sophisticated simulations of ice-structure and ice-ice interactions that is being implemented using High Performance Computing techniques — specifically, the STePS^2 model is utilizing a high-end graphics processing unit for the computational part of an ice floe simulation.

To compliment the numerical model, a graphical user interface (GUI) is being developed to allow the user to visualize the simulation results. This tool provides users with a 3D interactive environment for analyzing simulations provided by the model. Users can use zoom, pan, and rotate to move around the field and simulations can be played in slow motion, real-time, or accelerated time, or on a frame-by-frame basis. The functionality provided by the interface has a look and feel similar to popular 3D modelling software such as Google Sketchup so that new users feel comfortable when utilizing the tool. Figure 1 shows a screenshot of the main interface of the Ice Simulation Viewer with example objects loaded.

Furthermore, there are several methods for importing the ice fields, including from a proprietary text format, COLLADA files, and images. Once the ice field has been imported the user can edit the properties of individual floes, including X, Y, and angular velocities. Users also have the ability to export initial conditions of scenes for use with the model, or, for a simulation, data can be exported as a comma separated value (.csv) file to, for example, plot a floe's velocity against time. However, the functionality that is available to the user depends on the type of data they are currently viewing — a static scene, or an actual simulation.



Velocity (x) 5.0 m/s
Velocity (y) 3.0 m/s
Angular Velocity 1.0 rad/s

Velocity (x) (m/s):	5.000
Velocity (y) (m/s):	3.000
Angular Velocity (rad/s):	1.000

Fig. 1. Ice Simulation Viewer

II. STATIC VS. DYNAMIC SCENES/SIMULATIONS

The Ice Simulation Viewer Tool uses two types of files referred to as *static scenes* and *simulations*. Static scenes are files that contain just the position and attribute data for a number of ice floes, whereas simulations further contain a series of updated positions for the floes that were generated by the numerical model and can be displayed as an actual simulation of the floes in a given time domain. The user must choose which they wish to load before any functionality in the GUI will become available, with the exception of the image import tool, discussed in Section VIII. Scenes and simulations have to be treated differently because some data is only available to be displayed or manipulated in certain states - for example, a scene contains no data with which to display a simulation so this functionality is not available. Similarly, since the numerical model and the GUI are currently separate, a simulation cannot have its properties edited on-the-fly.

A scene can be created manually in a text file, in a 3D graphics program that supports exporting as COLLADA files, or by the image import tool. Within the scene view the user can click on an object and view or edit its properties: currently X-, Y-, and angular velocities, with others to be added as the simulation tool is enhanced. A scene is manually passed into the numerical model to be used as an initial condition state that can be used to generate a simulation. Any frame within a simulation can be extracted, edited and used as a scene. A simulation is therefore a series of static frames (scenes) played sequentially. Each frame contains information about the current objects to be displayed. A simulation is passed in three parameters to be initialized:

- Time Step: Integer value which indicates the amount of time, in seconds, between each frame of the simulation;
- Multiplier: Integer multiplier for the Time Steps parameter. The value of the Multiplier times the Time Steps value is the number of frames per second in the simulation;
- Length: This is the total length of the simulation in seconds.

Once a simulation as been loaded and initialized in the Ice Simulation Viewer the data is read by the viewer and displayed using OpenGL in the main viewing area. Once a simulation is loaded into the Ice Simulation Viewer the user can play, pause and stop the simulation as desired. Any time the simulation is paused the user can export the current frame as a scene and then edit its properties to then create a new simulation.

III. TOOLS USED

The Ice Simulation Viewer is written in C++ and utilizes several different libraries to achieve its various features.

A. Nokia Qt

The Nokia Qt library[1] is used for the creation of the graphical user interface in the Ice Simulation Viewer. The Qt library has wide range of features to allow designers to create graphical user interfaces quickly and efficiently. The library has numerous built in features including multimedia networking, scripting, database, XML, and multi-threading support. The library also includes a 2D graphics canvas which allows developer to use a wide array of 2D image manipulation tools and has integrated OpenGL support to allow 3D graphics programming with OpenGL syntax in line with the Qt application.

The Ice Simulation Viewer makes extensive use of the Qt library for all of its graphical user interface components. It utilizes the multi-threading support to ensure constant responsiveness and the integrated OpenGL for graphics.

B. OpenGL

OpenGL[2] is a 2D and 3D graphics Application Programming Interface (API) that can be used with a wide range of programming languages across a number of operating platforms. OpenGL offers a stable, reliable, fast and well supported graphics medium that abstracts the intricacies of video hardware from the developer, allowing a common graphics syntax across all OpenGL supported devices.

The Ice Simulation Viewer uses OpenGL for all its 3D graphics components. The 3D viewing area uses OpenGL for all functionality, including drawing the 3D objects (ice floes), displaying a pseudo-ocean (an ocean that currently has no collision detection with the ice-floes), applying textures to objects, and the pan, zoom and rotate operations.

[1]http://qt.nokia.com

[2]http://www.opengl.org

C. OpenCV

Analyzing and modifying images in the Ice Simulation Viewer for use with importing scenes from satellite imagery is accomplished using the Open Computer Vision (OpenCV) library[3]. Written in C, OpenCV provides many highly optimized algorithms for real-time computer vision and image processing. To extract ice floes from images, the Ice Simulation Viewer only utilizes a few basic functions from the OpenCV library, including thresholding (converting colour images to black and white) and some morphological operators. Much more advanced functionality is available in OpenCV, but it is currently not required for extracting ice-floes from images.

IV. IMPORT METHODS

A. Text

Text files are the main method of input for the Ice Simulation Viewer; the data in the files are configured with a custom formatting that is read on a frame by frame basis by the Ice Simulation Viewer. Each frame of the simulation is saved as an individual file that contains the data associated with each object in that current frame. Each object has a set of data that includes 3D positional information for each vertex of the object to be used in drawing as well as some attributes including the X-, Y-, and angular velocities of the object.

B. COLLADA

Collaborative Design Activity (COLLADA)[4] is an interchange format used for interactive 3D applications and defines an open standard XML schema for describing 3D objects and animations. Once selected for importing into the Ice Simulation Viewer, the COLLADA file's XML structure is parsed and its meshes are extracted and converted to planes, which are then converted to the Ice Simulation Viewer's custom object types for representing floes.

C. Images

Creating lengthy lists of point coordinates of 3D objects in text files and manually creating realistic-looking floes in a 3D modelling program can be time consuming, especially when large numbers of floes are involved (e.g., see Figure 2). A tool for the Ice Simulation Viewer was created using OpenCV to extract these floes from a selected image that can then be used as scenes such that the user can then assign properties to the objects and use the scene as initial conditions for a simulation; the details of the import process from image to a scene are discussed in Section V. A wide range of common file types are supported, including JPEG, PNG, GIF, TIFF, and BMP. For example, the user is able use a satellite image of ice-floes from any location and use the Ice Simulation Viewer to create a scene with the extracted data from the image. The user can then edit the properties of the scene and use it as an initial condition state for a simulation, allowing the user to create a realistic scenario for a real location and various conditions.

[3]http://opencv.willowgarage.com

[4]http://collada.org

Fig. 2. Ice Floe[1]

V. IMPORTING FROM IMAGES

Importing static ice fields into the Ice Simulation Viewer from satellite imagery allows for easy representation of real ice fields without having to manually create them using 3D modelling software or manually typing coordinates. Upon importing an image into the Ice Simulation Viewer, images are automatically adjusted and processed using various morphological operations[2] to isolate and extract the shapes in the image as polygons that represent ice floes. To accomplish this, the importer utilizes OpenCVs image processing functionality to perform some basic image operations to segment the background and foreground areas.

A. Segmentation Methodology and Algorithms

Segmenting the image foreground from its background is accomplished using three image processing techniques: thresholding, the morphological erosion and dilation operators, and contour tracking. An ongoing example in this section will utilize the Figure 3 as the original image:

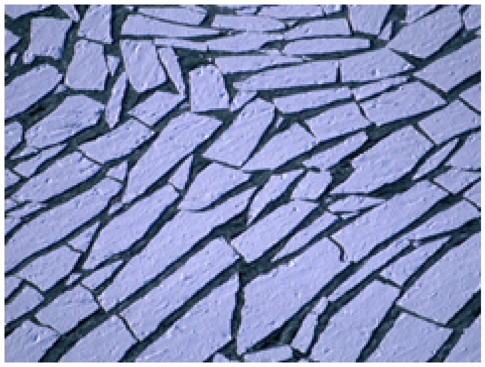


Fig. 3. Ice Floes Image Processing Example[3]

1) Thresholding: Thresholding[4] is the main function used to remove background information. Many of the image processing techniques that follow this step cannot be performed on a coloured or even grayscale image - a binary (black and white) image must be used. Thus, the first action performed on the image is a binary threshold. OpenCVs implementation of threshold provides the use of Otsus method; it assumes two classes in the image (background and foreground) and calculates the optimum threshold separating the two so that their combined intra-class variance is minimal. If the user is not satisfied with the result, a slider is provided to adjust the level at which the image is thresholded and the results are displayed to them in real-time. Figure 4 below shows the ice floes as foreground (white) with the background water removed:

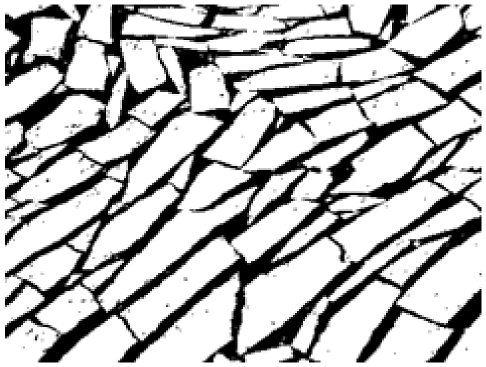


Fig. 4. Ice Floes Image Processing - Threshold

2) Morphology: Mathematical morphology is a technique for analyzing and processing geometrical structures, based on various mathematical theories. Binary morphological techniques use a structural element which passes over each pixel and, depending on the operation, uses various set theory operations (union, intersection, etc.) to calculate the value of the pixel of interest. For the image importer, the morphological erosion and dilation techniques are used to reduce noise caused by thresholding and to restore information to the actual ice floes that may have occurred due to the noise reduction[2].

a) Erosion: Binary erosion is a process which "shrinks" segmented foreground objects/contours in an image. This is achieved by passing a specified structural element over the image and performing an intersection with the surrounding pixels. If the current pixel's neighbours match the structural element, it is set to 1, but otherwise 0. The example below shows a 5×5 matrix A eroded by a 3×3 structural element B:

$$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} \ominus \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix} \rightarrow \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} \tag{1}$$

The center 1 in the 3×3 structural element is used as the center. In the second 5×5 matrix, only the center 1 survived, as they were the only pixels whose neighbours perfectly matched the structural element. With erosion (and many other operations), a number of iterations can be specified, which would pass the structural element over the image again. If a second iteration was performed in the above example, all pixels would be set to 0.

Figure 5 shows how erosion works in the ice simulation viewer on the on-going example, using three iterations.

Fig. 5. Ice Floes Image Processing - Erosion

In the figure, erosion has segregated most of the barely connected floes, which is desired to ensure the convexity conversion (explained later) and does not create excessively large objects; however, there has been significant information removed from the individual floes — this can be restored using the dilate operator, explained below.

b) Dilation: Binary dilation is similar to erosion, except that it "expands" segmented areas by performing an union operation with the structural element as it passes over the image. The example below shows a 5×5 matrix A eroded by a 3×3 structural element B:

$$\begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix} \oplus \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix} \rightarrow \begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{vmatrix} \quad (2)$$

The shape becomes fully expanded as each 0 element has at least one neighbour that is a 1. As with erosion, dilation can be performed a number of times, repeatedly expanding the shape.

Figure 6 restores some of the information to the ice floes that was removed by the eroison algorithm.

3) Contour Tracking: A contour is the outline of a shape in an image. To find the contours of the ice floes in the imported image, the *findContours* function provided by OpenCV is used[5]. This method uses a border following algorithm which looks for black-to-white (background to object) transitions in the image. It then traces the object's boundary by comparing each pixel's eight neighbours to determine if it, too, is part of the object. Conveniently, *findContours* allows just the first level of contours to be returned, allowing for easy removal of any holes generated by the erosion process.

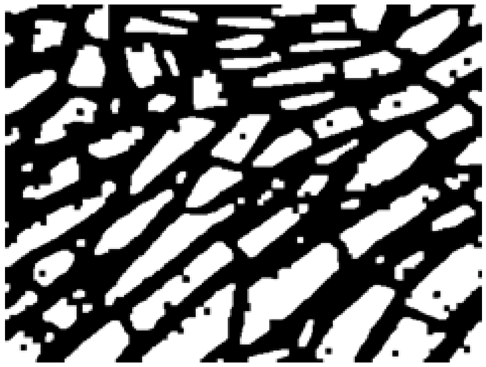


Fig. 6. Ice Floes Image Processing - Dilate

Once contours around the floes have been determined, a convex hull (a polygon with all internal angles less than $180\,^{\circ}$) is calculated. The results are shown to the user, at which point they can adjust the minimum area they wish to import, which removes small floes from the image that could have been created by the dilation. Furthermore, the user has the ability to adjust the origin in the image as well as the scale of the floes, which will affect how the floes are converted and shown in the 3D view.

B. Results

The first implementation of the image importer provides the basic functionality needed for extracting ice floes from images. Along with automatic and manual adjustments, the ability to adjust the field's origin and scale before importing allows users to achieve accurate and desired results. The figure below illustrates a fully processed image with discovered contours and adjusted origin which is ready to be imported as 3D ice floe objects:

Figure 7 shows the imported ice from the contours discovered in the example, with a centered origin in the 3D display of the Ice Simulation Viewer.

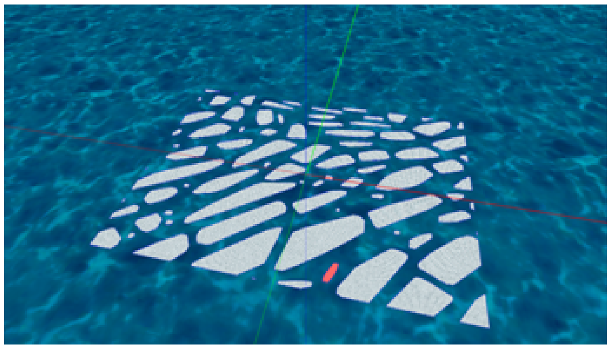


Fig. 7. Ice Floes Image Processing - Contours

VI. Exporting Data

Exporting data from the Ice Simulation Viewer can be done at two different times: during the editing of initial conditions and after a simulation has been initialized. Each of these situations are treated differently by the Ice Simulation viewer.

A. Exporting Initial Conditions

After a user has completed customizing an ice field the Ice Simulation Viewer allows the data to be exported in the form of a text file. This file can then be used to as the initial conditions of a simulation. The exported data is stored in the text file, each ice-floe represented in the data has its associated x, y, and z coordinates and a list of its initial properties.

B. Exporting Simulation Data

Once a simulation has been initialized the user can choose to export a CSV file containing the data from the simulation over a desired time period. The user can choose to export any combination of x, y and angular velocities for an individual ice-floe or the entire set of ice-floes. The export into a CSV file allows the user to easily use their data in any software with graphing capabilities that supports the CSV file type.

VII. Future Direction and Goals

Since the Ice Simulation Viewer is still in the early development stages there are a number of features that haven't been implemented and tools which will be further developed.

- Image Importer
  The Image Importer will be further developed to allow the user to have more control over how the Ice Simulation Viewer is processing the images. The user will be able to choose between several different erosion and dilation kernels which will alter the results of the operations. The user will able to preview the results and choose a kernel best suited for data they desire.
- Simulation Files
  As development progresses on the Ice Simulation Viewer a new method of input files for simulations will be implemented. The current method of one text file per frame will be replaced by a single file structured using XML. This file will contain all the data for the entire simulation which will be more efficient as the Ice Simulation Viewer will be reading from a single file.
- Exporting Data
  The option for exporting an initial condition state as a COLLADA file will also be available to the user as the development of the Ice Simulation Viewer progresses. This will allow the user to load the data into any supporting graphics software and further define the size and shape of each floe. This is advantageous, for example, when importing images causes some definition of floes to be lost - the user would be able to export to COLLADA and then refine the lines and shapes, or insert new ones, in their preferred 3D graphics program.

VIII. Conclusion

The Ice Simulation Viewer GUI provides users with an excellent means of visualizing and interacting with results produced by the ice collision numerical model. The user can create realistic scenes and use them as an inital state for simulations. These scences and simulations are displayed to the user with an integrated window with the main user interface using the OpenGL API. The Ice Simulation Viewer supports several input formats including text, COLLADA and images. The Image Importer Tool uses the OpenCV library to allow images of real world ice floes to be processed and imported into the Ice Simulation Viewer, allowing users to create simulations for real world locations. Data can then be exported from the Ice Simulation Viewer as a CSV file to be used in any supporting graphing software. The Ice Simulation Viewer is a step foward in the simulation of ice-floe movement and their interaction with polar ships and structures.

Acknowledgment

This research has been done as part of the STePS^2 project, under the leadership of Drs. Claude Daley and Bruce Colbourne, and was supported by: ABS, Atlantic Canada Opportunities Agency, BMT Fleet Technology, Husky Oil Operations Ltd, Research and Development Council, Newfoundland and Labrador and Samsung Heavy Industries.

References

[1] Haxon, "Ice floe at Oslofjord," http://www.panoramio.com/photo/19618780, March 2009.

[2] Edward R. Dougherty and Roberto A. Lotufo, *Hands-on Morphological Image Processing*, SPIE - The International Society for Optical Engineering, 2003.

[3] Jim Wark, "Ice floes on lake superior near Duluth, Duluth, Minnesota, usa," http://www.art.com/products/p14101032-sa-i2797699/jim-wark-ice-floes-on-lake-superior-near-duluth-duluth-minnesota-usa.htm.

[4] E. R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*, Morgan Kaufmann, third edition, 2005.

[5] Gary Bradski and Adrian Kaehler, *Learning OpenCV*, O'Reilly Media, 2008.