An Approach to Trip- and Route-Planning Problems

GEORGII KHACHATUROV
Departamento de Sistemas, Universidad Autónoma Metropolitana- Azc.,
Av. San Pablo 180, Col. Reynosa Tamaulipas, C.P. 02200, D.F., Mexico.

Abbreviated title: An Approach to Trip- and Route-Planning

Some practical planning problems can be interpreted as set-to-set shortest path problem (spp), i.e. as search of a shortest path between two sets of nodes, A and B, of a graph G. A straightforward reduction of such a problem to the search of solutions for point-to-point spps is impractical because the computational complexity is too high for a huge G. This paper presents a new approach to set-to-set spp for the case of not arbitrary A and B, but those which are represented by some nodes of an additional graph T. The graph T simulates a "geographic system" on G. Under some assumptions natural for many applications, this approach leads to a competitive algorithm for this kind of set-to-set spp. As prospective areas for this technique, two applications are discussed -- the problem of route planning for a visually guided robot in a static environment, and the problem of planning a fastest trip by means of all available timetables of all kinds of transport.

INTRODUCTION

Motivation

Numerous papers (see [Sharir, 1997] for a survey) treat the robot motion planning as the following problem of Cartesian navigation: *Given goal and initial states of a robot as some points in an Euclidean space, find a path avoiding obstacles between these points.*

Some talks between the author and Prof. E. Dantsin from the Steklov Institute of Mathematics at St.-Petersburg, Russia, were very useful for this work. These discussions were possible thanks to the TET-A-TET program of the Euler International Mathematical Institute and support by CONACYT, Mexico (grant 400200-5-1453PA)

Address correspondence to Georgii Khachaturov, Departamento de Sistemas, Universidad Autónoma Metropolitana- Azc., Av. San Pablo 180, Col. Reynosa Tamaulipas, C.P. 02200, D.F., Mexico.
Home page: http://centaury.uam.mx/~xgeorge/

This interpretation of navigation attracts researchers because it avoids use of complicated models of environment. Instead, the algorithms of path finding modify a current path reacting to a new obstacle. At the beginning of search, the robot may have zero-knowledge about its environment.

On the other hand, it is clear that for most of natural navigational problems the above Cartesian interpretation is inadequate. For example, the task "*go to the dining room*" does not assume use of any Cartesian system. In addition, if robot has already studied its environment, it is not natural to involve any exploration in planning this task.

What mathematical representation of robot environment fits better for posing and solving navigational problems, if this environment is already perfectly studied?

The following graph-based approach represents an attempt to respond to this question.

Let a model of robot environment be represented by directed graph $G = (V_G, E_G)$, for which the nodes of V_G and the edges of E_G have the following interpretation.

Let S denote the space of all robot poses inside an environment. It is assumed that *robot views* are put in one-to-one correspondence with points of S. Each node introduces a rule declaring some close views as equivalent. So each node of V_G codes a domain of S and, consequently, the set of nodes V_G represents a sampling of S by equivalent robot views.

An edge $l \in E_G$, where $l=(a,b)$ and $a, b \in V_G$, represents a rule for moving the robot from any state coded by a to a state coded by b. So execution of this rule changes any robot view associated with a to a view associated with b.

If weights $|l| \geq 0$ are assigned to all $l \in E_G$, then a simple interpretation of a navigational problem is presented by the well-known point-to-point shortest path problem (*spp*) in G: *Given* $a \in V_G$ *to* $b \in V_G$, *find a path from a to b with minimal summary weight*.

This approach [Khachaturov, 1999A] is close to the one of [Shölkopf and Mallot, 1995], which will be discussed in the last section.

However, this approach is not adequate yet to the navigational problems posed in natural terms. The above task "*go to the dining room*" does not specify any particular view to reach. It requires achieving any of a huge number of views. In terms of graph *G*, a more adequate problem is: *Find a path from a node to a set of nodes*.

Interpreting a natural navigational problem with such a point-to-set problem on graph, we can see that the goal set (*dining room*) is not an arbitrary set of nodes of *G*. Those goal sets form a specific system of subsets of V_G. This system does not participate in resolution of navigational problems by the methods known so far.

So, what mathematical model represents adequately the goal sets and how can it be used for search?

Furthermore, there exists a straightforward method to find a path for this point-to-set graph problem: *For all nodes of the goal set, execute cyclically an algorithm that finds a solution for point-to-point spp*. However the complexity of this method is too high.

Indeed, denoting by $|V_G|$ the size of V_G, the complexity of search for point-to-point *spp* by the Dijkstra's scheme [Dijkstra, 1959] is bounded [Leeuwen, 1990] by a function linear in $|V_G|^2$, or in $|E_G|+|V_G|log|V_G|$ if special data organization is used [Fredman and Tarjan, 1987]. To obtain the estimates of complexity for the straightforward method, these values should be multiplied by the number of nodes of the goal set. Hence, the straightforward method is unrealistic for a huge number of views in real scenes.

How can the complexity be reduced?

Description of Results

The approach presented in this paper gives certain responses to the above questions. It extends and refines the results of (Khachaturov, 1999B).

The same interpretation of graph *G* as above is used.

In addition to G, another directed graph T is introduced into the model which we intend to use for representation of a typical navigational problem. T simulates a kind of geographic or/and administrative organization of G.

By definition, T has a unique *source node* (i.e. the node with no in-coming edges), whereas the set of *terminal nodes* (nodes with no out-coming edges) coincides with V_G.

Each node α of T determines in a natural way a set $V_\alpha \subset V_G$. Informally, V_α represents a geographic object in G the name of which is α. Any edge of T stands for the relation of usual inclusion of geographical objects. That is, if l is an edge of T, and α and β are, respectively, the start- and end-nodes of l, then the geographic object labeled by α is wider than the one labeled by β.

Given pair $\{G,T\}$, a navigational task is represented mathematically as the following problem: *Let α and β be nodes of graph T, and let V_α and V_β be the corresponding subsets in V_G. Find a single shortest path that begins inside V_α and ends inside V_β.*

This is as an extension of *spp* on graph G. Under some assumptions quite natural for many applications, an efficient algorithm is developed that finds a solution for this extended *spp*.

Then this approach is modified to the problem of planning a fastest trip.

Note that actually many airline companies give customers the opportunity to plan interactively the flights for a trip from one airport to another. It works as an on-line Internet application. This is a typical example of the “planning-a-trip” problem. However, such a system does not cover all needs of a customer living in a small village who plans to go to another small village abroad. At the time of writing, there is no system yet able to process such a highly specified query taking into account the timetables of all available kinds of transport including local buses, for instance. The presented approach can be useful for this purpose.

For both prospective applications, the presented approach means that naturally posed navigational problems can be resolved efficiently provided that corresponding environment is represented adequately by

a pair $\{G,T\}$. Hence, future research and developments should be focused on construction of the components of this pair. The paper is concluded with a discussion on this point.

GT-MODEL: DEFINITION AND INTERPRETATION

This and the following section represents a navigational problem in terms of a new model (called below *GT-model*) composed by two graphs.

Definition 1. In formal terms, a *GT-model* is defined as pair $\{G, T\}$, where the components satisfy the following properties:

- $G=(V_G, E_G)$ is directed graph with non-negative weights assigned to its edges
- $T=(V_T, E_T)$ is directed graph which satisfy the following condition:

  T has a unique source-node whereas the set of its terminal nodes coincides with V_G.

An interpretation of this formal definition follows under assumption that a *GT*-model represents states of a visually guided robot in its environment. Physically, this environment is assumed to be static.

The interpretation of graph *G* for this example was already given in the Introduction.

Graph *T* represents a system of some sets composed by nodes of *G*. This system is determined by the following rule: *Any node* α *of* V_T *corresponds to set* $V_\alpha \subset V_G$ *defined as the set of all terminal nodes of all paths in T that start from* α.

Definition 2. This set V_α will be called *the geographical domain represented by* α, or, for brevity *"domain of* α*"*.

As any terminal node of T corresponds to a node of V_G, then due to the interpretation of the Introduction, it represents some close states of a robot inside its environment. Unlike that, for a non-terminal $\alpha \in V_T$, its domain V_α can contain far robot states. Graph T organizes such domains into a system quite similar to a geographical system. In particular, the domain of the source-node of V_T is represented by the whole V_G.

If there is a path in T from α to β, then the domain of α obviously contains the domain of β. I.e., the geographic object V_α in G represented by α is wider than V_β represented by β.

The simplest kind of T is tree. If T is a tree, it represents so-called *tree decomposition* of G [Leeuwen, 1990].

ROUTING PROBLEM IN TERMS OF *GT*-MODEL

For given *GT*-model, let $\alpha, \beta \in V_T$ and V_α, $V_\beta \subset V_G$ be domains of α and β, respectively.

In this notation, the problem under consideration is: *Find a path in G with minimal summary weight that begins inside V_α and ends inside V_β.*

Definition 3. This problem is denoted by $\{V_\alpha, V_\beta\}$ and called *the extended shortest path problem (espp)*.

The *espp* is a set-to-set extension of point-to-point *spp* in G. If α, β are some terminal nodes of T, i.e., if $\alpha, \beta \in V_G$, then *espp* $\{V_\alpha, V_\beta\}$ coincides with a point-to-point *spp*. In general case, *espp* implies search of a single path between two sets of nodes of G.

Note that *espp* seems to be quite adequate mathematical representation of many problems of natural navigation, in particular, of the problems of non-Cartesian navigation, like "*go to the dining room*".

The rest of this section deals with *espp*.

A Database Representation of *GT*-Model

Let us assume that the memory units (sets) of a network database are put in one-to-one correspondence with the nodes of T. Based on the correspondence, it is possible to deduce a rule that determines which memory unit should store the record about an edge of G.

Definition 4. It is said that $\beta \in V_T$ *is an owner of* $\alpha \in V_T$ if an edge exists in E_T with β as the begin and α as the end. In the same case, α *is called a child of* β.

Definition 5**.** The set $\boldsymbol{B} \subset V_T \times V_T$ is defined as the set of all ordered pairs $(\mu,\nu) \in V_T \times V_T$, such that μ and ν have a common owner. (I.e., $\boldsymbol{B}$ represents the set of brother pairs of T).

Definition 6**.** For $\gamma \in V_T$, the graph T_γ is defined as a sub-graph of T determined by two properties: (*i*) γ is the unique source-node of T_γ; (*ii*) T_γ contains all those paths of T that start at γ.

For any $\alpha,\beta \in V_T$, $\alpha \neq \beta$, it is possible to find a unique $\gamma \in V_T$ satisfying the following properties:

- Any path, from the source node of graph T to α, or to β contains γ. [In particular it means that T_γ contains α and β.]
- If the first property holds for some $\lambda \in V_T$, $\lambda \neq \gamma$, then $T_\gamma \subset T_\lambda$.

Definition 7. Given $\gamma \in V_T$ that satisfies these properties, the corresponding graph T_γ is called the *minimal sub-graph of T built by nodes* α,β.

The source-node γ of minimal graph T_γ built by nodes α and β has (otherwise T_γ would not be minimal) a pair of non-equal children, say μ and ν, such that some two paths in T_γ exist from μ and ν, respectively, to α and β.

Definition 8. Such nodes μ *and* ν *are called the brother nodes built by* α *and* β. (It is clear that $(\mu,\nu)\in \boldsymbol{B}$).

If T_γ is a tree, there exists just one pair of brother nodes built by α and β. In general case, there can exist more than one pair of brother nodes built by α and β.

Definition 9. Let $(\mu,\nu)\in \boldsymbol{B}$ and V_μ, V_ν be domains of μ and ν, respectively. By $\boldsymbol{M}(\mu,\nu)$, we denote *the set of all edges* $l\in E_G$ *that begin inside* V_μ *and end inside* V_ν.

I.e., for an edge $l\in E_G$ represented as $l=(\alpha_s,\alpha_e)$, where $\alpha_s,\alpha_e\in V_G$, the relation $l\in \boldsymbol{M}(\mu,\nu)$ *holds if and only if* μ,ν *are the brother nodes built by* α_s,α_e.

An algorithm below assumes that the nodes of T are put in correspondence with the units of memory of a network database for storing the edges of E_G. By means of just the introduced terms, we are able to determine the storing rule:

Given $l=(\alpha, \beta)$, where $\alpha,\beta\in V_G$, let T_γ be the minimal sub-graph of T built by nodes α,β. *The node* γ *represents the very memory unit where the record about* $l\in E_G$ *must be stored.*

Hence, *the set of all edges of* E_G *attached in the database to the node* $\gamma\in V_T$ *has the form* $\cup_{\mu,\nu}\boldsymbol{M}(\mu,\nu)$, *where* μ *and* ν *span all children pairs of* γ.

We can informally comment this rule in the following way.

Let us imagine that to any node of *T*, which represents a geographical object in *G*, a virtual transport department is assigned. It must administrate some routes represented by edges of *G*. In these terms, *each department keeps under its own control only the routes connecting the domains of its immediate subalterns*.

I.e., for the department corresponding to a node γ, if a route passes strictly inside the domain of a child-node of γ, this route is out of this department jurisdiction. As well, if a route begins inside V_γ (i.e. inside the domain of γ), but ends outside V_γ, then a higher level department must administrate this route. The only case of keeping a route under the control of this department is if the route begins inside the domain of a child-node of γ and ends inside the domain of another of its child-nodes.

Note that for a non-tree *T*, the ends α,β of an edge of *G* can generate different pairs of brother nodes $\mu,\nu \in V_T$, but there is only one memory unit where this edge can be stored.

Analysis of *espp*

For $\alpha,\beta \in V_T$, consider *espp* $\{V_\alpha, V_\beta\}$. If $V_\alpha \supset V_\beta$ or $V_\alpha \subset V_\beta$, then $\{V_\alpha, V_\beta\}$ has a trivial solution.

Otherwise, denoting by $|\{V_\alpha, V_\beta\}|$ the length $|l_1|+|l_2|+...+|l_k|$ of a solution $(l_1, l_2, ..., l_k)$ for $\{V_\alpha, V_\beta\}$, the following equation obviously holds

$$|\{V_\alpha, V_\beta\}| = \min_{\delta \in V_\mu, \gamma \in V_\nu} (|\{V_\alpha, \delta\}|+|\{\delta, \gamma\}|+|\{\gamma, V_\beta\}|), \qquad (1)$$

where while minimizing the right-hand side, $\mu,\nu \in V_T$ scan all pairs of brother nodes built by α, β, and for each pair of them, all $\delta \in V_\mu$, $\gamma \in V_\nu$ should be revised.

So by means of Eq. (1), the original *espp* $\{V_\alpha, V_\beta\}$ spawns three new *espp*s: $\{V_\alpha, \delta\}$, $\{\delta, \gamma\}$, and $\{\gamma, V_\beta\}$ with variable μ, ν and δ, γ.

Let T_λ and $T_{\lambda'}$ be the minimal sub-graphs of *T* built, respectively, by nodes $\alpha,\beta \in V_T$ and $\alpha',\beta' \in V_T$.

Definition 10. If graph $T_{\lambda'}$ contains as its proper part graph T_{λ}, it is said that *espp* $\{V_{\alpha}, V_{\beta}\}$ *is simpler than espp* $\{V_{\alpha'}, V_{\beta'}\}$.

Proposition 11. *The espps corresponding to the first and the third summands of the right-hand side of Eq. (1) are simpler than the espp participating in the left-hand side.*

This simple observation determines a strategy of involving some heuristics to solve *espp* for practically important cases.

Namely, due to this proposition, it is possible to develop an algorithm, which, while solving an *espp*, generates recursively two simpler *espp*s for the first and the third summands of Eq. (1). The point is how to simplify the second *espp* of the right-hand side of Eq. (1), and how to simplify the traversal of all μ, ν and δ, γ.

Simplifying the Middle Term of Equation (1)

The following heuristic represents a principal step to simplify *espp* $\{\delta, \gamma\}$ of Eq. (1).

Assumption 1. *If for espp* $\{\delta, \gamma\}$ *of Eq. (1), a solution exists, then there exists a solution of* $\{\delta, \gamma\}$ *represented by a path that belongs to the minimal sub-graph of T built by nodes* μ *and* ν. [Note that this minimal sub-graph coincides with the minimal sub-graph built by α and β.]

Informally, this assumption means that to build a best route between some towns of a state, there is no need to cross a border of another state.

Definition 12. For $\mu \in V_T$, a node $\alpha \in V_\mu$ is said to be *the exit customs* of V_μ when α is the start-node of an edge of G, the end-node of which belongs to the domain V_ν of a brother node ν of μ. (I.e. if there exist $\nu \in V_T$ and $l \in E_G$, where $l = (\alpha, \beta)$, such that $(\mu,\nu) \in \boldsymbol{B}$ and $\beta \in V_\nu$).

Definition 13. For $\nu \in V_T$, a node $\beta \in V_\nu$ is said to be *the entrance customs* of V_ν when β is the end-node of an edge of G, the start-node of which belongs to the domain V_μ of a brother node μ of ν. (I.e. if there exist $\mu \in V_T$ and $l \in E_G$, where $l = (\alpha, \beta)$, such that $(\mu,\nu) \in \boldsymbol{B}$ and $\alpha \in V_\mu$).

Proposition 14. If μ, ν *scan all brother nodes built by* α, $\beta \in V_T$ *and* δ_s *and* δ_e *scan not all elements of* V_μ *and* V_ν*, but only their exit- and entrance-customs, respectively, then, under assumption 1, the following equation*

$$|\{V_\alpha, V_\beta\}| = \min_{\delta_s \in V_\mu, \delta_e \in V_\nu} (|\{V_\alpha, \delta_s\}| + |\{\delta_s, \delta_e\}| + |\{\delta_e, V_\beta\}|) \qquad (2)$$

yields the same solutions of espp $\{V_\alpha, V_\beta\}$ *as Eq. (1).*

Proof. Note that the set of solutions of Eq. (1) contains all solutions of Eq. (2). Hence, it is sufficient to prove that each solution of Eq. (1) can be found by Eq. (2).

Let us consider a path $l_{\delta\gamma}$ that represents a solution for *espp* $\{\delta, \gamma\}$ of Eq. (1). In general case, $l_{\delta\gamma}$ can leave V_μ for the first time via a node which does not represent any customs, and then be continue with a node which does not belong to the minimal graph of T built by α and β.

However, this is impossible if assumption 1 holds. In this case, $l_{\delta\gamma}$ can always be subdivided into three following parts:

The first part is the maximal part of $l_{\delta\gamma}$ that begins with δ and ends with some node δ_s defined as the last node of $l_{\delta\gamma}$ before it leaves V_μ for the first time. So δ_s is an exit customs of V_μ. Due to assumption 1, the next after δ_s node of $l_{\delta\gamma}$ belongs to the domain of some brother node of μ.

The third part (tail) is the maximal part of $l_{\delta\gamma}$ that ends in γ and contains only nodes of V_ν. It begins at some entrance customs δ_e of V_ν, via which $l_{\delta\gamma}$ enters into V_ν from a domain of a brother node of ν.

All the rest of $l_{\delta\gamma}$ represents the middle (second) part of this subdivision. Its nodes belong to the domains of some brother nodes of μ and ν.

Then, a solution $l_{\alpha\beta}$ of *espp* $\{V_\alpha, V_\beta\}$ found by Eq. (1) can be represented as $l_{\alpha\beta} = l_{\alpha\delta} \cup l_{\delta\gamma} \cup l_{\gamma\beta}$, where each term in the right represents an optimal solution of the corresponding term of Eq. (1).

It is possible to re-distribute the nodes between $l_{\alpha\delta}$, $l_{\delta\gamma}$, and $l_{\gamma\beta}$ keeping unchangeable their union. In this new representation $l_{\alpha\beta} = l_1 \cup l_2 \cup l_3$. The path l_1 is defined as the union of $l_{\alpha\delta}$ with the just defined first part of $l_{\delta\gamma}$. The path l_2 coincides with the just defined middle part of $l_{\delta\gamma}$. Lastly, l_3 represents the union of the third part of $l_{\delta\gamma}$ with $l_{\gamma\beta}$.

By construction, the last node of l_1 is represented by the exit customs δ_s, and the first node l_3 is represented by the entrance customs δ_e.

As the whole path $l_{\alpha\beta}$ is not changed after this re-distribution, it still represents a solution of the original *espp*. On the other hand, this new representation of $l_{\alpha\beta}$ belongs to the set of paths generated by Eq. (2). QED.

This observation justifies involving new heuristics to simplify furthermore search for *espp*.

Assumption 2. Let $\mu,\nu \in V_T$ have a common owner and V_μ, V_ν be the domains of μ and ν, respectively. If a path from V_μ to V_ν exists, which starts at an exit customs $\delta_s \in V_\mu$ and ends in an entrance customs $\delta_e \in V_\nu$,

then there exists an edge $l \in \boldsymbol{M}(\mu,\nu)$, $l = (\delta_s, \delta_e)$, the length of which is not greater than the length of the path.

In other words, the existence of a path between some customs of V_μ and V_ν implies the existence of a single-edge optimal path between them.

Note that due to Assumptions 1-2, if $\mu \neq \nu \in V_T$ are brother nodes built by $\alpha,\beta \in V_G$, a shortest path exists that begins inside V_α and ends inside V_β, having the form $(a,\ldots,x, y,...,b)$, where $a,\ldots,x \in V_\mu$ and $y,...,b \in V_\nu$. The nodes μ and ν are brother nodes built by α and β, and x, y are some exit- and entrance-customs of V_μ and V_ν, respectively.

To justify Assumption 2, let us note the following.

Each *espp* $\{\delta_s,\delta_e\}$ of Eq. (2) represents a point-to-point *spp*.

Practically, assumption 2 means that all these *spp*s corresponding to the second summand of Eq. (2) were solved beforehand and their solutions were represented by some edges of G (perhaps, extending it, if necessary).

In many practical cases, it is possible to bound the number of either kind of customs for all nodes of graph T by a common constant C. Then, it is sufficient to solve not greater than C^2 *spp*s to find shortest paths that connect the customs for any pair (μ,ν) of brother nodes in graph T.

Here this assumption is introduced to make clearer an algorithm that resolves *espp*. In fact, this algorithm can be simply modified for the case without this assumption. However, such a modification leads to a greater algorithmic complexity.

So assumption 2 is not essential. In practice, keeping assumption 2 true is the point of a specific permanent maintenance of the database representing *GT*-model.

Note that assumption 2 is justified *only* if assumption 1 holds.

The following is a weaker heuristic than the assumption 2.

Assumption 3. Let $\mu,\nu \in V_T$ have a common owner and $\mu,\nu \notin V_G$. Let V_μ, V_ν be the domains of μ and ν, respectively. If a path from V_μ to V_ν exists, which starts at an exit customs $\delta_s \in V_\mu$ and ends in an entrance customs $\delta_e \in V_\nu$, then there exits an edge $l \in \boldsymbol{M}(\mu,\nu)$, where $l = (\delta_s, \delta_e)$, the length of which is not greater than the length of the path.

So assumption 3 means the same as assumption 2 for all cases except the one when at least one of μ and ν represents a terminal node of *T*.

Remark 1**.** The purpose of introducing assumption 3 is to separate the navigation inside a lowest level map from the global navigation. It allows a concrete application to involve its own specific for local navigation.

The following are different examples in support of this view:

1. In a city organized as a system of orthogonal streets and avenues, it is quite clear geometrically how to reach the corner of 5th Street and 13th Avenue departing from the corner of 13th Street and 45th Avenue;
2. For a visually guided robot, it is quite natural to assign its *local* navigational tasks just in terms of what the robot is seeing right now. In this case, such a task represents not a graph problem, but a typical problem of Visual Servoing. (It will be discussed further on in the last section).
3. If a *GT*-model represents environment for a blind person, it is natural to use the graph-based interpretation for all level of navigation, both for the “global” tasks as well as for the “local” ones.

Assumptions 1-3 introduce *a priori* knowledge about paths in *G*. Another interpretation of Assumptions 1-3 is that they are mandatory ways restricting the freedom of search, even if a shorter path exits. Anyhow, they are intended to reduce search of shortest paths.

RESOLUTION OF ROUTING PROBLEM UNDER ASSUMPTIONS 1-3

Since under Assumption 2, l is a solution for the *espp* $\{\delta_s, \delta_e\}$ of Eq. (2), the equation can be re-written in the same notation as

$$|\{V_\alpha, V_\beta\}| = \min_{l \in \boldsymbol{M}(\mu,\nu)} (|\{V_\alpha, \delta_s\}| + |l| + |\{\delta_e, V_\beta\}|), \qquad (3)$$

where while minimizing the right-hand side, $\mu,\nu \in V_T$ scan all pairs of brother nodes built by α, β.

This reminds the Bellman's equation. Unlike the Bellman's equation, which by a problem of its left-hand side spawns a new problem on the right-hand side, Eq. (3) spawns two new problems.

This leads to the recursive procedure of Fig. 1 that builds a solution for *espp*. This procedure converges due to proposition 11. Fig. 2 gives an illustration to the procedure.

Two crucial instructions of the pseudocode of Fig. 1 require special comments.

Namely, we must explain how the instruction

"**if** ($V_\alpha \supset V_\beta$ or $V_\alpha \subset V_\beta$)"

can be reduced to known graph problems, and how to construct the pairs $\mu,\nu \in V_T$ of all brother nodes built by given α,β.

Remark 2. Note that both questions are quite simple if it is known in advance that the graph T is a tree. Indeed, in such a case, it is sufficient to track backward two paths that connect the source-node of T with α and β, respectively. If one of the paths contains the other one, then $\{V_\alpha \supset V_\beta$ *or* $V_\alpha \subset V_\beta\}$ is true. Otherwise, the first two non-equal nodes in these paths (counting from the source-node) yield the only possible pair of brother nodes built by α and β. However, as it is explained in the last section, the general (non-tree) case of T is important for practice.

In the above terms, "$V_\alpha \supset V_\beta$" is equivalent to "β is a node of graph T_α determined by α as in definition 6". So "$V_\alpha \supset V_\beta$" holds if and only if a path from α to β exists.

Let S_α denote the set of nodes of T that belong to the paths ending in α.

Using this concept, the instruction *"if ($V_\alpha \supset V_\beta$ or $V_\alpha \subset V_\beta$)"* can be substituted by the equivalent instruction "*if ($\beta \in S_\alpha$ or $\alpha \in S_\beta$)*". The last instruction is simpler because it implies solutions of a well-known graph problem.

```
procedure solve_espp(α, β, path, weight)
begin
if (V_α ⊃ V_β or V_α ⊂ V_β)        //**  This instruction is commented in the main text
                                    //  Here and below "**" means that the instruction
                                    //  curries out an interaction with the database that stores GT-model
then  begin path=∅; weight=0; return; end
else   begin weight=∞;
       for  all μ,ν∈ V_T to be the brother nodes built by α,β do      //** This instruction is commented in the main text
          begin
             if | M(μ,ν)|≠0 then begin                                          //**
                                  for each l∈M(μ,ν), where l=(δ_s δ_e) do       //**
                                     begin solve_espp(α, δ_s, path1, weight1); solve_espp(δ_e, β , path2, weight2);
                  m= weight1+| l |+ weight2; if m<weight then begin weight=m; path=path1∪ l ∪ path2; end
                                     end
                                  end
          end;
       return;
      end
end
```

Figure 1. A procedure that finds a solution for *espp* under Assumption 2. The pseudocode uses general notation of the paper. Input of the procedure is formed by α,β∈ V_T. The output contains a shortest path *path* and its length *weight*. If *weight*=∞, *path* is indefinite. The trivial solution is given by *path*=∅ and *weight*=0.

Indeed, the problem of construction of S_α can be considered as a partial case of well-known *minimum spanning tree* problem (see a review in [Leeuwen, 1990]). In our case, S_α can be constructed by backtracking the paths that end in α.

Note that for most of practical cases, nodes of T can have many children but only a few owners. So the complexity of construction of S_α is expected to be quite low in practice. For example, if T is a tree, S_α contains just nodes of a single path.

The construction of all brother nodes μ,ν can be reduced as well to the same kind of graph problems.

Indeed, in accordance with the database representation of *GT*-model, the information about all brother nodes μ,ν that can be built by α,β, is attached to the source-node γ of minimal sub-graph T_γ built by α and β. So for our purpose, it is sufficient to find γ by α,β.

It is clear that $\gamma \in S_\alpha \cap S_\beta$. On the other hand, it is true $S_\gamma \subset S_\alpha \cap S_\beta$. In addition, γ generates the greatest possible S_γ in the following meaning: If for some $\lambda \in S_\alpha \cap S_\beta$, it is true that $S_\lambda \subset S_\alpha \cap S_\beta$, then $S_\lambda \subset S_\gamma$.

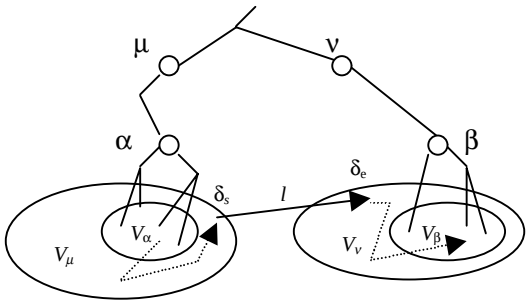

Figure 2. General step of the algorithm of Fig.1: *Find brother nodes* μ,ν *of* α, β *and chose* $l \in \boldsymbol{M}(\mu,\nu)$, *where* $l=(\delta_s\ \delta_e)$. (The paths of broken dotted lines must be found by recursive application of the general step for two new *espp*s.)

This observation suggests a clear scheme of search of γ:

- Construct $S_\alpha \cap S_\beta$
- For each node $\lambda \in S_\alpha \cap S_\beta$, construct S_λ
- If $S_\lambda \not\subset S_\alpha \cap S_\beta$ then reject λ
- Amongst all the candidates remaining after the rejection, select γ as the node that generates the greatest possible S_γ.

For example, if T is a tree, then $S_\alpha \cap S_\beta$ just means the common part of the only two possible paths from the source-node to α and β, respectively, whereas λ is the lowest node in this common part.

For the case of Assumption 3, a simple modification of the procedure of Fig.1 can follow it up to detection of some μ or ν that belong to V_G. Then this modified procedure must call a specific procedure that resolves lowest-level navigational problem (see remark 1), for example, as a classic *spp* in a sub-graph G' of G determined by the terminal branch of T spawned by the owner of μ or $\nu \in V_G$. (The details are omitted.).

RESOLUTION OF PLANNING-A-TRIP PROBLEM

For routing problem, the robot can start moving along any road any time. Unlike that, the planning-a-trip problem is not a stationary problem: after arriving at a node, the next segment of the trip depends on the timetables.

However, a new interpretation of the components of *GT*-model makes the above approach as well valid for the planning-a-trip problem.

In this new interpretation, graph T means the same as before. The nodes of graph G, although they do not refer to any view now, have a very close interpretation to the old one: each node of new G stands for an atomic state of the traveler in the world. Unlike that, the difference between the new and the old interpretations of edges of G is essential.

Any edge $l \in E_G$, where $l = (a, b)$ and $a, b \in V_G$, means now a joint timetable of all available kinds of transport that connect a with b.

This timetable can be represented as function $l(t,tr)$ of two arguments, where t and tr means, respectively, the time of arrival in a and the identifier of a transportation line (flight, bus, ship, etc.) that connects a with b. Given t and tr, the value $l(t,tr)$ represents the time of arrival in b.

The *trip* from node *a* to node *b* is defined as a sequence

$\{l_0,tr_0\},\{l_1,tr_1\},...,\{l_n,tr_n\}$,

where $l_0,l_1,,...,l_n$ is a path in graph *G* from *a* to *b*, and each tr_k, k=0, 1,..., *n*, represents identifier of a transportation line that connects the start-node of l_k with its end-node.

The schedule of a trip depends on the initial time t_0. Recursively, the time of arrival in each next node is given by $t_k=l_{k-1}(t_{k-1}, tr_{k-1})$, k=1,...,*n*.

Then, any segment of a trip can be optimized by choosing the fastest way to achieve the end-node of the corresponding edge. For this purpose, $|l_k(t)|$ will denote $\min_{tr}[l_k(t,tr)-t]$, where *t* is the time of arrival in the start node of l_k. In other words, the function $|l_k(t)|$ yields *the minimal duration of k-th link of a trip*.

---

procedure *solve_ptp*(*t*, α, β, *trip*, *duration*)
begin
if ($V_\alpha \supset V_\beta$ or $V_\alpha \subset V_\beta$)
then begin *trip*=∅; *duration*=0; **return; end**
else begin *duration*=∞;
for all μ,ν∈ V_T to be the brother nodes built by α,β **do**
begin
if $|M(\mu,\nu)|\neq 0$ **then begin**
for each $l \in M(\mu,\nu)$, where $l=(\delta_s\ \delta_e)$ **do**
begin *solve_ptp*(*t*, α, δ_s, *trip*1, *duration*1); *t*1= *t* + *duration*1;
trans= arg $\min_{tr}$ |*l*(*t*1,*tr*)-*t*1|; *t*2= *l* (*t*1, *trans*); *solve_ptp*(*t*2, δ_e, β , *trip*2, *duration*2); m= *duration*1+| *l(t1)*|+ *duration*2;
if m<*duration* **then begin** *duration*=m; *trip*=*trip*1∪{ *l* , *trans*} ∪ *trip*2; **end**
end
end
end;
return;
end
end

Figure 3. A procedure that finds a solution for planning-a-trip problem under Assumption 2. The procedure input is formed by the initial time of the trip *t* and α,β∈ V_T. The output contains a fastest trip *trip* and its *duration*. If *duration*=∞, *trip* is indefinite. The trivial solution is given by *trip*=∅ and *duration*=0. The comments to instructions of the procedure are omitted because they are identical to the ones of Fig.1.

In these terms, the planning-a-trip problem is: *Given* $\alpha,\beta \in V_T$, *find a trip of the minimal summary duration*

$|l_0(t_0)|+|l_1(t_1)|+\ldots+|l_n(t_n)|$,

which begins inside V_α *and ends inside* V_β.

The transportation line tr_k of k-th link of a fastest trip is supposed to be chosen as

$tr_k = \arg\min_{tr}[l(t_k,tr)-t_k]$.

Under the same assumptions 1-2 as before, this new interpretation leads to a modification of the algorithm of Fig. 1 for the planning-a-trip problem. This modification is presented on Fig. 3. The above comments on the principal instructions of the procedure of Fig.1 are true as well for the algorithm of Fig.3.

COMPLEXITY

Let X be the complexity of algorithm of Fig.1. This section presents an estimate of X for the case when graph T is a tree.

Let $f<\sim g$ denote the fact that function f is less than a function linear in function g. I.e., that $f<Cg$ holds for some constant C and all values of the arguments of f and g.

In this notation, $X<\sim NY$ holds, where N and Y means, respectively, the number of recursive calls of the procedure and the complexity of the body of each copy of the procedure spawned in the recursive process.

Denoting by C_1 the maximal number of nodes in paths of T, and $C_2=max|M(\mu,\nu)|$, one has $N\leq (2C_2)^{C_1-2}$.

To estimate Y, let us estimate the complexity of the principal operations of the body.

Let P be the complexity of retrieval of a record from the database that stores the *GT*-model. All lines of Fig. 1 marked with "**" assume such retrievals. P can be accepted as linear in the logarithm of size of *GT*-model.

Let us assume that this database grows downwards uniformly in the sense that the number of children for all nodes is approximately the same, and that for all brother nodes μ,ν, it is true $C_2=|M(\mu,\nu)|$. In this case, logarithm of the database size grows as a function linear in C_1.

Since T is assumed to be a tree, then, as it was mentioned above, the instruction

if ($V_\alpha \supset V_\beta$ or $V_\alpha \subset V_\beta$)

just means the backtracking of two paths from α and β, respectively, to the source-node of T. Since each path contains $\leq C_1$ nodes, the complexity of this instruction is bounded by a function linear in $2C_1P$. That is, it is by a function linear in $2(C_1)^2$.

The same relation $<\sim 2(C_1)^2$ holds for the complexity of construction of a single possible pair of brother nodes $\mu,\nu \in V_T$ built by α,β.

For each pair (μ,ν), the complexity of retrieval of all $l \in M(\mu,\nu)$ is linear in C_2P. That is, it is linear in C_2C_1.

Hence, $Y <\sim [2(C_1)^2 + 2(C_1)^2 + C_1\, C_2]$.

Finally, we have come to

$$X <\sim Y\, N <\sim [2(C_1)^2 + 2(C_1)^2 + C_1\, C_2]\ (2C_2)^{C_1-2},$$

i.e. $$X <\sim (4\, C_1 + C_2) C_1 (2C_2)^{C_1-2}.$$

Note that this estimate does not depend on the sizes $|V_\alpha|$ and $|V_\beta|$ of arguments of *espp*. Informally, it means that *the required time for planning a best route between, for example, two buildings all over the world is bounded by the same limit as between a street and a country*. This leads to an essential gain in complexity with respect to the straightforward approach to *espp* mentioned in Introduction.

A similar estimate holds for the planning-a-trip algorithm of Fig.3. In this case, we may assume that for a model that represents geography of the real world, C_1=6. For instance, the six nodes of a path in *T* can mean, respectively: *world, country, state, region, town,* and *street*.

If maximal number C_2 of the edges that connect the domains of two brother nodes of *T* is limited, for example, by five, we obtain

$(4\ C_1+C_2)C_1(2C_2)^{C_1-2}=(24+5)\times6\times10^4<10^6$.

Although the assumptions used for this estimate are rather rough, this analysis shows that a virtual project of a system able to find in real time a fastest trip between any pair of geographical objects of the world is quite realistic.

DISCUSSION: CREATION OF *GT*-MODELS

Informally, the practical result of this paper is *that if all states of a moving agent inside its environment are represented by a GT-model, then there exits a universal and efficient algorithm that finds an optimal solution for any navigational problem posed in terms of this GT-model, avoiding exploration of the environment by the agent*.

So, keeping in mind the same two applications as above, a natural question is how to create a *GT*-model for either of them.

Creation of a *GT*-model for the planning-a-trip problem is rather a matter of development than of research. The goal of such a development is representation of all available timetables in the form of a dedicated database that satisfies the specifications presented above. For this application, graph *T* should represent a real geographic system.

Development of such a database involves many specific aspects as standards, protocols, actualization, etc. Nevertheless, there is no principal obstacle for such a development.

In contrast with that, learning *GT*-model for a visually guided robot contains a large field of work for researchers. In fact, for robotics applications, *GT*-approach cannot be used literally and needs some modifications.

The point is that, since the nodes of graph *G* are interpreted as classes of similar *views*, it is not clear how to store and recognize them in practice. Indeed, for a real environment with a huge number of views, it is unrealistic to interpret *views* as crude digital pictures because they require much of memory and are unstable due to change of light conditions and other factors.

Note that, for a laboratory experiment when the number of nodes is respectively small, learning *G* is quite possible. In particular, successful experiments of learning view graphs are presented in [Shölkopf and Mallot, 1995] and in [Franz et al., 1998].

Since the set of nodes of *G* coincides with the set of terminal nodes of *T*, the huge size of *G* represents as well an obstacle for learning *T*.

Nevertheless, this and some other obstacles can be eliminated modifying *GT*-model. The goal of the rest of this section is not to present such a modified *GT*-model, but rather to show that it is possible.

At the same time, we want to present some specific research lines motivated by the above approach.

One of these lines is represented by the problem of *merging GT-models*. Informally, this problem is similar to the following question: let some *GT*-models of *table* and of *room* be given. How can they be used for the creation of a *GT*-model that would represent *room containing table*?

Below we consider some aspects of learning *T* and *G*.

Learning Graph *T*

Due to the fact that the graph *T* simulates some fundamental properties of human geographic mentality, *a simple language can be developed for supervised learning T*.

Indeed, we can assume that such a language involves instructions of the two following kinds:

- Presentation in any order of geographic identifiers (like *USA, Paris, Broadway,...*)
- Presentation of inclusions (like *USA⊃Broadway*).

It is possible to develop a translator able to build the nodes and edges of *T* interpreting instructions of these kinds.

For example, after instruction *USA⊃Broadway*, the translator must introduce an edge between nodes representing *USA* and *Broadway*, respectively. However, after instructions *USA* ⊃ *New_York* ⊃ *Broadway*, this edge must be deleted and replaced by two new edges, the one between *USA* and *New_York*, and another one between *New_York* and *Broadway*.

Remark 3. As far as graph *T* belongs to a wider class than trees, the *GT*-approach gives tutor much of freedom for expressing inclusions. In particular, it allows introducing both *Moscow⊂Russia* and *Moscow ⊂Europe*. For *T* given as a plain tree, these two instructions are incompatible because *Russia* ⊄ *Europe* and *Europe* ⊄ *Russia*.

However, if *T* is aimed to represent geography for navigation in real scenes, it is unrealistic to introduce all its nodes and edges by supervisor.

For example, *dining room* represents an explicit name of a geographical object of a natural environment. It can be used for localization of robot in scene or as a goal of a navigational task. However in real life, we use also smaller geographical objects for navigation, although they have no proper name.

To some extent, natural language allows us to determine them by sentences. For example, a small geographic object can be defined as *"the set of robot states inside the dining room specified by the condition that the table is on the robot's left side and, at the same time, the robot can see a cherry tree outside the window"*.

In real life, we chose automatically some landmarks to determine such sets of states in a scene.

These arguments show that the lower levels of graph *T* should be constructed automatically rather than presented by a tutor.

Furthermore, note that for a visually guided robot, *instead of storing all nodes of a lowest level map of T, it is possible to store only one its representative.* It leads to a considerable gain of the required memory and the learning time.

This opportunity is closely related to p. 2 of remark 1. Indeed, the assumption that any local navigational task is expressed only in terms of visible landmarks implies that the paradigm of Visual Servoing [Hashimoto, 1993] can be used for robot local navigation. That is: *Given a set of landmarks in an input view, control the robot by visual feedback to achieve a view with a specified state of these landmarks*.

If a navigational task is understood as a problem of Visual Servoing, there is neither need to interpret it as a graph problem (which was mentioned in remark 1), nor the necessity of storing in memory all possible intermediate views inside a low level map.

These observations suggest a clear idea of how to modify *GT*-model to make realistic the volume of required memory. Note that it leads to a significant gain of the needed memory due to the dimension factor: The nodes of a lowest level map represent a sampling of a six-dimensional space (we mean the space of states of a rigid body in R^3).

One more way to reduce the plain memorizing of nodes of *T* is to allow the robot a limited exploration of its environment. I.e. the exploration can be allowed if the minimal sub-graph of *T* built by the goal and the initial nodes, which determine a navigational task, is respectively small. Future research must determine to what extent small.

Remark 4. In principle, it is possible to develop a scheme of unsupervised learning *T*. However supervised learning of real geographical concepts has a specific purpose: *If the nodes of T stand for some real geographic objects, then the natural navigational problems can be simply translated into the problems posed in terms of a corresponding GT-model and planned automatically*. For example, considering again the task "*go to the dining room*", if we wish a *GT*-model of the real world to fit this task, then a node of *T* must represent *the dining room*.

Learning Graph *G*

An application of *GT*-model could be relatively free of the interpretation of *G*, meanwhile learning *G* depends on that.

For example, the peculiarities of the planning-a-trip problem transform the process of learning *G* into respectively simple writing the timetables into a database.

Unlike that, for robot motion planning, any interpretation of *G* changes learning *G*.

Note that in addition to the interpretation presented in the Introduction (see [Khachaturov, 1999A] for more details of this interpretation), graph *G* of a *GT*-model may be substituted with view graph [Mallot and Shölkopf, 1995] developed furthermore in [Franz et al., 1998], or even with Canny's road map [Canny, 1987].

A brief comparison of these three interpretations follows.

"The edges of the view graph indicate temporal coherence: Two views are connected if and only if they can be experienced in immediate temporal sequence" [Shölkopf and Mallot, 1995]. Unlike that, an edge of *G* presented in the Introduction connects two "relevant" views by a control rule. While moving due to such a rule, the robot can meet and use in its control loop many "irrelevant" views, i.e. those views which should not be stored in the model.

Both approaches avoid working with any explicit model of robot configuration space. They introduce *G* as cognitive maps with the purpose to make *G* useful not only for planning, but for physical execution of paths in *G* as well.

Unlike that, the edges of Canny's road map refer to one-dimensional objects of robot configuration space. So, if a path in Canny's map is found, its execution by a concrete robot control system represents an independent problem: the Canny's map does not deal with any data which would link a concrete robot sensor (e.g. vision) with a path to be executed.

However for the first two interpretations, the inclusion of *G* into the process of low-level robot control is rather a goal than a real fact, especially for the case of huge volume of visual data for a real robot environment. It is the point of future research to change this situation.

To achieve this goal, in particular, it makes sense to codify each view by parameters of some landmarks. We will touch again this point in the last item of this section.

Note that no essential property of vision is involved yet in the above approach. Indeed, for all interpretations of *G*, the "views" are related to the sensor input in general. It can be poor (as for a blind man), or rich. However, the robot with poor sensor input can navigate successfully without exploration (as a blind man), provided that its *GT*-model is adequate. So, what are those essential properties of vision contrasting with a sensor in general? How can they be described in terms of *G* and *T*? What is their role for learning?

In this respect, the ability of vision to perform a remote reachability test seems very promising for learning *GT*-model. For example, this ability can be expressed in the following way [Zhukov et al., 1998]: *Seeing a pair of remote objects, say A and B, we can detect sometimes that the object A can be reached moving from B. Moreover, we can track visually a shortest path between B and A which could be tracked physically provided that our initial state is at B.*

So the point of future research is how to provide a robot with a similar ability combined with the ability of a correct identification of *A* and *B* with some nodes of graph *T*. If robot can do that, then filling in the database of *GT*-model could be simplified significantly: most of the data would be written after remote tracking paths instead of tracking them physically.

Database and Graph Theory Aspects

Development of a system based on the presented approach involves some specific problems related to databases, graphs, and complexity. Two of them follow.

- The algorithms of Fig. 1 and Fig. 3 are based on the database representation of *GT*-model. In particular, it implicitly assumes that each newly learnt edge of *G* would be stored in accordance with those rules that were introduced above for this database. A system support of this assumption meets no serious obstacles unless database contains some errors (like *USA*⊂*Russia*). However in general case, the development of an error tolerant system requires additional research.

- As it was shown above, the complexity of search is respectively low as long as graph *T* is "good" in a certain sense. However, in general case a system of real geographic concepts can lead to a “bad” for complexity graph *T*. Future work should study this contradiction in details.

Perception

Trying to introduce a discipline into a vast variety of possible architectures for the processing of input images, we find that *GT*-approach can be useful for that.

We may understand the views of a visually guided robot in a wide sense, i.e. as the images of any kind of remote sensor. In particular, the sensor can be a laser system producing range images, a binocular system, a gray-scale TV-camera with variable focus, etc.

The compression of an input image into a compact description of parameters of current landmarks depends strongly on the kind of sensor.

Similarly, the set of parameters relevant for navigation, that must be extracted by perception, is not a fixed set: As the number of parameters as well as their semantics are free to large extent for variation. For a fixed robot pose in a scene, the relevancy of each parameter depends on the current operational goal of the robot.

The important feature of *GT*-approach is that it represents the properties of perception, which are essential for navigation, in the form of specifications on the perception system. It provides a distinct interface between what-we-need from perception for a good navigation and how-to-achieve that.

For example, the above-mentioned ability of remote reachability testing can be easily expressed in terms of *GT*-approach in axiomatic form, as something given. Then we can study what this property contributes for learning *GT*-model. Independently, we can study how to develop a perception system that provides this ability.

There is another peculiarity of *GT*-approach that can be used to simplify perception.

Note that there are only two system tasks that perception should perform for tracking physically a path found by a *GT*-model: (i) auto-localization of the current robot state as a node of *T*, (ii) providing a feedback for the robot control loop like in Visual Servoing approach.

Combining properly these two tasks with the *GT*-approach, it is possible to use "perception by adjustment", which is much simpler than "perception in general". The following clarifies this idea.

Let us assume that the robot knows its actual state as a node of *T*. The next robot state and, consequently, its next view are not arbitrary: They depend on the actual state and the current action that robot will do.

It is similar to the following example "*I am in the hall. Passing this door, I do enter the kitchen. When I will have done that, I see a table on my left.*" For this example, the perception by adjustment means that entering the *kitchen*, the robot already knows that what he sees on his left must contain a *table*. So the point is to identify exact positions of all details of the table in the view. Of course, the required

processing (perception by adjustment) is much simpler than analysis of the same view, but without any historical and spatial context (perception in general).

For the same example, *hall* and *kitchen* correspond to certain nodes of *T*. While the robot moves from *hall* to *kitchen* due to a correct plan, the matter of current localization is only to detect the moment when *hall* is no longer valid. After that, the robot knows that he is already in *kitchen*. This makes localization quite simple.

Although some of these ideas might seem to be quite straightforward, unlike other approaches, the *GT*-approach presents a mean for their practical integration.

On the other hand, it is clear that these ideas are not a solution, but just a challenge to start research that should find it.

References

Canny, J. 1987. *Complexity of robot motion planning*, Ph.D. Dissertation, Comp. Science Dept., M.I.T., Cambridge, MA.

Dijkstra, E.W. 1959. A note on two problems in connection with graphs, *Numer. Math.* **1**, 169-271.

Franz, M.O., B. Schölkopf, H.A. Mallot, and H.H. Bülthoff. 1998. Learning view graphs for robot navigation, *Autonomous Robots* **5**, 111- 125, Kluwer.

Fredman, M.I. and R.E. Tarjan. 1987. Fibbonacci heaps and their use in improved network optimization problems, *J. ACM* **34**, 596-615.

Hashimoto K. 1993. *Visual Servoing*, World Scientific.

Khachaturov, G. 1999A. An approach to the non-Cartesian navigation, in *Proc. of SIARP'99* - 4th Iberoamerican Simposium on Pattern Recognition, 67-78, Havana, Cuba.

Khachaturov, G. 1999B. Robot Navigation with an Almost Natural Interface, in *Proc. of 10th Irish Conference on Artificial Intelligence & Cognitive Science (AICS99)*, 116-122, University College Cork, Ireland.

Leeuwen, J. van. 1990. Graph algorithms, in *Handbook of Theoretical Computer Science*, v.A. Algorithms and Complexity, 525-631, Elsevier.

Sharir, M. 1997. Motion planning, in *Handbook of Discrete and Computational Geometry*, (J. E. Goodman and J. O'Rourke, Eds.), 733—754, CRC Press, Boca Raton, Florida,

Shölkopf, B. and H.P. Mallot. 1995. View-based cognition mapping and path planning, *Adaptive Behavior*, v.3, No. 3, 311-348, M.I.T.

Zhukov,S., Iones,A., Kronin,G. 1998. Navigation of intelligent characters in complex 3D synthetic environment in real-time applications, in *Proc. of WSCG'98 - Central European conference on Computer Graphics and Visualization'98*, 456-463, Czech Rep., Plzen.