15th International Conference on

Computer and IT Applications in the Maritime Industries

COMPIT'16

Lecce, 9-11 May 2016

Edited by Volker Bertram

TUHH
Technische Universität Hamburg-Harburg
Institut für Entwerfen von Schiffen und Schiffssicherheit
Schwarzenbergstraße 95 Gebäude C D-21073 Hamburg

TuTech
TUHH Technologie GmbH

Sponsored by

DNV·GL

www.dnvgl.com

BETA CAE Systems SA

www.beta-cae.gr

www.ndar.com

NUMECA INTERNATIONAL

www.numeca.be

AVEVA CONTINUAL PROGRESSION

www.aveva.com

CPU 24|7

www.cpu-24-7.com

CD-adapco www.cd-adapco.com

www.cd-adapco.com

Dell OEM Solutions

www.dell.com/IoT

SARC Software and services for the maritime industry

www.sarc.nl

SpecTec

www.spectec.net

SSI®

www.ssi-corporate.com

NAPA

www.napa.fi

MARORKA

www.marorka.com

SERTICA By Logimatic

www.sertica.com

CADMATIC SOFTWARE SOLUTIONS

www.cadmatic.com

SENER

www.foran.es

INTERGRAPH®

www.intergraph.com

PROSTEP

www.prostep.com

This work relates to a Department of the Navy Grant issued by the Office of Naval Research Global. The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein.

15th International Conference on Computer and IT Applications in the Maritime Industries, Lecce, 9-11 May 2015**,** Hamburg, Technische Universität Hamburg-Harburg, 2016, ISBN 978-3-89220-690-3

Schriftenreihe Schiffbau
Schwarzenbergstraße 95c
D-21073 Hamburg
http://www.tuhh.de/vss

Index

VISIR: A Free and Open-Source Model for Ship Route optimization

Gianandrea Mannarini, Centro Euro-Mediterraneo sui Cambiamenti Climatici, Lecce/Italy, gianandrea.mannarini@cmcc.it
Nadia Pinardi, Università di Bologna, Bologna/Italy, n.pinardi@sincem.unibo.it
Giovanni Coppini, Centro Euro-Mediterraneo sui Cambiamenti Climatici, Lecce/Italy, giovanni.coppini@cmcc.it

Abstract

VISIR (discoVerIng Safe and effIcient Routes) represents an attempt to build a fully open ship routing model. This is achieved through a GPL licensing of the source code and detailed model documentation on open-access journals. This way, numerical optimization methods, hydrodynamic effects considered, approximations used, and their ranges of application are documented and made available to the research, technical, and even business communities. VISIR's main architectural choices, main logical components, and an outline of possible goals for a future community of VISIR developers and users are presented here.

1. Introduction

Several commercial weather routing software programs for ships are currently available. They often have appealing features in terms of vessel modeling and algorithms used for route computation and optimization, *Walther et al. (2014)*. However, as their source code is not available and related technical documentation is poorly published, such products partially act as "black boxes" with respect to how the nominal product features are actually implemented. This situation is likely to hinder possible breakthroughs arising from a collaborative scientific environment, or an "open science" paradigm, *Pontika (2015)*. Given the global significance of maritime transportation and the push for increasing its overall efficiency and in particular for reducing its environmental impact, *Mannarini (2015a),* such further developments should be more than fostered.

As an outcome of publicly funded research (projects TESSA http://tessa.linksmt.it, IONIO http://www.ionioproject.eu/, AtlantOS http://atlantos-h2020.eu/structure/overview/), VISIR ([vi'zi:r], discoVerIng Safe and effIcient Routes) represents an attempt to build a fully open ship routing model. This includes a copylefted (GPL v.3) free and open-source code, www.visir-model.net, *Mannarini et al. (2015b)*, and an open review of related scientific publications on open access journals. This way, hydrodynamic effects, numerical methods, approximations used, and their ranges of application are extensively documented and made publicly available.

It is not just the research and technical community that benefits from this approach, but possibly also the business community. As a matter of fact, the open setup of the VISIR model did not prevent the building of an operational, user-oriented Decision Support System for safer and efficient navigation in the Mediterranean Sea, *Mannarini et al. (2016)*.

Whether for science or for business, numerical code quality remains a top priority in the agenda of maritime stakeholders, as it may indirectly affect the safety of navigation, *Lee and Alexander (2013)*. This is why in the construction of the VISIR source code we attempted to follow the best practices from software engineering, *Wilson et al. (2014),* though we recognize that the system is still far from being a full-featured industrial product. Nevertheless, we are glad to offer the best of our efforts for enabling and easing further improvements of VISIR.

2. Ship route optimization

VISIR aims at optimizing nautical routes in the presence of both static and dynamic constraints arising from the ocean topology and environmental conditions, *Mannarini et al. (2015b)*.

Motor vessels with displacement hulls are considered in the first version of the model (VISIR-I). Friction and form resistance can be parameterized as polynomials in the Froude number. The peak value of wave added resistance is assumed to scale linearly in terms of Froude number and as a power-law with respect to the principal particulars (length, beam, draught), *Alexandersson (2009)*. The spectral and angular dependence of the wave added resistance is neglected in VISIR-I.

Vessel intact stability is considered through checks for the activation of parametric roll, pure loss of stability, and surf-riding/broaching-to conditions, *Belenky et al. (2011)*.

The optimization algorithm is based on a graph search method by *Dijkstra (1959)*. The graph itself is represented as a list of edges and, for enhancing computational performance, the algorithm uses a "forward star" data structure, *Ahuja et al. (1988)*. Furthermore, VISIR allows for voluntary speed reduction whenever useful in order to skip route diversions arising from the safety checks.

VISIR is suited for use in the presence of complex topology: routes can be optimized even in sea domains containing peninsulas, islands, and archipelagos. Besides a shoreline and a bathymetry database, the model needs an input forecast or analysis fields of sea state (significant wave height, wave period, and direction). In the current version of the software, they are provided through netcdf files from operational runs of the Wave Watch-III model in the Mediterranean Sea, *Tonani et al. (2014)*.

3. Code quality and structure

VISIR model was designed to ingest operational meteo-oceanographic forecasts and eventually become the engine of a new operational Decision Support System, *Mannarini et al. (2016)*. Thus, the model had to be robust enough to handle diverse and extreme field values from the operational data feeds. This is why in developing the code attention was given not just to the scientific soundness of the output, *Mannarini et al. (2015b)*, but also to the quality of the computer program and its layout, to aid future development by multiple programmers. In the next two subsections we provide more information regarding these topics.

3.1. Best coding practices

VISIR-I is coded in Matlab®. The code of VISIR was designed and implemented with an effort to follow the guidelines from software engineering manuals and collections of best practices in scientific programming, such as those reported by *Wilson et al. (2014)*. Nevertheless, VISIR-I is not an industrial product stemming from consolidated requirements. Rather, it is research software developed and re-designed on the fly while at least part of the user-requirements was still being collected, and literature analysis and community interaction were in progress. Thus, some of the best practices have strictly been followed since the beginning, while others are left for the forthcoming development of the code. In general, compliance with principles, *Lee and Alexander (2013),* and ISO norms (especially ISO/IEC 9126) of software quality assurance should be pursued for any (and especially for) safety-related software products, such as those for maritime applications.

In VISIR we considered and implemented following principles:

- Modularity. The code is organized into high-level modules, accessed by the main function and corresponding to the main logical steps of execution (Tier #1 functions in Fig.1). Each module is further structured, including four total tiers of functions, from the level of the principal functions, down to the utility functions. A "design by contract" approach was applied at least for the more general functions. There are in total more than one hundred functions making up the VISIR distribution and their average size is about 70 LOC, Table I.
- Semantic consistency. The backbone of the VISIR filesystem is the set of input, code, and output directories shown in Fig.1. The choice of array names and functions (e.g. *_edges, *_Inset, readout_*.m, write_*.m) is meant to be self-consistent. Furthermore, Matlab® structures are used to store nature constants (e.g. "const.rho_water" or "const.g0" in settings.m) and various software parameters.

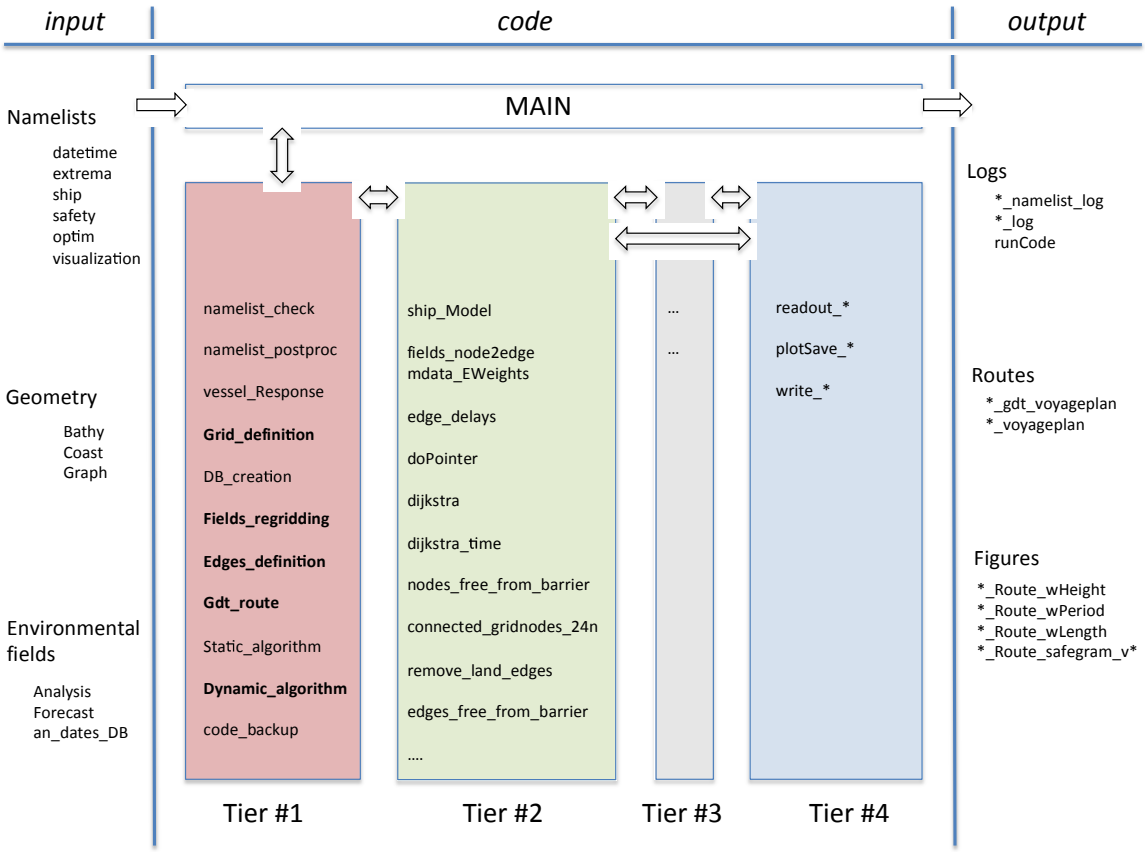

Fig. 1: Scheme of VISIR package structure. Arrows indicate the information flow direction. MAIN is the function where code execution starts and ends. Names in bold within Tier#1 indicate principal functions. File extensions are omitted. The Tiers are defined in Section 3.2.

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
Edges_definition	function	1	35.399 s	53.9%	
Dynamic_algorithm	function	1	13.644 s	20.8%	
Gdt_route	function	1	12.495 s	19.0%	
Fields_regridding	function	1	2.571 s	3.9%	
Grid_definition	function	1	0.847 s	1.3%	
code_backup	function	1	0.207 s	0.3%	
vessel_Response	function	1	0.114 s	0.2%	
close	function	1	0.094 s	0.1%	
readout_namelists	function	1	0.029 s	0.0%	
namelist_check	function	1	0.016 s	0.0%	
clearvars	function	1	0.011 s	0.0%	
strcat	function	4	0.011 s	0.0%	
eta_Job	function	1	0.004 s	0.0%	
namelist_postproc	function	1	0.004 s	0.0%	
num2str	function	1	0.003 s	0.0%	
settings	function	1	0.002 s	0.0%	
Self time (built-ins, overhead, etc.)			0.267 s	0.4%	
Totals			65.719 s	100%	

Fig. 2: Profiling of VISIR execution for N=39 780, with Tier#1 functions ranked by computing-time consumption. Self time is time spent within a given function, excluding time spent within its child functions. Self time also includes overhead resulting from the process itself of profiling.

Table I: VISIR code metrics

		notes
Number of Functions	128	
Lines of Code (LOC)	10 902 (8 952)	Including all comments and blank lines (Excluding 15 lines of license header per file)
Average lines per function	85.2 (69.9)	``
Disk space occupation	1088 kB	

- DRY ("Don't Repeat Yourself") principle. There is a single place in the code where each constant quantity is defined. Hardcoded information is deprecated, and it is grouped into external "namelist" files (see Fig.1 and Table II) or in the namelist_postproc.m and settings.m functions.
- Checks. The user-defined parameter space is quite large and the code cannot provide a correct answer for any input value. E.g. departure date cannot be earlier than the first time step in the forecast file used and ship length cannot be a negative quantity. This and similar checks are performed within the namelist_check.m function and on derived variables in the body of the code (see e.g. the checks ending with exit() in readout_envFields.m or Grid_definition.m).
- Tracking. The initial development of the code spanned a relatively long time (about three years) and the capacity to track changes and recover previous states was crucial, even though a full-featured versioning system was not employed. Thus, for each VISIR run, all input parameters, names of used files, and even the run code suite are saved to the output directory of the job run (*_namelist_log.txt, *_log.txt, runCode.tar).
- Regression tests. For the aforementioned reason, it is essential to always have tested runs, whose output files act as a reference. Some of them are provided along with the VISIR distribution.

Table II: Namelist files driving VISIR code execution

Namelist file name	purpose
extrema_pars.txt	Start/end position, bufferzone of bounding box, minimum offshore distance
datetime_pars.txt	departure date and time
ship_pars.txt	vessel parameters
safety_pars.txt	flags for intact stability constraints
optim_pars.txt	optimization options
visualization_pars.txt	rendering options

- Oracles. Any possible comparison with benchmarks is exploited. For the shortest path routine an analytical result is available for a specific spatial configuration of ship speeds, *Mannarini et al. (2015b)*. It served as a validation for the code development, and it is left in the distribution as a tool for checking that the core code is not corrupted (see forcing.analytic flag in namelist_postproc.m).
- Profiling. Once the code reached maturity, its performance was profiled in order to highlight bottlenecks. E.g. Figs. 2 and 3 show that, for larger simulations, the most time consuming part is the preprocessing of the graph quantities ("edges"), followed by the calls to shortest path routine. Not just CPU resources but also RAM memory requirements were profiled, and an example is provided in Fig. 4.
- Portability. The code was developed on a workstation and a laptop computer and it is meant to be run operationally on a supercomputing facility. Thus, code portability has been a must from its inception. An installation wizard eases the porting of the distribution (though just *nix systems are supported by us, an extension to Windows systems is possible).
- Documentation. The internal code documentation is focused on the interfaces rather than on individual method descriptions. Though documentation is not yet fully developed with respect to the design-by-contract standards, a compact user manual is provided along with the code distribution. It is complemented by other open-access resources such as the present paper and, for a technical description of the ship routing model, by *Mannarini et al. (2015b)*.

3.2. Package organization and performance

The organization of the VISIR package and its performance documented here correspond to the code version of 2016-02-09 and to the results achieved on an iMac with 3.5 GHz Intel Core i7 processor with 32 GB RAM memory (1600 MHz DDR3). The code suite is organized into input, code, and output folders, see Fig.1. There are input subfolders containing static and dynamic databases (shoreline, bathymetry, and graph connectivity information: "geometry/" and environmental field forecasts: "fields/"), as well as subfolders containing the namelists (see Table II) relative to the individual job runs ("in/<job_name>"). There are then output subfolders matching the input ones ("out/<job_name>"). The code functioning starts and ends within the MAIN.m function. Tier 1 functions are those directly called by MAIN.m and include, for instance, the reading of the namelist and environmental field files, the basic-level checks, the graph pre-processing, and the calls to the shortest path routine, Fig. 1. Tier 2 functions are those directly invoked by Tier 1 functions. Tier 4 is made up of utilities that are called both by Tier 3 and Tier 2 functions.

VISIR computing time depends on route length, through the graph size N, Fig. 3. The log-log scale and the polynomial fits of the performance, Table III, indicate that there is a linear regime for small N and a quadratic one for large N. The main contributors to such time expenditure are the graph edge preparation and the two calls (geodetic and optimal route) to the shortest path routine. As seen from Table III, the edge preparation outweighs the sum of the geodetic and optimal route computation time. Furthermore, the computational cost for edge preparation is dependent on actual sea conditions: the rougher the sea, the lower the sustained ship speed, and consequently the longer the route duration. This results in a larger number of forecast time steps needed, and, in turn, a longer edge pre-processing time. Furthermore, edge preparation is quite a memory consuming process, and memory allocation may grow significantly for large graphs, Fig. 4. The output consists in log files and the actual route information (for both the geodetic and the optimal route, Table IV). Furthermore, a geographical rendering based on third party software libraries (m_map package) is provided along with the VISIR distribution. The output file names contain a prefix indicating whether VISIR was run using a static (0_) or a time-dependent (2_) version of the shortest path routine.

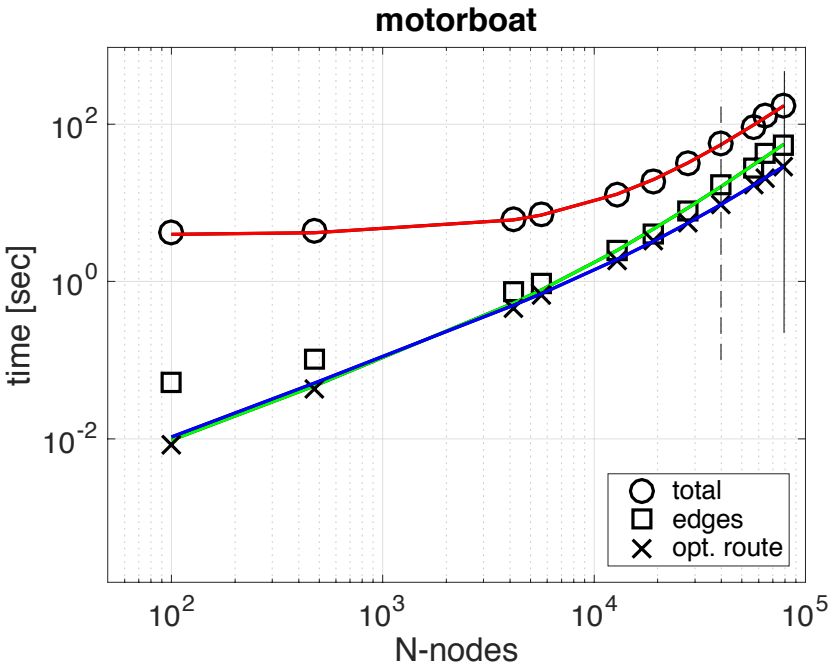

Fig. 3: VISIR code performance as function of the number N of graph grid points within the selected bounding box. The fit model and the parameters of the fitted curves are provided in Table III. The vertical dashed (solid) line indicates results obtained for value N=39 780 (N= 79 068), corresponding to the CPU time and RAM memory profiling shown in Figs. 2 and 4, respectively.

A few key technical choices made for the VISIR code are highlighted in the following:

- Use of a vessel Look-Up-Table (LUT). Sustained ship speed in various sea states and engine throttle values is computed within ship_Model.m. The LUT is then interpolated to the actual sea state corresponding to the various (time – dependent) edges, speeding up computations.
- Use of a limited spatial domain. As in any graph search method, grid size N is the key factor affecting the CPU time for computing the routes (see Fig.3 and Table III). In order to reduce the time, just a limited subset of the environmental fields is used. The user specifies the extent of such "bounding box" through four parameters for the buffer zone around the box, with departure and arrival as vertices (see extrema_pars.txt in Table II). This way, the number of gridpoints is easily reduced from about 10^6 (whole Mediterranean Sea grid at 1 nautical mile resolution, see *Mannarini et al. (2015b)*) to typically a few 10^4.
- Search of sea-next grid point. Even in case that the VISIR user specifies a departure or an arrival point located on the landmass, the code recovers the next sea grid point, compatible with a positive Under Keel Clearance (UKC) and the minimum offshore distance set by the user. In order to do so, VISIR computes a joint mask, accounting for both vessel UKC and shoreline distance (Grid_definition.m).

Table III: Performance of VISIR components shown in Fig. 3. The fit model for the computing time τ is: $\tau = c_0 + c_1 N + c_2 N^2$. The parameter c_0 is constrained to 0, except for the fitting of the total job performance.

	c_0 [s]	c_1 [s]	c_2 [s]	R^2 [%]	*RMSE* [s]
opt	0	1.1e-04	3.3e-09	99.9	0.1
edges	0	9.6e-04	7.7e-09	98.9	2.1
tot	3.8	4.2e-04	2.2e-08	99.4	4.9

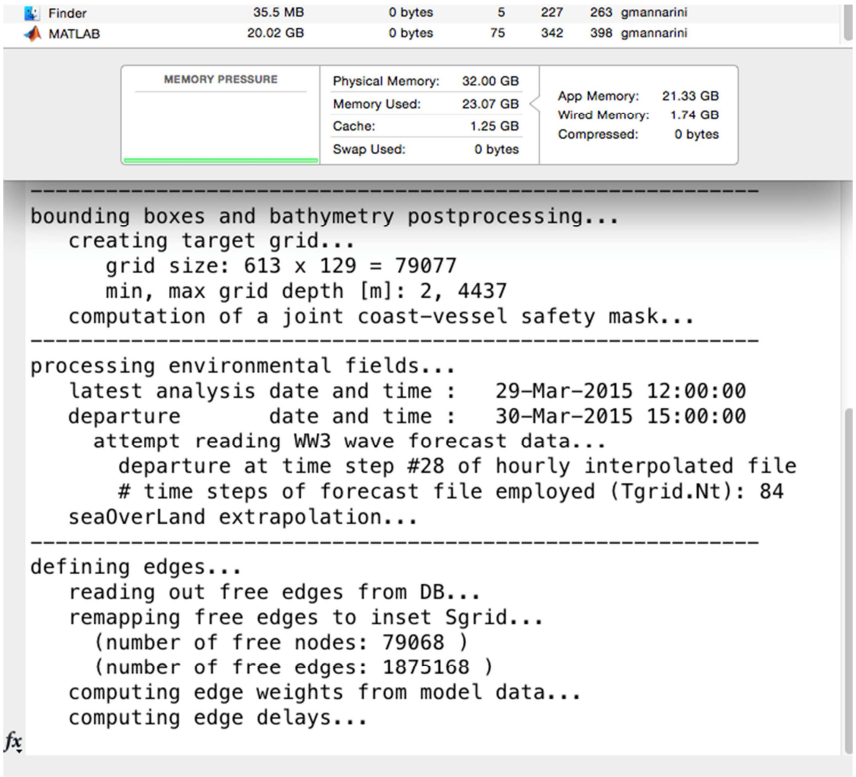

Fig. 4: VISIR RAM-memory usage at peak and excerpt of the messages output to the command line. The number of graph grid-points used in this route computation was N= 79068.

- Use of pointers for the graph edges. In exact graph search methods such as Dijkstra's algorithm, there is an unavoidable work load due to the scan of all the edges for checking the

"complementary slackness conditions" *Bertsekas (1998)*. In particular, they are checked sequentially for each candidate node on the shortest path. In order to speed up the computations, it is convenient to adopt the viewpoint of the edges emanating from a given node ("forward star", see *Ahuja et al. (1988)*). In the absence of a corresponding data structure available in Matlab®, the forward star is implemented through an explicit provision of pointers to each node in the list of graph edges. They are prepared in the doPointer.m function and then employed in both the dijkstra.m (static algorithm) and dijkstra_time.m (time-dependent algorithm) optimization functions.

4. Example of results

In order to illustrate the results that can be obtained via VISIR, we provide here a full-featured description of a relatively long sea route, in the presence of a complex domain (coastline, islands, and a strait) and rough enough seas. This highlights the time-dependent structure of the algorithm, its capacity to retrieve the route in a complex topology, and the strategic savings it offers.

The route is set in the Ligurian and Tyrrhenian Seas, between Cannes (France) and Porto Cervo (Italy), during a typical Mistral event. It starts on 2016-02-19 at 17:00 UTC. Sea state forecasts from Wave Watch-III model stemming from an analysis for 2016-02-19 at 12:00 UTC provide the significant wave height, period, and direction of the peak wave spectrum component. A vessel with parameters from Table V is employed for the computations.

The route job, excluding maps and time series rendering, took (on the same computer described in Sect.3.2) $\tau = 47 \pm 1$ s, with a peak Matlab® RAM memory allocation of about 4.7 GB.

Table IV: Route model stored by VISIR. The column "Type" specifies the graph element to which the attribute refers (n: node; e:edge). The index-like attributes (last 3 rows) are available for the geodetic route only.

Attribute	Description	Units	Format	Type
WP	Waypoint index	-	integer	n
ISO_date	Date and time	-	ISO 8601	n
lon	longitude	deg E	real	n
lat	latitude	deg N	real	n
cum_dist	cumulative distance	NM	real	n
course	Course over ground	deg	real	e
throttle	Engine throttle	%	real	e
speed_out	Speed over Ground	kn	real	e
alpha	wave-ship relative direction	deg	real	e
T_E	Encountered wave period	s	real	e
redLambda	Wavelength divided by ship length	-	real	e
swh	Significant wave height	m	real	e
UKC	Under Keel Clearance	m	real	e
parRoll	Parametric roll index	-	integer	e
pureLossStab	Pure Loss of Stability index	-	integer	e
surfRid	Surf-riding index	-	integer	e

The four panels in Fig. 5 (a-d) display four snapshots of both the significant wave height and direction fields, and the progress of both the geodetic and optimal route computed by VISIR. The geodetic route minimizes spatial distance between departure and arrival. It considers the shoreline, checks for the condition UKC>0, and uses the sea state information just for computing the sustained speed at each route leg. The optimal route, besides checking for both shoreline distance and UKC>0, keeps into account vessel intact stability as a constraint, and total time of navigation as an optimization objective (see Section 2). It is seen that the different objectives and constraints for the two routes lead to quite different paths. In fact, one route keeps Corsica to the port side (geodetic), while the other to

starboard (optimal). This way, the optimal route is significantly longer in distance than the geodetic one, but still 7% shorter in navigation time (see Table VI). This is due to the calmer sea experienced east of Corsica, the consequent reduction in wave added resistance, and involuntary vessel speed loss. The kinematics of the routes is further explored in Fig. 6(a-d).

The time series for sustained speed clearly shows that, for the optimal route, when the vessel is navigating in calm seas in the lee of more than 1000 m a.s.l. high mountains, Fig. 5b-c), there is a sudden rise of sustained speed, up to the top speed c (Fig. 6a at Δt >13 h after departure). Along the optimal route, speed is up to four knots larger than along the geodetic one, and this allows for overall savings in navigation time, despite a longer sailed path.

Panel Fig. 6b monitors the wavelength λ with respect to vessel length L. In fact, a specific value of the λ/L ratio is one of the necessary conditions for the loss of intact stability, *Mannarini et al. (2015b)*. This occurs along the geodetic route at Δt>25 h, Fig. 6d). However, it cannot occur along the optimal route, since route legs leading to stability loss cannot be employed by the optimization algorithm, per construction (if corresponding flags in the safety_pars.txt namelist are set to 1, Fig. 1.

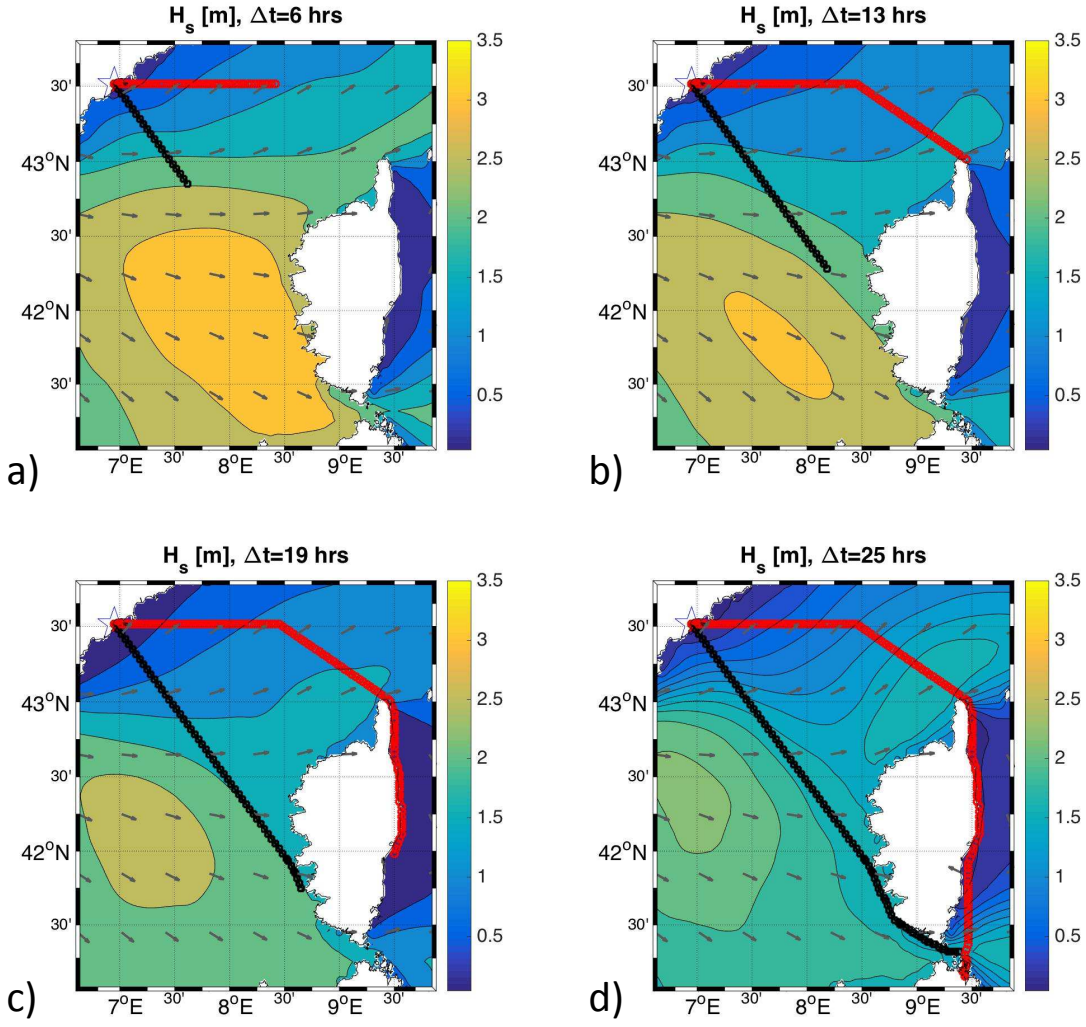

Fig.5: Geodetic (black markers) and optimal (red markers) route from Cannes (France) to Porto Cervo (Italy) for a displacement hull vessel with the parameters from Table V and departure on 2016-02-19 at 17:00 UTC. Panels *(a–d)* refer to various time steps after departure. Significant wave height forecast fields H_s are displayed by coloured shadings and wave directions by arrows. The graph generated by VISIR for solving this route optimization task has a size N=27927.

Finally, the chosen route also demonstrates VISIR's capacity to suggest a voluntary speed reduction. This occurs at Δt=25 h along the optimal route, Fig. 6c. The algorithm suggests reducing the engine throttle to 85% for a few minutes long to avoid stability loss. The possible route diversion to avoid throttle reduction and keep from stability loss in fact would lengthen the route, which would not offset the time saving due to keeping engine throttle at its maximum value.

Table V: Values of the ship parameters used for the VISIR routes shown in this manuscript

1	Maximum engine brake power	650	hp
2	top speed	10.7	kt
3	length at waterline	22	m
4	beam (width at waterline)	6	m
5	draught	1.9	m
6	natural roll period	5.4	s

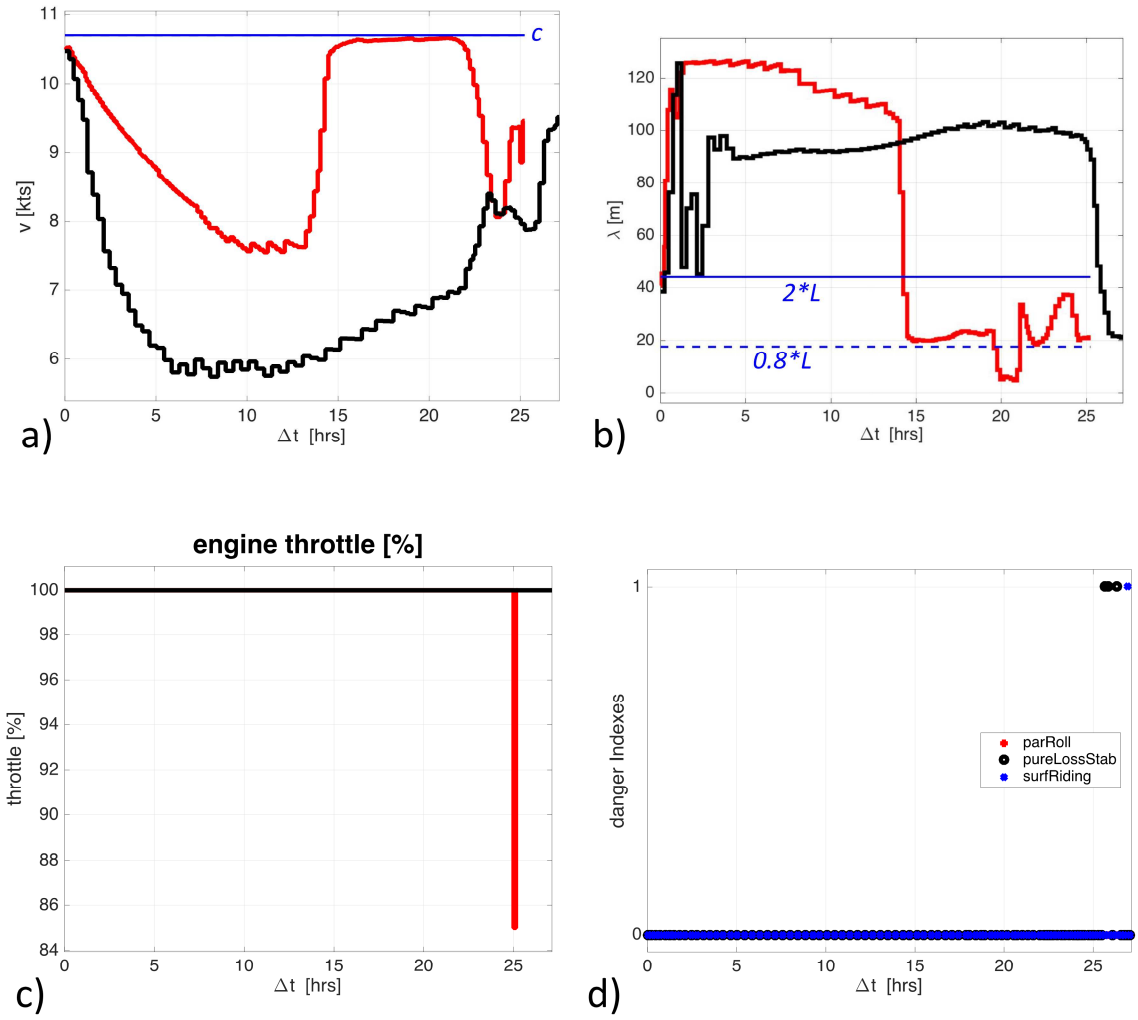

Fig. 6: Information along geodetic (black) and optimal (red) route of Fig. 5. Δt is time elapsed after departure. *(a)* sustained speed v and reference line at maximum speed c; *(b)* wavelength λ and reference lines at vessel length multiples *0.8L* and *2L*; *(c)* engine throttle; *(d)* danger indices along geodetic route, 0: safe; 1: dangerous.

Table VI: Summary metrics for the route computation shown in Fig. 5 and Fig. 6. The relative differences are computed as *Δ= 100*(Optimal/Geodetic - 1).*

	Units	*Geodetic*	*Optimal*	*Δ□□□□*
Length	NM	187.38	233.56	+24.6
Duration	hh:mm:ss	27:08:31.1	25:14:37.4	-7.0
Average speed	kt	6.90	9.25	+34.0

5. Conclusions

This paper presents some features of the numerical code of ship routing model VISIR. The model is made free and open-source, and its documentation is open-access. This should enable the formation of a community of users and developers.

A few principles of robust scientific programming were adopted. Even before adding new functionalities or better parameterizations of vessel response, there is certainly room left for code performance optimization (see e.g. discussion about edge pre-processing in Section 3.2), redesign by-contract, and internal documentation. Full compliance with code quality standards such as ISO/IEC 9126 or ISO/IEC 25010:2011 would be also beneficial. Refactoring into an object-oriented language is an attractive option and a research activity in this direction has already been started at CMCC.

Furthermore, the authors foster the creation of a VISIR developer consortium and a large community of users. Non-commercial agreements with companies are also possible. Cooperation with the industry would be especially useful for collecting feedback from the maritime end-users, while the research community might consider VISIR as an open platform for testing optimization methods, vessel response parameterizations, and the impact of various types of environmental data relevant to navigation.

References

AHUJA, R.K.; MAGNANTI, T.L.; ORLIN, J.B. (1988), *Network Flows*, MIT, Cambridge

ALEXANDERSSON, M. (2009), *A study of methods to predict added resistance in waves*, Master thesis, KTH Centre for Naval Architecture, Stockholm
www.kth.se/polopoly_fs/1.151543!/Menu/general/column-content/attachment/Alexandersson.pdf

BELENKY, V.; BASSLER, C.G.; SPYROU, K.J. (2011), *Development of Second Generation Intact Stability Criteria*, 5 Tech. Rep., DTIC Document
http://www.uscg.mil/hq/cg5/cg5212/docs/dtmb-2ndgen-is-rpt2011.pdf

BERTSEKAS, D.P. (1998), *Network Optimization: Continuous and Discrete Models,* Athena Scientific, http://web.mit.edu/dimitrib/www/netbook_Full_Book.pdf

DIJKSTRA, E.W. (1959), *A note on two problems in connexion with graphs,* Num. Math. 1.1, pp.269-271

LEE, S.; ALEXANDER L. (2013), *Software Quality Assurance Issues Related to e-Navigation,* Marine Navigation and Safety of Sea Transportation: Advances in Marine Navigation, CRC Press

MANNARINI, G. (2015a) *The twofold aspect of climate change on navigation: the search for new maritime routes and the challenge of reducing the carbon footprint of ships,* CMCC Research Paper RP0252, http://www.cmcc.it/wp-content/uploads/2015/03/rp0252-opa-03-2015.pdf

MANNARINI, G.; PINARDI, N.; COPPINI, G.; ODDO P.; IAFRATI A. (2015b), *VISIR-I: small vessels, least-time nautical routes using wave forecasts,* Geosci. Model Dev. Discuss. 8, pp.7911-

7981, http://www.geosci-model-dev-discuss.net/8/7911/2015/gmdd-8-7911-2015.pdf

MANNARINI, G.; TURRISI, G.; D'ANCA, A.; SCALAS, M.; PINARDI, N.; COPPINI, G.; PALERMO, F.; CARLUCCIO, I.; SCURO, M.; CRETÌ, S.; LECCI, R.; NASSISI, P.; TEDESCO, L. (2016), *VISIR: Technological infrastructure of an operational service for safe and efficient navigation in the Mediterranean Sea,* Nat. Hazards Earth Syst. Sci. Discuss. 32, pp.1-19
http://www.nat-hazards-earth-syst-sci-discuss.net/nhess-2016-32/nhess-2016-32.pdf

PONTIKA, N., KNOTH, P., CANCELLIERI, M., PEARCE, S. (2015), *Fostering open science to research using a taxonomy and an eLearning portal*, 15th Int. Conf. on Knowledge Technologies and Data-driven Business, ACM, p.11, http://libeprints.open.ac.uk/44719/2/kmi_foster_iknow.pdf

TONANI, M., ODDO, P., KORRES, G., CLEMENTI, E., DOBRICIC, S., DRUDI, M., PISTOIA, J., GUARNIERI, A., ROMANIELLO, V., GIRARDI, G., GRANDI, A., BONADUCE, A., PINARDI, N. (2014), *The Mediterranean Forecasting System: recent developments*, Geophys. Res. Abstr., EGU2014-16899-1, EGU General Assembly, Vienna. 7939
http://www.tandfonline.com/toc/tjoo20/current#aHR0cDovL3d3dy50YW5kZm9ubGluZS5jb20vZG9pL3BkZi8xMC4xMDgwLzE3NTU4NzZYLjIwMTUuMTA0OTg5MkBAQDE=

WALTHER, L.; BURMEISTER, H.C.; BRUHN, W. (2014), *Safe and efficient autonomous navigation with regards to weather*, 13th Int. Conf. on Computer and IT Applications in the Maritime Industries, Redworth, pp. 303–317,
http://data.hiper-conf.info/compit2014_redworth.pdf

WILSON, G.; ARULIAH, D.; BROWN, C.T.; HONG, N.P.C.; DAVIS, M.; GUY, R.T.; HADDOCK, S.H.D.; HUFF, K.D.; MITCHELL, I.M.; PLUMBLEY, M.D.; WAUGH, B.; WHITE, E.P.; WILSON, P. (2014), *Best practices for scientific computing,* PLoS Biol. 12/11
http://iacs-courses.seas.harvard.edu/courses/cs207/resources/BestPratices.pdf