Autonomous Robotic Vehicles

Hellenic Mediterranean University

Lecture 14

Dr. Alina Eqtami

Lecture Goals - Στόχοι σημερινού μαθήματος!

Μετά το σημερινό μάθημα θα μπορείτε:

- να εξηγείτε τη λογική των **Potential Fields** και τις βασικές τους συνιστώσες.
- να κατανοείτε πώς λειτουργούν βασικοί **obstacle avoidance** αλγόριθμοι (Bug, VFH, DWA, CVM).
- να αναγνωρίζετε τα πλεονεκτήματα/όρια κάθε μεθόδου σε πραγματικά περιβάλλοντα.
- να περιγράφετε τις **Navigation Architectures** και τον ρόλο τους.
- να αντιληφθείτε τη συνεργασία **global planner – local planner – controller**.

Potential Field Path Planning: Βασική Ιδέα

- Η μέθοδος Potential Field Path Planning δημιουργεί ένα **τεχνητό πεδίο δυναμικού** πάνω στον χάρτη του ρομπότ.
- Το ρομπότ θεωρείται **σημείο στο C-space** και κινείται όπως μια μπάλα που κυλά στην «κατηφόρα» στο πεδίο.
- Το goal μοντελοποιείται ως **ελκτική περιοχή** (attractive potential).
- Τα εμπόδια λειτουργούν ως **απωστικές περιοχές** (repulsive potential) που το κρατούν σε ασφαλή απόσταση.

Συνολικό Δυναμικό

$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

Το ρομπότ ακολουθεί το **αρνητικό gradient** του U:

$$F(q) = -\nabla U(q)$$

Το πεδίο δυναμικού λειτουργεί όχι μόνο ως **μέθοδος path planning**, αλλά και ως **κανόνας ελέγχου** (control law) σε πραγματικό χρόνο.

Potential Field Example

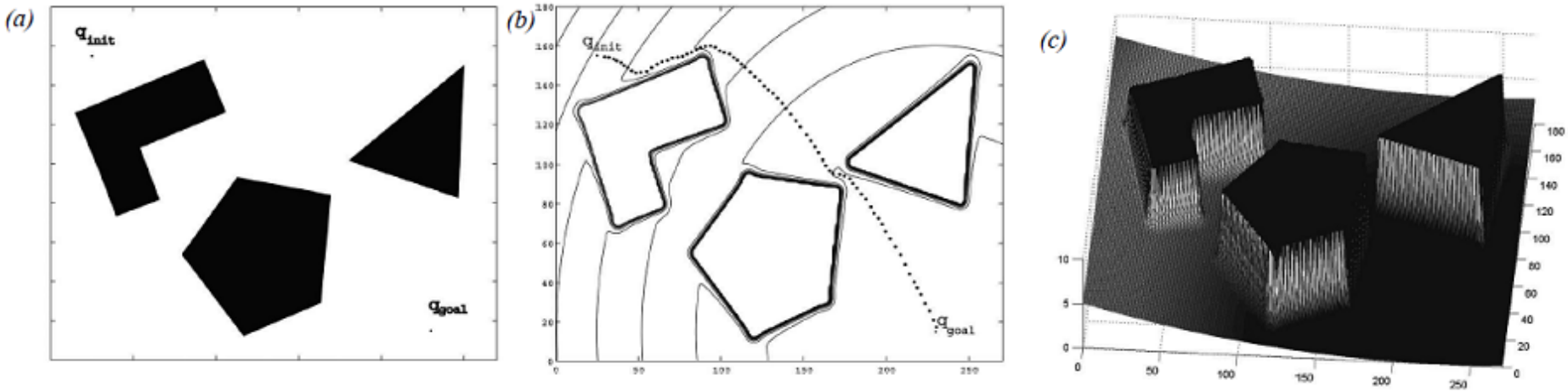


Figure: Typical potential field generated by the attracting goal and two obstacles.
(a) Configuration of the obstacles, start (top left) and goal (bottom right).
(b) Equipotential plot and path generated by the field. (Σαν υψομετρικοί χάρτες σε τοπογραφικό διάγραμμα)
(c) Resulting potential field generated by the goal attractor and obstacles.

Σχέση Ισουψών Καμπύλων και Γραδιεντ

Οι ισουψείς καμπύλες (*ισοδυναμικές*) έχουν την ιδιότητα ότι το ∇U είναι πάντα κάθετο σε αυτές. Επομένως η τεχνητή δύναμη $-\nabla U$ είναι επίσης κάθετη στις ισουψείς. Έτσι, το ρομπότ ακολουθεί κατευθύνσεις που τέμνουν κάθετα τις ισουψείς καμπύλες, «κατηφορίζοντας» από περιοχές υψηλότερου δυναμικού προς χαμηλότερο.

Gradient: Τι είναι η Κλίση

- Το gradient (κλίση) δείχνει **την κατεύθυνση μέγιστης αύξησης** μιας συνάρτησης.
- Για μια συνάρτηση δυναμικού $U(x, y)$, το gradient δίνεται από:

$$\nabla U = \left[\frac{\partial U}{\partial x}, \frac{\partial U}{\partial y} \right]$$

- Είναι ένα διάνυσμα που δείχνει «προς τα πάνω» στο πεδίο δυναμικού.
- Το ρομπότ κινείται προς την **αντίθετη κατεύθυνση**:

$$F(q) = -\nabla U(q)$$

Attractive Potential: Παράδειγμα για Unicycle

- Το unicycle έχει θέση $q = (x, y)$ και θέλουμε να κινηθεί προς το goal $q_g = (x_g, y_g)$.
- Ορίζουμε ένα απλό **ελκτικό δυναμικό**:

$$U_{\text{att}}(x, y) = \frac{1}{2} k \left[(x - x_g)^2 + (y - y_g)^2 \right]$$

- Το gradient του δυναμικού:

$$\nabla U_{\text{att}} = [k(x - x_g),\ k(y - y_g)]$$

- Η τεχνητή δύναμη που οδηγεί το ρομπότ στο goal:

$$F = -\nabla U_{\text{att}} = [-k(x - x_g),\ -k(y - y_g)]$$

Attractive Potential: Ελκτικό Δυναμικό προς το Goal

- Το attractive potential οδηγεί το ρομπότ προς το **goal** q_g, όπου η τιμή του δυναμικού είναι ελάχιστη.
- Αυτό το **παραβολικό δυναμικό**:

$$U_{\text{att}}(x, y) = \frac{1}{2} k \left[(x - x_g)^2 + (y - y_g)^2 \right]$$

- δημιουργεί ένα «μπολ» δυναμικού: όσο πιο μακριά από το goal, τόσο μεγαλύτερη η τιμή του U.
- Το ρομπότ κινείται προς την κατεύθυνση **μέγιστης μείωσης** του U (σαν να «κυλά» προς το κέντρο).
- Η τεχνητή δύναμη υπολογίζεται με το **αρνητικό gradient**:

$$F_{\text{att}} = -\nabla U_{\text{att}} = [-k(x - x_g),\ -k(y - y_g)]$$

Attractive Potential Variants: Εναλλακτικές Μορφές

- Ανάλογα με το ρομπότ και το περιβάλλον, χρησιμοποιούνται διαφορετικές μορφές **attractive potential**.

1. Conical (Linear) Potential

$$U_{\text{att}} = k\sqrt{(x - x_g)^2 + (y - y_g)^2}$$

Λιγότερο απότομο κοντά στο goal, κατάλληλο για ομαλή επιβράδυνση.

2. Saturated Potential

$$U_{\text{att}} = \begin{cases} \frac{1}{2}kd^2, & d \leq d_0 \\ kd_0(d - \frac{1}{2}d_0), & d > d_0 \end{cases}$$

Αποτρέπει πολύ μεγάλες δυνάμεις όταν το ρομπότ είναι μακριά από τον στόχο.

Attractive Potential: Conical (Linear)

Conical / Linear Attractive Potential

$$U_{\text{att}} = k\, d \qquad \text{where } d = \sqrt{(x - x_g)^2 + (y - y_g)^2}$$

Το gradient του είναι σταθερό σε μέτρο:

$$\nabla U_{\text{att}} = k \frac{q - q_g}{d}$$

Άρα η δύναμη:

$$F_{\text{att}} = -k \frac{q - q_g}{d}$$

- Το μέγεθος της δύναμης μένει σχεδόν σταθερό ακόμη και όσο $d \to 0$.
- Το ρομπότ επιβραδύνει ομαλά καθώς πλησιάζει το goal.
- Δεν τραβιέται υπερβολικά δυνατά προς το τέλος.
- Αποφεύγονται ταλαντώσεις ή υπερ-σύγκλιση.
- Πολύ καλό για **motion smoothing**.

Attractive Potential: Parabolic

Parabolic Attractive Potential

$$U_{\text{att}} = \frac{1}{2} k d^2$$

$$\nabla U_{\text{att}} = k(q - q_g) \qquad F_{\text{att}} = -k(q - q_g)$$

- Η δύναμη μειώνεται γραμμικά όσο $d \to 0$.
- Το goal είναι σταθερό ελάχιστο (προσελκύει χωρίς να "τινάζει" το ρομπότ).
- Ιδανικό για σταθερή σύγκλιση στον στόχο.
- Αποτρέπει υπερ-σύγκλιση και ταλαντώσεις κοντά στο goal.
- Χρησιμοποιείται συχνά στην περιοχή γύρω από τον στόχο.
- Note: $k > 0$ είναι ο συντελεστής έντασης (scaling factor).

Repulsive Potential

- Το απωστικό δυναμικό (Repulsive Potential) κρατά το ρομπότ μακριά από τα εμπόδια.
- Πρέπει να είναι **ισχυρό κοντά στο εμπόδιο** και **μηδενικό μακριά από αυτό**.
- Μία συνηθισμένη μορφή είναι:

$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2} k_{\text{rep}} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2, & \rho(q) \leq \rho_0 \\ 0, & \rho(q) \geq \rho_0 \end{cases}$$

όπου:

- $k_{\text{rep}} \rightarrow$ ένταση απωστικής δύναμης,
- $\rho(q) \rightarrow$ ελάχιστη απόσταση από το εμπόδιο,
- $\rho_0 \rightarrow$ **απόσταση επιρροής** του εμποδίου.

Γιατί στο Repulsive Potential εμφανίζεται ο όρος $1/\rho$;

- Το απωστικό δυναμικό πρέπει:
  - να **αυξάνει απότομα** κοντά στο εμπόδιο,
  - να **μηδενίζεται** όταν το ρομπότ είναι μακριά.
- Ο όρος $\frac{1}{\rho(q)}$ ικανοποιεί αυτά τα δύο:
  - $\rho \to 0 \Rightarrow 1/\rho \to \infty \to$ πολύ ισχυρή απωστική δύναμη.
  - $\rho \gg \rho_0 \Rightarrow 1/\rho \approx 0 \to$ καμία επίδραση.
- Η μορφή $U_{\text{rep}} \propto (1/\rho - 1/\rho_0)^2$ είναι **ομαλή** και **διαφορίσιμη**, ιδανική για υπολογισμό της δύναμης:

$$F_{\text{rep}} = -\nabla U_{\text{rep}}$$

Repulsive Force & Total Force Field

Repulsive force προκύπτει από το **αρνητικό** gradient του U_{rep}.

$$F_{\text{rep}}(q) = -\nabla U_{\text{rep}}(q) = k_{\text{rep}} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right) \frac{1}{\rho(q)^2} \nabla \rho(q)$$

- Πολύ ισχυρή κοντά στο εμπόδιο ($\rho \to 0$).
- Μηδενίζεται πλήρως όταν $\rho(q) \geq \rho_0$.
- Κατευθύνει το ρομπότ **μακριά από εμπόδια** με ομαλό τρόπο.

Total Force Field

$$F(q) = F_{\text{att}}(q) - F_{\text{rep}}(q)$$

Ο συνδυασμός αυτός ωθεί το ρομπότ προς το goal ενώ ταυτόχρονα το απομακρύνει από τα εμπόδια.

Limitations: Περιορισμοί της μεθόδου των Potential Fields

- **Τοπικά ελάχιστα (local minima)**
  - Το άθροισμα ελκτικών/απωστικών δυνάμεων μπορεί να μηδενιστεί σε σημείο που **δεν είναι το goal**.
  - Τυπικά εμφανίζεται όταν το εμπόδιο βρίσκεται ανάμεσα στο ρομπότ και στο goal.
- **Κοίλα εμπόδια (concave obstacles)**
  - Περιοχές με πολλαπλές ελάχιστες αποστάσεις οδηγούν σε **ταλαντώσεις**.
  - Το ρομπότ μπορεί να κινείται μπρος–πίσω μεταξύ δύο «τοίχων» χωρίς να προχωρά.
- **Υπερβολική απωστική δύναμη**
  - Πολύ μεγάλα gradients κοντά σε οξεία εμπόδια $\rightarrow$ απότομες αλλαγές κατεύθυνσης.
  - Απαιτείται φιλτράρισμα ή κορεσμός (saturation) στη δύναμη.
- **Μη πληρότητα (incompleteness)**
  - Δεν υπάρχει εγγύηση ότι το ρομπότ θα φτάσει στο goal ακόμη κι αν υπάρχει λύση.

Γιατί δεν αρκεί ο Path Planner?

- Ένας **global path planner** χρησιμοποιεί μόνο τα **γνωστά εμπόδια** του χάρτη.
- Όμως, κατά την εκτέλεση της διαδρομής:
  - οι αισθητήρες μπορεί να διαφωνούν με τον χάρτη,
  - το περιβάλλον μπορεί να είναι **δυναμικό** ή ανακριβές.
- Συνεπώς, το ρομπότ πρέπει να μπορεί να **τροποποιεί τη διαδρομή του σε πραγματικό χρόνο** βάσει των πραγματικών μετρήσεων.
- Αυτή είναι η ικανότητα της **obstacle avoidance**, την οποία εξετάζουμε στη συνέχεια.

Obstacle Avoidance: Τοπική Αποφυγή Εμποδίων

- Η **τοπική αποφυγή εμποδίων** επικεντρώνεται στην αλλαγή της πορείας του ρομπότ **κατά την κίνησή του**, βάσει:
  - των άμεσων (ή πρόσφατων) μετρήσεων των αισθητήρων,
  - της θέσης και κατεύθυνσης του goal.
- Οι αλγόριθμοι obstacle avoidance διαφέρουν ως προς:
  - την εξάρτηση από τον **global map**,
  - την ακρίβεια της εκτίμησης θέσης,
  - το αν λαμβάνουν υπόψη **κινηματική** ή **δυναμική** του ρομπότ.
- Από τους απλούστερους:
  - **Bug Algorithms** – βασίζονται μόνο σε τρέχουσες μετρήσεις και μια εκτίμηση κατεύθυνσης προς το goal.
- Πιο σύνθετοι αλγόριθμοι ακολουθούν, λαμβάνοντας υπόψη:
  - ιστορικό μετρήσεων,
  - γεωμετρία και προσανατολισμό του ρομπότ,
  - μοντέλο κίνησης.

Bug Algorithm: Βασική Ιδέα και Bug1

- Το **Bug algorithm** είναι από τους απλούστερους αλγόριθμους **τοπικής αποφυγής εμποδίων**.
- Η βασική ιδέα:
  - Το ρομπότ κινείται προς το goal μέχρι να χτυπήσει ένα εμπόδιο.
  - Έπειτα **ακολουθεί την περίμετρο** του εμποδίου μέχρι να βρει κατάλληλο σημείο για να συνεχίσει.

Bug1

- Το ρομπότ **πλήρως περιφέρεται γύρω από το εμπόδιο**.
- Καταγράφει το σημείο που έχει τη **μικρότερη απόσταση** από το goal.
- Φεύγει από το εμπόδιο από αυτό το σημείο.

Πλεονέκτημα: Εγγυάται ότι φτάνει το goal (αν είναι εφικτό).
Μειονέκτημα: Πολύ αναποτελεσματικό — μεγάλη απόσταση διαδρομής.

Bug2 & Tangent Bug: Βελτιώσεις του Bug Algorithm

Bug2

- Το ρομπότ ακολουθεί την περίμετρο του εμποδίου όπως στο Bug1, αλλά **φεύγει αμέσως** μόλις μπορεί να συνεχίσει **ευθεία προς το goal**.
- Πολύ πιο **αποδοτικό** σε σχέση με το Bug1.
- Ωστόσο, υπάρχουν χειρότερες περιπτώσεις όπου παραμένει **μη βέλτιστο**.

Tangent Bug

- Χρησιμοποιεί πληροφορία από **range sensing**.
- Δημιουργεί τοπική αναπαράσταση του περιβάλλοντος: **Local Tangent Graph (LTG)**.
- Το ρομπότ βρίσκει **σημεία εφαπτομένων** που το οδηγούν:
  - σε συντομότερες διαδρομές,
  - σε ασφαλή απομάκρυνση από εμπόδια,
  - σε πιο έγκαιρη επάνοδο προς το goal.
- Σε απλά περιβάλλοντα παράγει κοντά σε **βέλτιστες** διαδρομές.

Bug1 & Bug2 Algorithm Illustration

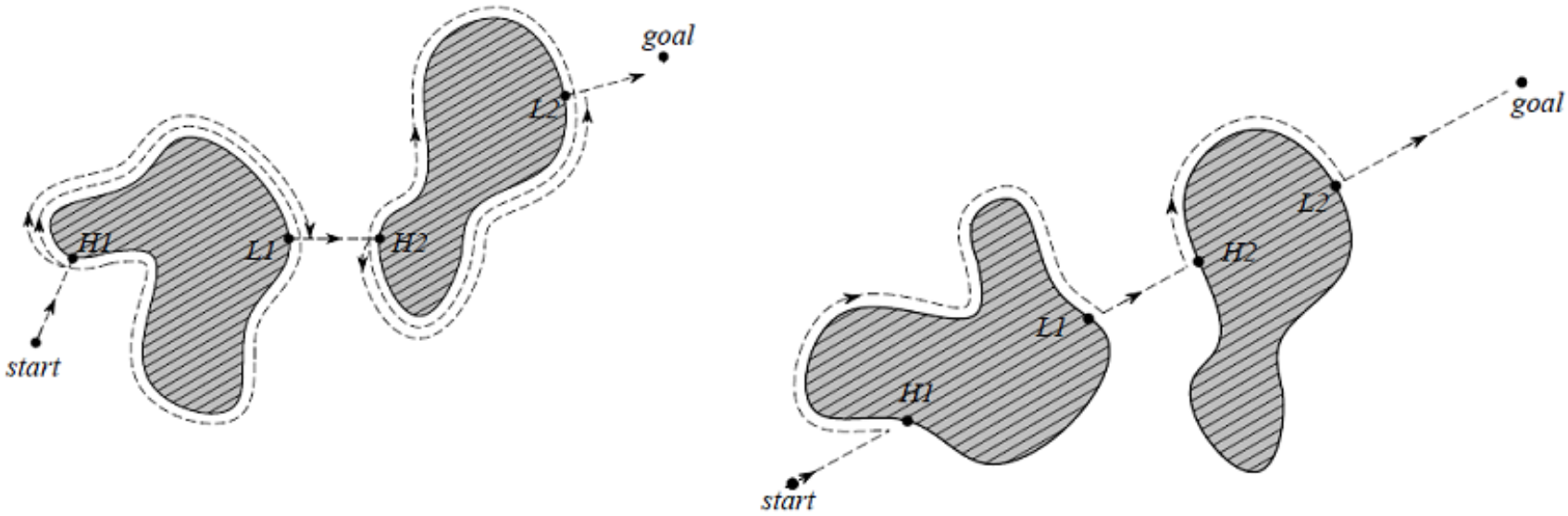


Figure: Left: Bug1 algorithm — the robot fully circumnavigates each obstacle before choosing the closest leave point to the goal. Right: Bug2 algorithm — the robot leaves the obstacle boundary as soon as a direct line toward the goal becomes available.
Source: Siegwart, Nourbakhsh & Scaramuzza, *Introduction to Autonomous Mobile Robots*.

Vector Field Histogram (VFH) — Βασική Ιδέα

- Το VFH αναπτύχθηκε ως βελτίωση των μεθόδων potential fields, που ήταν ασταθείς και αδύναμες σε στενά περάσματα.
- Κύριο πρόβλημα των Bug algorithms:
  - βασίζονται μόνο σε **άμεσα** αισθητήρια δεδομένα,
  - άρα αγνοούν χρήσιμη πρόσφατη πληροφορία.
- Το VFH λύνει αυτό το θέμα δημιουργώντας **τοπικό χάρτη** γύρω από το ρομπότ:
  - μικρό occupancy grid,
  - ενημερωμένο από πρόσφατες μετρήσεις range sensors.
- Από τον τοπικό χάρτη δημιουργείται το **polar histogram** που δείχνει εμπόδια ως συνάρτηση της διεύθυνσης.

Vector Field Histogram (VFH) — Επιλογή Κατεύθυνσης

- Από το polar histogram εντοπίζονται όλες οι **ελεύθερες κατευθύνσεις** για να περάσει το ρομπότ.
- Για κάθε υποψήφια κατεύθυνση υπολογίζεται **συνάρτηση κόστους**:

$$\text{cost} = \alpha \cdot \text{target\_direction} + \beta \cdot \text{wheel\_orientation} + \gamma \cdot \text{previous\_direction}$$

- target_direction: ευθυγράμμιση με το goal.
- wheel_orientation: πόσο πρέπει να γυρίσουν οι τροχοί.
- previous_direction: ομαλότητα μεταξύ διαδοχικών εντολών.

VFH+: Λαμβάνει υπόψη τους **κινηματικούς περιορισμούς** του ρομπότ (όχι όλες οι κατευθύνσεις είναι εφικτές). Τα εμπόδια "μεγεθύνονται" ώστε να αποκλείσουν και όλες τις μη επιτρεπτές τροχιές, παράγοντας το **masked polar histogram**.

Vector Field Histogram — Παράδειγμα

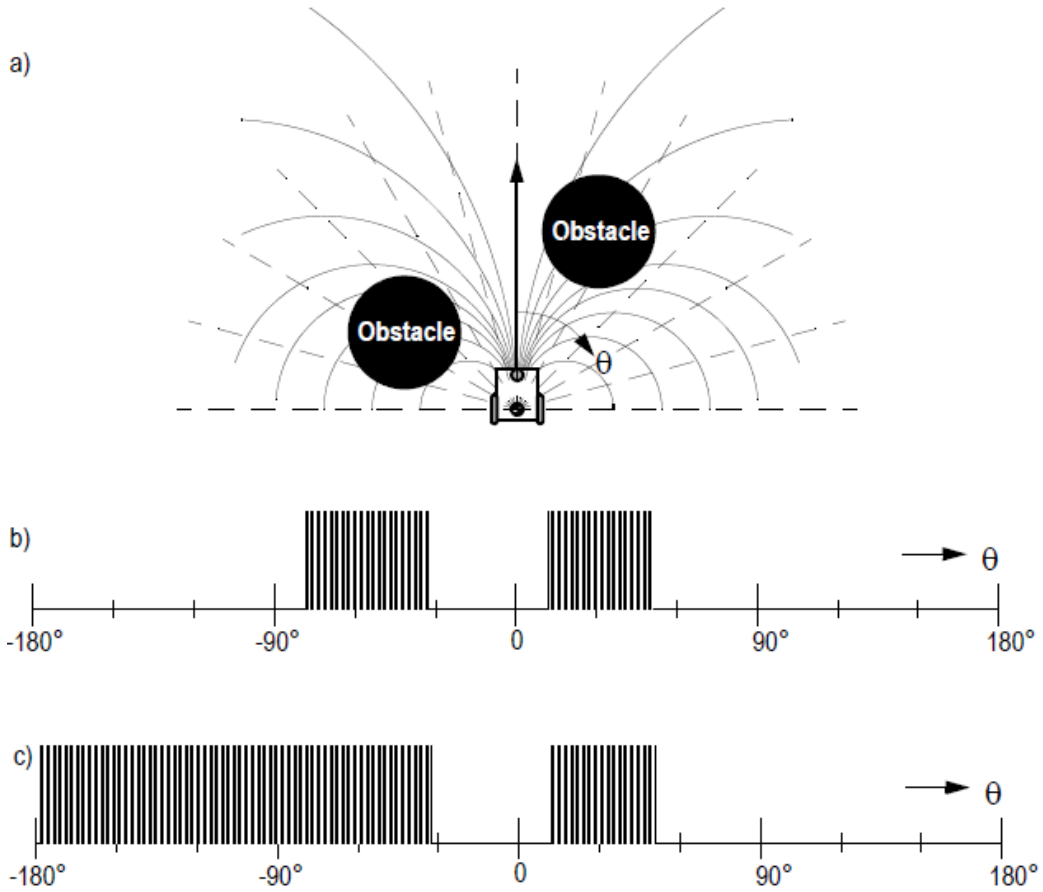


Figure: Example of blocked directions and resulting polar histograms. (a) Robot and blocking obstacles. (b) Polar histogram. (c) Masked polar histogram.
Source: Siegwart, Nourbakhsh & Scaramuzza, *Introduction to Autonomous Mobile Robots*.

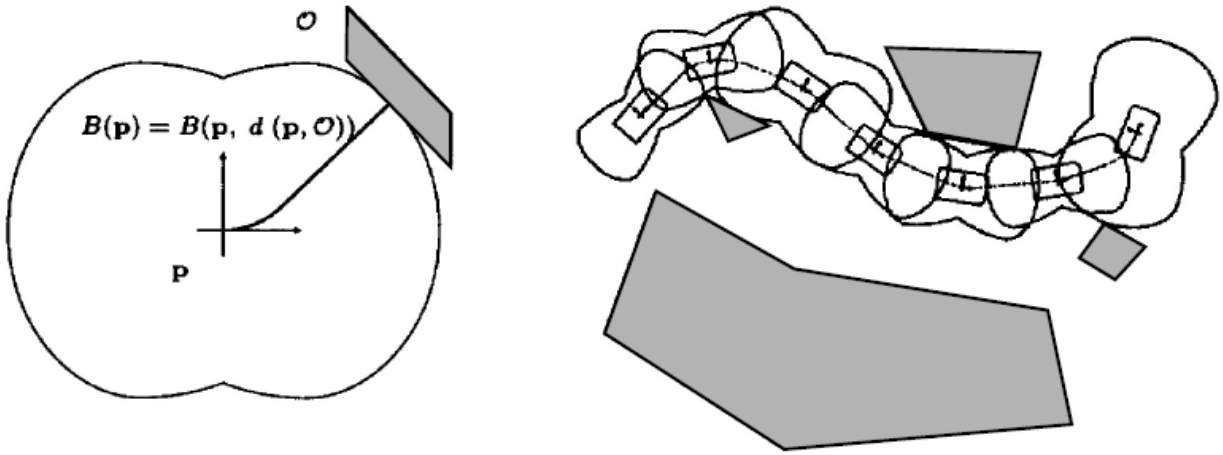
Bubble Band Technique — Βασική Ιδέα

- Η μέθοδος Bubble Band λειτουργεί και για **μη-ολονομικά** (nonholonomic) ρομπότ.
- **Τι είναι ένα bubble**; Ένα bubble είναι η **μέγιστη περιοχή ελεύθερου χώρου** γύρω από μια θέση του ρομπότ μέσα στην οποία το ρομπότ μπορεί να κινηθεί χωρίς σύγκρουση.
- Το μέγεθος του bubble υπολογίζεται:
  - από τον χάρτη εμποδίων,
  - από τα range sensors,
  - και από ένα απλοποιημένο μοντέλο του σχήματος του ρομπότ.
- Τα bubbles αλλάζουν μέγεθος και σχήμα ανάλογα με το τοπικό ελεύθερο χώρο.

Bubble Band — Δημιουργία και Προσαρμογή Τροχιάς

- Αφού βρεθεί η αρχική τροχιά από έναν global path planner, υπολογίζεται μια **αλυσίδα από bubbles** κατά μήκος της διαδρομής.
- Αυτή η «αλυσίδα» δείχνει το **αναμενόμενο ελεύθερο χώρο** του ρομπότ σε όλη την πορεία.
- Ο αλγόριθμος εφαρμόζει:
  - **εξωτερικές δυνάμεις** από τα εμπόδια,
  - **εσωτερικές δυνάμεις** που κρατούν τα bubbles «τεντωμένα» με ελάχιστη ενέργεια,
  - **τελικό smoothing** για ομαλή τροχιά.
- **Ο ρόλος του στην αποφυγή εμποδίων**: Αν το ρομπότ βρεί νέα εμπόδια, το bubble band **παραμορφώνεται δυναμικά** ώστε η τροχιά να αλλάζει ελάχιστα και χωρίς απότομες κινήσεις.
- Πλεονέκτημα: λαμβάνει υπόψη **το πραγματικό σχήμα** του ρομπότ. Μειονέκτημα: απαιτεί **καλό global map** και λειτουργεί καλύτερα σε γνωστά περιβάλλοντα.

Bubble Band — Παραδείγμα



Left: Shape of the bubbles around the vehicle. **Right:** A typical bubble band along a planned trajectory.
Source: Siegwart, Nourbakhsh & Scaramuzza, *Introduction to Autonomous Mobile Robots*.

Curvature Velocity Method (CVM) — Βασική Ιδέα

- Το CVM λαμβάνει υπόψη **κινηματικούς και δυναμικούς περιορισμούς** του ρομπότ κατά την αποφυγή εμποδίων.
- Το ρομπότ υποτίθεται ότι κινείται πάνω σε **τόξα κύκλου**:

$$\kappa = \frac{\omega}{v}$$

- Ορίζεται έτσι το **velocity space**:

$$(v, \omega)$$

όπου κάθε ζεύγος αντιστοιχεί σε μια καμπύλη τροχιάς.

- Το CVM ελέγχει ποια ζεύγη (v, ω) είναι:
  - **επιτρεπτά από την κινηματική**,
  - **εφικτά από δυναμικούς περιορισμούς** (επιτάχυνση, μέγιστη ταχύτητα),
  - **μη συγκρουόμενα** με εμπόδια.

CVM — Ανάλυση σε Χώρο Ταχύτητας

- Τα εμπόδια βρίσκονται αρχικά σε **Cartesian grid**.
- Μετασχηματίζονται στον **velocity space** υπολογίζοντας την απόσταση από το ρομπότ για κάθε πιθανή καμπύλη με καμπυλότητα κ.
- Μόνο καμπυλότητες:

$$\kappa_{\text{min}} \leq \kappa \leq \kappa_{\text{max}}$$

  θεωρούνται, επειδή αντιστοιχούν σε **πρακτικά εφικτές τροχιές**.
- Για πραγματικό χρόνο:
  - τα εμπόδια προσεγγίζονται ως **κύκλοι**
- Ο τελικός συνδυασμός (v, ω) επιλέγεται από **objective function** που αξιολογείται μόνο σε επιτρεπτές/μη συγκρουόμενες περιοχές του velocity space.

CVM — Γεωμετρική Ερμηνεία

- Κάθε ζεύγος (v, ω) αντιστοιχεί σε τροχιά με συγκεκριμένη καμπυλότητα.
- Τα εμπόδια αποκλείουν ολόκληρες περιοχές του velocity space.
- Οι **tangent curvatures** δείχνουν ποιες καμπυλότητες οδηγούν σε σύγκρουση και ποιες είναι ασφαλείς.

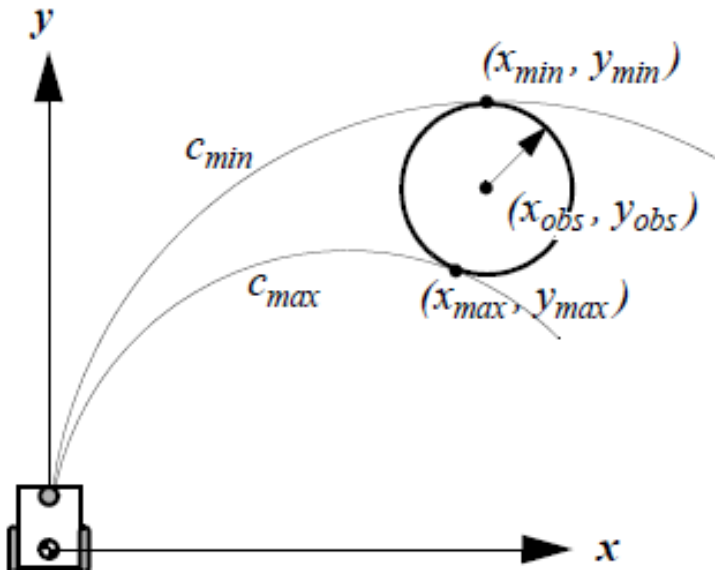


Figure: Tangent curvatures for an obstacle.
Source: Siegwart, Nourbakhsh & Scaramuzza, *Introduction to Autonomous Mobile Robots*.

Dynamic Window Approach (DWA) — Βασική Ιδέα

- Το Dynamic Window Approach (DWA) λαμβάνει υπόψη **κινηματικούς περιορισμούς** και **δυναμική** του ρομπότ.
- Ορίζεται ένας **χώρος ταχυτήτων**:

$$(v, \omega)$$

όπου v η γραμμική ταχύτητα και ω η γωνιακή ταχύτητα.

- Το ρομπότ θεωρείται ότι κινείται σε **τόξα κύκλου** κατά ένα μικρό χρονικό βήμα (όπως στο CVM).
- Από το σύνολο όλων των πιθανών (v, ω), επιλέγεται ένα μικρό υποσύνολο:

Dynamic Window

που περιλαμβάνει μόνο τις ταχύτητες που το ρομπότ μπορεί να επιτύχει στο επόμενο χρονικό διάστημα, με βάση τις επιταχύνσεις του.

Local Dynamic Window Approach — Αλγόριθμος

Βήματα του DWA:

1. Υπολογισμός του **dynamic window**: Όλα τα (v, ω) που μπορούν να επιτευχθούν στο επόμενο sample period, λαμβάνοντας υπόψη:

$$v_{\max},\ \omega_{\max},\ \dot{v}_{\max},\ \dot{\omega}_{\max}$$

2. Φιλτράρισμα του window ώστε να μείνουν μόνο οι **admissible velocities**: ταχύτητες που επιτρέπουν στο ρομπότ να **φρενάρει εγκαίρως πριν χτυπήσει εμπόδιο**.
3. Εφαρμογή objective function στα admissible (v, ω):

$$O(v, \omega) = \alpha \cdot \mathit{heading} + \beta \cdot \mathit{velocity} + \gamma \cdot \mathit{dist}$$

- **heading**: πρόοδος προς το goal.
- **velocity**: ενθάρρυνση γρήγορης κίνησης.
- **dist**: μέγιστη απόσταση από εμπόδια.

Global Dynamic Window Approach

- Η **global** εκδοχή του αλγορίθμου προσθέτει πληροφορία από global path planning.
- Προστίθεται ο NF1 / grassfire αλγόριθμος στο objective function:

$$O'(v, \omega) = O(v, \omega) + \delta \cdot NF1$$

- Το NF1 υπολογίζεται μόνο σε ένα **ορθογώνιο παράθυρο** μπροστά από το ρομπότ, για ταχύτητα πραγματικού χρόνου.
- Αν το NF1 δεν μπορεί να υπολογιστεί (π.χ. το ρομπότ είναι περικυκλωμένο):
  - το σύστημα υποβαθμίζεται σε local DWA,
  - μέχρι το ρομπότ να «ξεμπλοκάρει».
- Πλεονέκτημα: **Συνδυάζει local obstacle avoidance + global στόχευση**.
- Έχει επιδείξει λειτουργία πραγματικού χρόνου ($> 15Hz$) ακόμη και με πυκνά occupancy grids.

Dynamic Window Approach

Τι δείχνει η εικόνα:

- Το **ορθογώνιο παράθυρο** είναι το **dynamic window**: όλες οι ταχύτητες (v, ω) που το ρομπότ μπορεί να επιτύχει στο επόμενο χρονικό βήμα, με βάση τις επιταχύνσεις του.
- Οι «σκιερές» περιοχές δείχνουν **μη επιτρεπτές ταχύτητες**, γιατί οδηγούν σε σύγκρουση με εμπόδια.
- Οι **admissible velocities** είναι το καθαρό κομμάτι του παραθύρου όπου: το ρομπότ μπορεί να κινηθεί ασφαλώς.
- Από αυτές τις επιτρεπτές ταχύτητες επιλέγεται η καλύτερη μέσω objective function (πρόοδος προς το goal, ταχύτητα, απόσταση από εμπόδια).

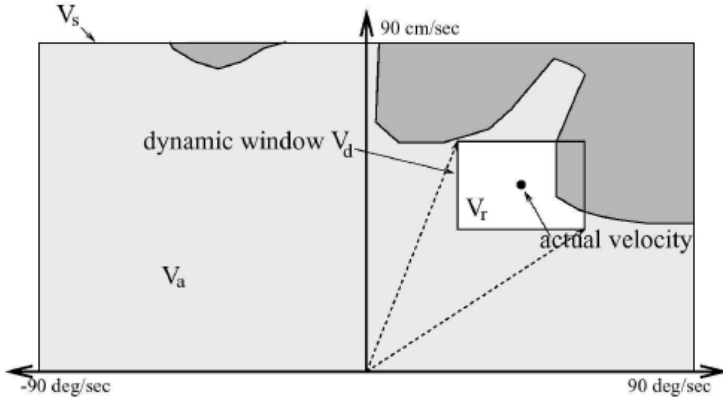


Figure: The dynamic window approach. The rectangular window shows the possible speeds (v, ω) and their overlap with obstacles in configuration space.

Schlegel & Nearness Diagram — Απλή Περιγραφή

Schlegel approach

- Λαμβάνει υπόψη **το πραγματικό σχήμα** και τη **δυναμική** του ρομπότ.
- Χρησιμοποιεί **raw laser data** και ένα Cartesian grid.
- Υπολογίζει την απόσταση σύγκρουσης για κάθε καμπυλότητα και **αποθηκεύει τα αποτελέσματα σε lookup tables** για πραγματικό χρόνο.
- Ιδανικό για ρομπότ με **πολύπλοκο σχήμα** (π.χ. forklift).

Nearness Diagram (ND)

- Προτείνει κατευθύνσεις βασισμένες στο **πόσο "κοντά"** είναι εμπόδια σε κάθε γωνία.
- Είναι σαν "πυξίδα" που αποφεύγει τοπικά εμπόδια με βάση την εγγύτητα.

GND, Gradient Method & Ego-Dynamic Space

Global Nearness Diagram (GND)

- Προσθέτει **global πληροφορία** στο ND, όπως στο Global DWA.
- Χτίζει χάρτη free space και ενημερώνεται συνεχώς από τους αισθητήρες.
- Συνδυάζει **τοπική αποφυγή + global κατεύθυνση**.

Gradient Method

- Υπολογίζει συνεχώς ένα **navigation function** (σαν NF1 αλλά πιο ομαλό).
- Δημιουργεί gradient που οδηγεί το ρομπότ σε **ομαλές, όχι "κοφτές"** τροχιές.

Ego-Dynamic Space

- Μετατρέπει τα εμπόδια σε **"δυναμικές αποστάσεις"** που εξαρτώνται από το φρενάρισμα και το sampling time.
- Προσθέτει **δυναμικά μοντέλα** πάνω από υπάρχοντες planners (ND, PF κ.ά.).
- Στόχος: ασφαλής αποφυγή εμποδίων με **πραγματικούς δυναμικούς περιορισμούς**.

Navigation Architectures — Τι είναι;

- Μια **Navigation Architecture** είναι ο τρόπος με τον οποίο ένα ρομπότ οργανώνει:
  - την αντίληψή του (sensing),
  - τον σχεδιασμό διαδρομής,
  - και τον έλεγχο κίνησης.
- Στόχος: το ρομπότ να φτάσει με ασφάλεια στο goal, αποφεύγοντας εμπόδια και κινούμενο ομαλά.
- Κάθε αρχιτεκτονική καθορίζει **ποιος παίρνει αποφάσεις**, **πότε**, και **με ποια πληροφορία**.

Main Navigation Architectures — Βασικοί Τύποι

1. **Ιεραρχική / Deliberative (Top–Down)**
   - Πρώτα χτίζει χάρτη, μετά σχεδιάζει διαδρομή, μετά κινείται.
   - Καλή σε στατικά/γνωστά περιβάλλοντα.
   - Αργή αντίδραση στα απρόοπτα.

2. **Αντιδραστική (Reactive)**
   - Αποφασίζει μόνο από **άμεσα αισθητήρια δεδομένα**.
   - Πολύ γρήγορη και κατάλληλη για δυναμικά περιβάλλοντα.
   - Δεν έχει "μεγάλη εικόνα", μπορεί να κολλήσει.

3. **Υβριδική (Hybrid)**
   - Συνδυάζει global σχεδιασμό + local αντίδραση.
   - Είναι η πιο συνηθισμένη προσέγγιση σε σύγχρονα ρομπότ.
   - Ο planner δίνει κατεύθυνση, ο local controller αποφεύγει εμπόδια.

How Navigation Architectures Work in Practice

- **Global Planner** ($\sim 0.001Hz$)
  Χρησιμοποιεί χάρτη (occupancy grid) για να βρει μια **μακρινή διαδρομή**. Π.χ. A*, NF1, D*.
- **Local Planner** ($\sim 1Hz$)
  Προσαρμόζει την κίνηση σε πραγματικό χρόνο ώστε να **αποφεύγει εμπόδια**. Π.χ. VFH, DWA, ND, CVM, Bubble Band.
- **Controller** ($\sim 150Hz$)
  Μετατρέπει την επιθυμητή ταχύτητα σε **ορθή κινηματική κίνηση**.

Το αποτέλεσμα: Το ρομπότ ακολουθεί ένα global μονοπάτι αλλά **αντιδρά έξυπνα** σε ό,τι δεν έχει προβλέψει ο χάρτης.