Motion Planning in a Stratified Workspace Manifold of a Quadruped Walking Robot

Report on Independent Study
Submitted for the partial fulfilment of requirements for Qualifying Exam

by
Subhrajit Bhattacharya

1 Introduction:

1.1 Stratified Workspace Manifolds:

Stratified manifolds are encountered in the configuration manifolds of many robots. Some typical examples are the walking robots [4, 5]. The configuration manifolds of such robots are in general high dimensional with some distinctive features. The most important characteristic feature of stratified manifolds are that they are filled with discontinuities [5]. These discontinuities give rise to certain sub-manifolds which are essentially the boundaries of the workspace manifold. These boundaries or sub-manifolds are known as strata [5]. The most interesting feature of a stratified manifold is that the system is allowed to stay only on these strata. That means even though the workspace of the robot is $N_{\mathbb{W}}$ dimensional, it is allowed to stay only on certain sub-manifolds of its workspace whose dimension in general is $N_{\mathbb{S}} < N_{\mathbb{W}}$. However there is still a lot more to these stratified manifolds. Even while residing on a stratum in a stratified manifold, the flow of the system is allowed only along certain directions on the strata, called leaves or foliages of the strata [5]. These foliations basically constitute some subsets of the tangent space of the strata. The following example in context to a quadruped robot will make the concept more clear.

Consider the example of a quadruped robot with each leg having three degrees of freedom. Thus we need 6 coordinates to define the position & orientation $g \in SE(3)$ of the robot's body fixed frame, and 3 coordinates for the position of the tip of each foot $\mathbf{r}_i \in \mathbb{R}^3$. Hence in general the configuration space of the robot is $6 + 3 \times 4 = 18$ dimensional. We denote this workspace manifold by $\mathbb{W}$. However the topology of this configuration manifold is characterized by discontinuities because of the facts like the robot is allowed to stay only above the terrain surface and that the foot tips of the robot are allowed to stay inside a limited workspace. This gives rise to the strata in the workspace. Consider the simple stratum that arises because of the fact that the foot tips of the robot cannot penetrate the terrain surface. We note that at any particular configuration of the robot, at least three of its legs need to touch the terrain surface. Hence we have four strata which are those sub-manifolds corresponding to which one leg is not touching the terrain surface, while the other three are touching the terrain surface. We denote the stratum corresponding to which the i^{th} leg is not touching the terrain surface by $\mathbb{S}_i$, $i = 1, 2, 3$ or 4. It is easy to note that

the dimension of each $\mathbb{S}_i$ is $18 - 3 = 15$. This is because of the fact that a leg touching the terrain surface will imply that the system's degree of freedom is reduced by 1. However the stratum which correspond to configurations for which all the four foot-tips are touching the terrain surface is basically a subset of the previously mentioned four strata. We denote this stratum by $\mathbb{S}_0$ and its dimension is $18 - 4 = 16$. It is not tough to realize that $\mathbb{S}_0$ actually forms connections between the $\mathbb{S}_i$'s for $i = 1, 2, 3$ or 4. In fact we can write,

$$\mathbb{S}_0 = \bigcup_{\substack{i,j \in \{1,2,3,4\} \\ i \neq j}} (\mathbb{S}_i \cap \mathbb{S}_j)$$

However it is to be noted that even within each of these strata there are more discontinuities because of the fact that the foot-tips are allowed to stay only within certain allowable regions relative to the robot's body-fixed frame. Moreover if we consider the force constraints, we'll find that even when three or four feet are touching the terrain surface, not all configurations are allowed to keep the reaction forces positive. But for the time being we simplify the problem by ignoring these complications.

Now we add the further constraint that there is no slippage at the foot-tips. This leads to the foliation of the strata. Let us consider a particular point on $\mathbb{S}_i$, $i = 1, 2, 3$ or 4. Thus the i^{th} leg is above the terrain, while all the other three legs are touching the terrain surface. Had there been no 'no-slippage' constraints, the following types of motions and/or their combinations would have been possible while staying on $\mathbb{S}_i$:

a. The robot could have slide along the terrain on the 3 feet that are touching the terrain surface.

b. It could have stayed at one place and swing its body to attain new position and orientation while keeping the foot-tips at the same place.

c. It could just have moved/swinged its i^{th} leg in the air.

These basically constitute the tangent bundle of the stratum $\mathbb{S}_i$, i.e. the possible directions in which the flow can take place.

However after we impose the no-slippage constraint, it is easy to note that the motion 'a' is no more allowed. However the robot may perform the remaining two types of motion while keeping its three feet touching the terrain surface. This leads to foliation of the stratum. The directions along which now motion is allowed are called the leaves/foliages of the stratum. At a particular point on the stratum, they basically constitute a subset (not necessarily a subspace) of the tangent space at that point. The foliation of the stratum thus constitute a subset of the tangent bundle of the stratum. Similarly we can identify foliations in $\mathbb{S}_0$ due to the no-slippage condition.

It is customary to denote the tangent bundle of the manifold $\mathbb{S}_i$ as $\mathbb{TS}_i$ [4, 5]. A particular leaf of the foliation of $\mathbb{S}_i$ is in general denoted by $\bar{\Delta}\mathbb{S}_i$ [5]. Since $\mathbb{S}_i$ has a dimensionality of 15, the tangent space at any point in it will be a 15 dimensional vector space. However the dimensionality of $\bar{\Delta}\mathbb{S}_i$ is further reduced by $3 \times 2 = 6$ since each of the 3 legs touching the terrain surface loose 2 degrees of freedom that they were having before the foliation. Thus $\bar{\Delta}\mathbb{S}_i$ is a 9 dimensional space. However, as mentioned before, it should be kept in mind that all these

manifolds and tangent spaces are marked by further discontinuities due to the limited working range of the foot-tips relative to the robot's body fixed frame, and the force constraints.

As mentioned before, the strata $\mathbb{S}_i$, $i = 1, 2, 3, 4$ are connected to each other via $\mathbb{S}_0$. It is thus evident that it would have been possible to move from one stratum to another if there were no foliations of the $\mathbb{S}_i$'s. However after imposing the no-slippage condition, we can still hope that the foliations $\bar{\Delta}\mathbb{S}_i$, $i = 1, 2, 3, 4$ will somehow be connected to each other via the foliations $\bar{\Delta}\mathbb{S}_0$. Thus now our primary aim is to plan a path through these foliations to reach as close to our goal as possible, but adhering to the workspace and force constraints all along.

Now we'll try to provide a diagrammatic representation of the workspace manifold. As it is evident, it is difficult to visualize such high dimensional manifolds and sub-manifolds. Even more difficult it will be to represent them on the paper. Thus we'll simplify the manifold grossly and will use a very crude 2-3 dimensional representations just to highlight some characteristic features of the workspace manifold. We will visualize the strata $\mathbb{S}_i$, $i = 1, 2, 3, 4$ as 2 dimensional surfaces, and hence the curves along which they intersect will constitute $\mathbb{S}_0$. The tangent space at a point on the manifold $\mathbb{S}_i$ will be the tangent plane at a point on the surfaces representing it. Then a foliation will be represented by a particular direction along the manifold in which the flow can take place. The following figure illustrates these in a lower dimensional representation.

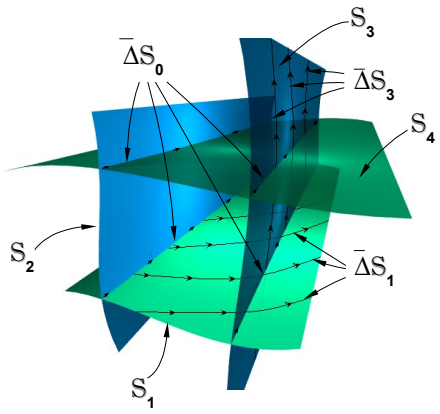


Fig. 1: Schematic reduced dimensional representation of the stratified manifold of a quadruped robot

It is to be noted that in the actual manifold all the $\mathbb{S}_i$, $i = 1, 2, 3, 4$ intersect with each other, and the intersections essentially forms a common manifold $\mathbb{S}_0$. That means $\mathbb{S}_0$ itself is a connected manifold. In the above figure the intersecting curves between any two $\mathbb{S}_i$ and $\mathbb{S}_j$ may be looked upon as foliations of $\mathbb{S}_0$. Thus, even if all these intersection curves representing the foliations that make up $\mathbb{S}_0$ are connected, it may not possible to move directly from any of these curves to another. Moreover the workspace & force constraints will induce discontinuities, the result of which will be torn and broken manifolds with boundaries. Although

the representation in *Fig. 1* is quite inaccurate, later on we'll use concepts from it in planning our strategy for motion along the manifold.

1.2 Dynamics of walking vehicles:

The development of walking vehicles has been inspired by biological walking animals [1]. It is known that animals which walk can travel over difficult terrains with much agility which is rather difficult, if not impossible, for wheeled or tracked vehicles. This possibility of dexterity has inspired the study and development of walking vehicles. However the issues like complex dynamics, large number of degrees of freedom, difficulty to control and energy efficiency of such walking robots pose a major challenge in their design and development. As we have just seen, the workspace manifold of walking robots are filled with discontinuities and limitations in the directions along which motion can take place. Thus motion planning in such manifolds is a rather challenging task.

The study of walking robots has been quite an old and extensive one. The earliest of the walking vehicles were built during the middle of the 20th century. The rapid development of computer technology and computational powers assisted in the development of walking robots [1]. Although there are several examples of successful development of hexapod, quadruped and even biped walking robots, their application has rather been limited by the specific natures of the working terrain they are meant for, comparatively low energy efficiency and challenges in the development of control systems. The present day industrial application of walking robots is rather limited. Although a few domestic applications have been implemented, no suitable cost-effective industrial application is yet in use. However in future, further development of walking robots and computational systems may make them more applicable. Moreover it is interesting to study the dynamics and workspace manifold of a system like that of a walking robot and coming up with methods to control them. This motivated us to take up this particular research topic for study.

The dynamics of a system with high degree of freedom like a quadruped robot may be highly complex. However with suitable assumptions the problem may be simplifies to a great extant. Assumptions like negligible inertia of the legs are quite acceptable. Moreover, as we will discuss later, we'll make the assumption of a quasi-static model, hence ignoring any inertial forces on the system. This will drastically simplify our system and will reduce it into a system of 6 equations - 3 for forces and 3 for moments. The force constraints (that the force system at the foot-tips touching the terrain surface should lie within the allowed friction cone) are imposed as inequalities that need to be satisfied.

2 Relevant literature:

In [5] an extensive study on the nature of Stratified manifolds and Control techniques for systems residing on such manifolds has been discussed. As described before, stratified manifolds are those where discontinuities form an inherent part of the manifold. The configuration manifold of our present system, the quadruped robot, is such a manifold. This makes the issue of controllability in stratified manifolds is a rather complex study because of the presence of the discontinuities giving rise to the strata and foliations of the strata. Issues like

controllability and controller design (motion planning) has been discussed in great details in [5].

Similar discussions has been made in [4] where modular robots have been studied in great details. Motion planning in Stratified manifolds has also been discussed to some extent. Issues of trajectory generation and gait stability has also been discussed in [5]. The dynamics of modular robots has been studied in [4]. Lagrangian mechanics has been used to incorporate the dynamics as well as the holonomic and non-holonomic constraints involved.

3 Overview of the problem and solution approach:

In our present problem we make the assumption of quasistatic model. Thus we ignore all the inertia forces and inertia moments. This simplifies the system to a great extent. The dynamics of the system now reduces to balance of forces and moments. The fact that the configuration manifold is a stratified manifold has been incorporated in a bunch of equalities and inequalities describing the discontinuities and the directions of possible flow in the manifold and the strata. Moreover throughout the problem we assume that the center of mass of the robot coincides with the origin of the robot's body fixed frame.
In the present problem we'll primarily be concerned with determining a suitable flow path in the stratified configuration manifold to reach a target point, starting from an initial point. Since the space is high dimensional and there are constraint of moving along the leaves of the foliations in the manifold, determining a global optimal path will be rather difficult. We have addressed this optimization problem by separating it into local and global optimization problems.

In the present work we have done a systematic study of the dynamics of a quadruped robot with a target to develop an open-loop controller for the robot for the purpose of motion planning. We have analyzed the quasistatic model of a quadruped and addressed the issues like equilibrium of forces and moments, friction constraints at the footholds, etc. Given a terrain with an initial and target point we'll come up with an open-loop controller for motion planning, taking care of the foresaid issues and trying to attain some optimization in the process. In order to attain some level of optimization we divide our problem into two parts:

i. *Local Planning:* Given a trajectory near the surface of the terrain in the 3 dimensional physical space of the robot, we try to plan the motion such that the center of mass of the robot follows the trajectory. At present we have two versions of the way to deal with this problem. In the first way we use a *hard persuasion* method, where we perform exact tracking of the trajectory. Whereas in the second case we use a *soft persuasion* method, where we relax on the condition of exact tracking, and instead try to minimize the distance of the center of mass of the robot from the given trajectory. While the first method is computationally less expensive, the second method is more adaptive and robust. As we'll discuss later, the process of *local planning* consists of primarily two phases — moving the body and placing foot.

ii. *Global Planning:* Once we are able to make the robot move along a given

trajectory, we can calculate some costs associated with the trajectory. In this part of the problem we try to make variations in the trajectory and determine the one which has the minimum cost associated with it. As quite evident, this process of *global planning* will be extremely computationally expensive.

Our system consists of a rigid body (the body of the robot along with it's legs) which is in contact with the terrain surface at at least 3 points(the footholds).

Fig. 2: The Little Dog quadruped robot

The forces acting at these points include the normal reaction from the terrain as well as the frictional force. In the *local planning* phase of the motion, it is possible to move the robot body relative to a fixed foothold by varying the joint angles of the legs (This basically corresponds to the flow along a leaf of the foliation of the strata, $\bar{\Delta}\mathbb{S}_i$). In particular, given a set of footholds, we can determine the joint angles required to attain a desired position and configuration of the robot body (within the bounds in the configuration space) by using inverse kinematics. Moreover if at some instant there are 3 feet in contact with the ground, it is possible to change footholds by putting down the fourth feet and subsequently lifting up one of the three feet that were previously on the ground (this basically corresponds to the movement of the robot from $\mathbb{S}_i$ to $\mathbb{S}_j$, $i, j = 1, 2, 3, 4$ via the connecting $\mathbb{S}_0$). Thus there are basically two phases of motion corresponding to the flow along two different sets of foliations: One is changing the position and configuration of the robot body by changing the leg joint angles, but keeping the footholds fixed; The other being when the footholds are being changed.

4 Notations:

4.1 Notations for describing the configuration manifold:

$\mathbb{W}$	18 dimensional configuration manifold of the quadruped robot.
$\mathbb{S}_i$	Strata corresponding to which three feet are touching the surface of the terrain and the i^{th} foot is above the terrain surface.
$\mathbb{S}_0$	Stratum corresponding to which all four feet are touching the surface of the terrain.
$\mathbb{TS}_i$	Tangent bundle of stratum $\mathbb{S}_i$.
$\bar{\Delta}\mathbb{S}_i$	A leaf of the foliation of $\mathbb{S}_i$.

4.2 Coordinate systems, geometric parameters and constants:

G	The global frame of reference fixed to the terrain.
B	The body fixed frame of the robot. The center of mass of the robot body coincides with the origin of this frame. (Refer to [6] for further details about the body-fixed frame).
${}^{G}\mathbf{r}$, ${}^{B}\mathbf{r}$	The homogeneous component representation of vector $\mathbf{r}$ in the global and body fixed frames respectively.
$\theta_{i,j}$	The angle of the j^{th} joint of the i^{th} leg, $i = 1, 2, 3$ or 4, $j = 1, 2$ or 3. (Refer to [6] for details about the *zero configuration* angles).
$\theta_{i,j}^{min}, \theta_{i,j}^{max}$	The lower and upper bounds on the joint angles [6].
$\theta_{i,j}^{nor}$	Joint angles corresponding to a 'normal' configuration of the leg joints. This is used later for defining some optimization objectives.
$\mathbf{r}_{i,j} \in \mathbb{R}^3$	The position vector of a joint.
$\mathbf{r}_{i,j}^0 \in \mathbb{R}^3$	The position vector of a joint at the *Zero Configuration* of the legs, i.e. the joint angles being all zero.
$\mathbf{r}_c$	Position vector of center of mass of the robot. The component representation of $\mathbf{r}_c$ is made in the global frame.
P	Set of points that define the body of the robot. We approximate the robot's body as a polyhedron. Thus P is the set of all the vertices of the polyhedron.
m	Mass of the robot body. We ignore the mass of the legs.
$\mathbf{g}$	Acceleration due to gravity vector.

4.3 State-space variables that parametrize the configuration manifold:

$g \in SE(3)$	The homogenious transformation matrix [7] describing the position and orientation of the robot's body-fixed frame measured in the global frame. (6 independent variables)
$\mathbf{r}_i \in \mathbb{R}^3$	The position vector defining tips of each leg, $i = 1, 2, 3$ or 4. (3×4 independent variables)
$\Psi = \{g, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4\}$	A particular configuration of the robot – a point in the configuration manifold.
$\Psi.\chi$ or $\tilde{\chi}(\Psi)$	The component 'χ' of Ψ. Here 'χ' is either g or $\mathbf{r}_i$.
Ξ	A collection/set of Ψ's. Occasionally (in the *hard persuasion* procedure) we may also include the information about the λ corresponding to the $\mathbf{r}_c$ of each Ψ.

4.4 Algebraic Sets (inequalities) for implicitly defining the allowed regions in the workspace — defining the discontinuities in the workspace manifold:

f_T Defines the terrain surface. $f_T(\mathbf{r}) \begin{cases} > 0 & \text{if } \mathbf{r} \text{ is above the terrain surface.} \\ = 0 & \text{if } \mathbf{r} \text{ is a point on the terrain surface.} \\ < 0 & \text{if } \mathbf{r} \text{ is below the terrain surface.} \end{cases}$

ζ_i Defines the dextrous workspace of the i^{th} foot. If S_i is the set of all points that the i^{th} foot tip can reach at a particular position/configuration, g, then we define, $\zeta_i(\mathbf{r}) \begin{cases} > 0 & \text{if } \mathbf{r} \in S_i \\ < 0 & \text{if } \mathbf{r} \notin S_i \end{cases}$

4.5 Forces and Normals:

$\mathbf{F}_i$	Force acting at the i^{th} foot-tip. This consists of both the normal reaction and the friction forces.
$\mathbf{n}_i$	The normal at the i^{th} foot-tip. The normal or *pseudo-normal* at an arbitrary point $\mathbf{r}$ in space is calculated as $\left[\frac{\nabla f_T}{\lvert\nabla f_T\rvert}\right]_{\mathbf{r}}$.
$\nabla_s f_T\rvert_{\mathbf{r}}$	*Smoothened pseudo-normal* at an arbitrary point $\mathbf{r}$. The smoothening is performed by taking an weighted average of *pseudo-normals* over points in the neighborhood of $\mathbf{r}$.
μ	The coefficient of friction between the terrain surface and the foot-tips.

4.6 Variables and parameters used for motion planning:

$\mathbf{R}_\tau$	A point on a given trajectory, τ.
λ	Parameter that parametrizes a given trajectory. Thus $\mathbf{R}_\tau(\lambda)$ represents the parametrization of the trajectory τ.
$S = \{s_1, s_2, \cdots, s_n\}$	Gait step sequence for n steps. Essentially this specifies the sequence of strata $\mathbb{S}_i$ which the robot needs to follow while moving to one strata to another via $\mathbb{S}_0$. Here s_k is the numeric index of the strata that the robot must be on at the k^{th} step.

4.7 Other notations and conventions:

$SE(3)$	The Special Euclidian Group.
$SO(3)$	The Special Orthogonal Group.
$\widehat{\ }$	The *hat operator* that maps the 3×1 component vector space to $so(3)$ [7].
$\tilde{\varepsilon}(g)$	The *Compact Exponential Coordinate* representation of any $g \in SE(3)$. We define this slightly different, but analogous to the exponential coordinate representation of g. Given a $g = \begin{bmatrix} R_{3\times3} & {}^{G}\mathbf{v}_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}$, corresponding to the $R \in SO(3)$ we can obtain the unit vector along the axis of rotation and the rotation angles (the exponential coordinates)[7], and represent them by ω and ϑ respectively. Then we define the *compact exponential coordinate* representation of g as the 6×1 vector given by $\tilde{\varepsilon}(g) = \begin{bmatrix} {}^{G}\mathbf{v} \\ \vartheta\omega \end{bmatrix}$.
$\tilde{R}(g)$,$\tilde{\mathbf{v}}(g)$, $\tilde{\omega}(g)$,$\tilde{\vartheta}(g)$, $\widetilde{\vartheta\omega}(g)$	We represent the quantities R, $\mathbf{v}$, ω and ϑ associated with a particular g (as described above) as $\tilde{R}(g)$, $\tilde{\mathbf{v}}(g)$, $\tilde{\omega}(g)$ and $\tilde{\vartheta}(g)$ respectively. Moreover we represent the vector $\vartheta\omega$ associated with g by $\widetilde{\vartheta\omega}(g)$. We will sometimes use an alternative shorter way of representing all the above mentioned quantities. For denoting these same quantities we can write $g.R$, $g.\mathbf{v}$, $g.\omega$, $g.\vartheta$ and $g.(\vartheta\omega)$ respectively.
$\tilde{g}(\varepsilon)$	The inverse of *Compact Exponential Coordinate* representation.

5 Tools used for motion planning and optimization processes:

5.1 Forward and Inverse Kinematics:

Figure 3, and the equations following it describe the Inverse and Forward Kinematics for a leg of the robot. Please refer to [7] for details of the computation process. The numerical values for the different symbols mentioned can be found from [6].

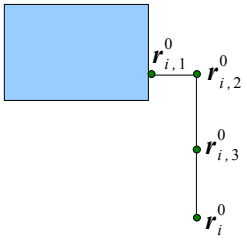


Fig. 3: One single leg at the *Zero Configuration*

5.1.1 Forward Kinematics:

We define

$$\bar{g}_{i,j} = \exp[{}^B\widehat{\xi}_{i,j}\theta_{i,j}] \tag{1}$$

where,

$$\widehat{\xi}_{i,j} = \begin{bmatrix} \widehat{\omega}_{i,j} & -\omega_{i,j} \times \mathbf{r}^0_{i,j} \\ 0 & 0 \end{bmatrix}$$

where $\omega_{i,j}$ denotes [7] an unit vector along the axis of the j^{th} joint of the i^{th} leg at the leg's Zero Configuration.

Then we can find the position vectors of the joints and the foot-tips as follows:

$$\begin{aligned} {}^B\mathbf{r}_{i,1} &= {}^B\mathbf{r}^0_{i,1} \\ {}^B\mathbf{r}_{i,2} &= \bar{g}_{i,1}\ {}^B\mathbf{r}^0_{i,2} \\ {}^B\mathbf{r}_{i,3} &= \bar{g}_{i,1}\,\bar{g}_{i,2}\ {}^B\mathbf{r}^0_{i,3} \\ {}^B\mathbf{r}_{i} &= \bar{g}_{i,1}\,\bar{g}_{i,2}\,\bar{g}_{i,3}\ {}^B\mathbf{r}^0_{i} \end{aligned} \tag{2}$$

Thus, knowing the $\theta_{i,j}$'s, we can find the $\mathbf{r}_i$'s. Hence we have performed the Forward Kinematics. [Note that all the vector quantities are expressed as their homogeneous components in the body-fixed coordinate frame.]

5.1.2 Inverse Kinematics:

We note that $\mathbf{r}_{i,1}$ always lie on the axes of $\widehat{\xi}_{i,1}$ and $\widehat{\xi}_{i,2}$. Hence, ${}^B\mathbf{r}_{i,1} = \bar{g}_{i,1}\,\bar{g}_{i,2}\ {}^B\mathbf{r}_{i,1}$. Again, ${}^B\mathbf{r}_i = \bar{g}_{i,1}\,\bar{g}_{i,2}\,\bar{g}_{i,3}\ {}^B\mathbf{r}^0_i$. Hence,

$$|\ {}^B\mathbf{r}_i - {}^B\mathbf{r}_{i,1}\ | = |\ \bar{g}_{i,3}\ {}^B\mathbf{r}^0_i - {}^B\mathbf{r}_{i,1}\ |$$

This enables us to use *Paden Kahan Subproblem 3* [7] to solve for $\theta_{i,3}$. Hence $\bar{g}_{i,3}$ can be computed.

Thus once $\bar{g}_{i,3}$ is known, we can use ${}^{B}\mathbf{r}_i = \bar{g}_{i,1}\ \bar{g}_{i,2}\ \bar{g}_{i,3}\ {}^{B}\mathbf{r}_i^0$ and *Paden Kahan Subproblem 2* [7] to solve for $\theta_{i,1}$ and $\theta_{i,2}$.

5.2 Computation of Forces:

The quadruped robot standing on its three feet is a force indeterminate system. There are three components of forces acting on each foot tip which need to be solved for. Thus we have 9 unknowns. However the force and moment balance equations give us only 6 equations. Hence there will not be an unique system of forces that will keep the robot in equilibrium at a particular given configuration.

There are various ways to deal with this difficulty. One way is to determine the compliance of each leg along two different directions and assume zero compliance along a third direction. This will reduce the number of unknowns to 6, and hence we can solve for the forces. However the main challenge in this procedure is the determination of the compliances, which will be a function of the configuration of the legs.

A comparatively easier and probably more legitimate approach is to assume that the forces at the foot-tips have zero interaction component. Any system of forces acting on a rigid body may be decomposed into two components [2] — *equilibrating component* and *interaction component*. The interaction force between any two foot-tips is defined as the component of the difference of the contact forces along the line joining the two foot-tips. With this definition it can be proved [2] that the equilibrating component of the forces form a *helicoidal vector field*. A helicoidal vector field can be defined using 6 independent parameters. Hence if we assume that the interaction components are zero, we can solve for the forces from the force & moment balance equations. The reasoning behind assuming a zero interaction force component is that there is no *internal* force between the legs that try to push them apart. This assumption is valid if the terrain surface is moderately flat and the robot is placed *lightly* on the terrain. However in general this way of solving the forces won't actually give the exact solution. The solution obtained by this method is just a particular solution. Whereas the actual force system still remains indeterminate. *Figure 4* and the equations that follow describe how the equilibrating component of the forces are computed.

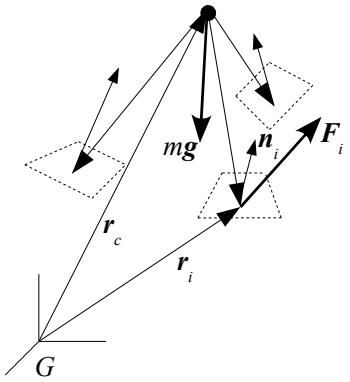


Fig. 4: The force and moment system of the quadruped robot with three feet touching the terrain surface

If the *interaction component* of all the $\mathbf{F}_i$ are zero, then the *equilibrating component* can be written as [2],

$$\mathbf{F}_i = L\bar{\mathbf{u}} \times (\mathbf{r}_i - \rho) + hL\bar{\mathbf{u}}$$

Thus we can write

$$\mathbf{F}_i = \mathbf{u} \times \mathbf{r}_i + \mathbf{w} \tag{3}$$

where, $\mathbf{u} = L\bar{\mathbf{u}}$ and $\mathbf{w} = h\mathbf{u} + \rho \times \mathbf{u}$.
Then from the force and moment balance equations

$$\begin{array}{l} \sum_i \mathbf{F}_i + m\mathbf{g} = 0 \\ \sum_i \mathbf{F}_i \times (\mathbf{r}_i - \mathbf{r}_c) = 0 \end{array} \tag{4}$$

it can be proved quite easily that,

$$\begin{bmatrix} -\widehat{\mathbf{R}} & \eta\mathbf{I} \\ \widehat{\widehat{\mathbf{R}}} - \widehat{\mathbf{r}}_c\widehat{\mathbf{R}} & \eta\widehat{\mathbf{r}}_c - \widehat{\mathbf{R}} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} -m\mathbf{g} \\ 0 \end{bmatrix} \tag{5}$$

where, $\widehat{\mathbf{R}} = \sum_i \widehat{\mathbf{r}}_i$ and $\widehat{\widehat{\mathbf{R}}} = \sum_i \widehat{\mathbf{r}}_i^2$.
Thus from equation (5) we can solve for $\mathbf{u}$ and $\mathbf{w}$.
It is to be noted that the $\sum_i$ denotes summation over all those feet which are *carrying the weight* of the robot. When on a stratum $\mathbb{S}_j$, the sum will be over all $i \neq j$. But when on $\mathbb{S}_0$, the choice of the *weight carrying feet* will be subjective and will depend on which stratum we want to move to next or we were previously. After all, the solution returned by this method is just one possible solution. By changing the choice of the *weight carrying feet* we can obtain a few other possible solutions.
The *hat operator*($\widehat{\ }$) used here generates a 3×3 matrix from a component vector. Essentially it is a tensor. Refer to [7] for more details.

6 A short discussion on Matlab's Optimization Toolbox - *fmincon*:

For a most of our optimization problems we have made extensive use of Matlab's optimization toolbox. As we'll see, most of our optimization problems will be highly nonlinear, non-smooth (sometimes even with discontinuities) and essentially non-convex. Matlab's *fmincon* is a powerful optimization tool and is supposed to be capable enough to deal with such problem. Hence, essentially we used the *fmincon* tool like a black box. Since a major part of our problem relies on the successful working of *fmincon*, it is worthful to have some understanding about its internal working procedures. More details about Matlab's Optimization Toolbox can be found at the MathWorks website.

The following are some of the methods used by *fmincon* for solving constrained nonlinear optimization problems. The descriptions about these methods are primarily excerpts from MATLAB's user manual. Please refer to MATLAB's user's manual for more details.

i *Trust-region method for Nonlinear Minimization: To understand the trust-region approach to optimization, consider the unconstrained minimization*

problem, $\min_x f(x)$*, where the function takes vector arguments and returns scalars. Suppose we are at a point* x *in* n*-space and we want to improve, i.e., move to a point with a lower function value. The basic idea is to approximate* f *with a simpler function* q *which reasonably reflects the behavior of function* f *in a neighborhood* N *around the point* x*. This neighborhood is the trust region. A trial step* s *is computed by minimizing (or approximately minimizing) over* N*.*

ii. *Sequential Quadratic Programming (SQP): SQP methods represent the state of the art in nonlinear programming methods. Schittkowski, for example, has implemented and tested a version that outperforms every other tested method in terms of efficiency, accuracy, and percentage of successful solutions, over a large number of test problems. Based on the work of Biggs, Han, and Powell, the method allows us to closely mimic Newton's method for constrained optimization just as is done for unconstrained optimization. At each major iteration, an approximation is made of the Hessian of the Lagrangian function using a quasi-Newton updating method. This is then used to generate a QP subproblem whose solution is used to form a search direction for a line search procedure.*

There are several other methods that are used by *fmincon* to ensure robustness of the optimization toolbox. However we'll restrict our present discussion on MATLAB's optimization toolbox upto this.

7 Algorithms used at different stages of *Local Planning*:

In this section we'll discuss different algorithms that we have used at the various stages of motion planning. They consist of an interpolation procedure, some optimization problems, and a search operation. In the subsequent sections we'll use these algorithms and patch them up to define full algorithms for motion planning along a given trajectory.

7.1 Interpolation between two points on a particular leaf of a stratum:

As the heading suggests, here we have been given two points in the configuration space that lie on the same leaf of the foliation of a particular stratum, $\mathbb{S}_i$, $i = 0, 1, 2, 3$ or 4. Let these points be $\Psi_1 \in \mathbb{S}_i$ and $\Psi_2 \in \mathbb{S}_i$ respectively. Our objective is to interpolate between these two points such that the interpolated points lie on the stratum $\mathbb{S}_i$. We will assume that $\Psi_1 \in \mathbb{S}_i$ and $\Psi_2 \in \mathbb{S}_i$ lie close enough so that there isn't any discontinuity present in the region *between* them and we can assume the manifold $\mathbb{S}_i$ to be smooth while performing the interpolation.

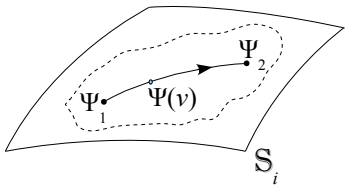


Fig. 5: Interpolating on a leaf of a stratum

In *figure 5* the dotted boundary represents discontinuity. Thus we cannot come up with a point beyond that boundary as an interpolated point. However with our foresaid assumption we'll assume that the boundary is far enough to be worried about. We define a parameter $\nu \in [0, 1]$ such that as ν changes from 0 to 1, we move from Ψ_1 to Ψ_2.

We define the *compact exponential coordinate representation* of any $g \in SE(3)$ as $\varepsilon = \tilde{\varepsilon}(g)$. The inverse transformation from this *compact exponential coordinate representation* to $SE(3)$ is simply written as $g = \tilde{g}(\varepsilon)$. Refer to *section 4.7* for details of this notation.

Interpolation on $\mathbb{S}_i, i \neq 0$: When $i \neq 0$ we define $\Psi_1 = \{g_1, \{\mathbf{r}_i^1\}, \{\mathbf{r}_j\}_{j\neq i}\}$ and $\Psi_2 = \{g_2, \{\mathbf{r}_i^2\}, \{\mathbf{r}_j\}_{j\neq i}\}$ (Note: Here the position vectors of the foot-tips are represented in the global coordinate frame, G. Refer to section *4.3* for the notations). Hence the interpolated point is given by,

$$\Psi(\nu) = \{\tilde{g}\left((1-\nu)\tilde{\varepsilon}(g_1) + \nu\tilde{\varepsilon}(g_2)\right), \{(1-\nu)\mathbf{r}_i^1 + \nu\mathbf{r}_i^2\}, \{\mathbf{r}_j\}_{j\neq i}\} \qquad (6a)$$

Interpolation on $\mathbb{S}_0$: When $i = 0$ we define $\Psi_1 = \{g_1, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4\}$ and $\Psi_2 = \{g_2, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4\}$ (Once again the position vectors of the foot-tips are represented in the global coordinate frame, G. Refer to section *4.3* for the notations).

Hence the interpolated point is given by,

$$\Psi(\nu) = \{\tilde{g}\left((1-\nu)\tilde{\varepsilon}(g_1) + \nu\tilde{\varepsilon}(g_2)\right), \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4\} \tag{6b}$$

7.2 Finding an optimum point on a particular leaf of $\mathbb{S}_0$ such that the center of mass coincides with a desired point on a trajectory in the physical space of the robot — the *hard persuasion* strategy:

This is basically an optimization problem with well defined inequality & equality constraints and search space. However we will have freedom in choosing the optimization objective. The optimization problem in general will be non-convex and highly nonlinear. The physical picture of this problem is that the robot is standing with its four feet touching the terrain surface and hence the coordinates of its four foot-tips are fixed. Thus now the robot has 6 degrees of freedom, which is essentially the dimensionality of the leaf in $\mathbb{S}_0$. Moreover in this *hard persuasion* method we are given 3 more equality constraints that makes the center of mass coincide with a desired point $\mathbf{r}_c^*$ (the superscript '*' denotes that this is a given/constant quantity) on a predefined trajectory τ. Thus we need to search on a 3 dimensional sub-manifold for a suitable orientation of the robot body. It is to be noted that since the position of the foot-tips and the center of mass are all given, the force system computed by the method in *section 5.2* will be independent of the point in the 3 dimensional search space that we are in. If $\Psi = \{g, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4\}$ defines a particular point in the configuration space, the optimization problem can defined as below:

Given/constants: The first three elements of $\tilde{\varepsilon}(g)$, are related directly to $\mathbf{r}_c^*$. Hence they are given. In fact with our assumption that the center of mass of the robot coincides with the center of the body fixed frame, we can write $\tilde{\mathbf{v}}(g) = \mathbf{r}_c^*$. Moreover the trajectory τ along with its parametrization is known, $\mathbf{R}_\tau(\lambda)$. Furthermore all the footholds, $\mathbf{r}_i = \mathbf{r}_i^*$, $i = 1, 2, 3, 4$ are known. The footholds remain fixed in the particular leaf of $\mathbb{S}_0$. And finally we are also given which of the three feet area actually *carrying the weight* of the robot. These three feet will be used to compute the forces by the method as mentioned under *section 5.2*. Say the index of the foot which will not be used to compute the forces be i_f. Hence this optimization problem can be represented completely by the following notation that mention the given/constant parameters:

$$\Psi_{optimum} = \Theta_1(\mathbf{r}_c^*, \tau, \mathbf{r}_1^*, \mathbf{r}_2^*, \mathbf{r}_3^*, \mathbf{r}_4^*, i_f) \tag{7}$$

Search space: The search space consists of last three elements of the 6×1 vector $\tilde{\varepsilon}(g)$. They basically represent the orientation of the robot. That is, our search space consists of the elements of the 3×1 vector $\widetilde{\vartheta\omega}(g)$.

Bounds on search space: Since we keep the twist angle corresponding to g between $-\pi$ and $+\pi$, each of the search space variables should lie in $[-\pi, \pi]$.

Equality constraints: Essentially there isn't any. However for better convergence of *fmincon*, we impose a constraint by specifying the imaginary part of the joint angles returned by inverse kinematics to be equal to zero.

Inequality constraints: There are three sets of inequality constraints:

a. The vector that makes up the search space, $\widetilde{\vartheta\omega}(g)$, as evident, have a magnitude less than π since we restrict the value of ϑ to be in $[-\pi, \pi]$ and ω is an unit vector. Thus we impose the inequality constraint

$$| \widetilde{\vartheta\omega}(g) | \leq \pi$$

b. We desire that the robot body should maintain a minimum clearance of δ_c from the terrain surface. Thus we have the inequality constraint

$$\min_{\mathbf{r} \in P} f_T(\mathbf{r}) \geq \delta_c$$

c. The joint angles obtained from inverse kinematics should lie within the allowed upper and lower bounds. Thus we must have

$$\theta_{i,j}^{min} \ \leq \ \theta_{i,j} \ \leq \ \theta_{i,j}^{max};\ i = 1, 2, 3, 4;\ j = 1, 2, 3$$

We note that the force constraints will not be function of the search space variables. Hence we don't include them in the list of inequalities. However we perform an initial calculation of the forces and check if they satisfy the required conditions (please refer to the next sub-section). If they don't, then that will imply that there does not exist any feasible solution that will satisfy the force constraints.

Optimization objectives: The optimization objectives are not unique and may be changed to obtain more desired results. The following are a few objective functions that we have used:

a. We define a *desired orientation* of the robot body such that it remains parallel to the terrain surface and remains oriented along the tangent at that point on the trajectory. Let the given trajectory be $\mathbf{R}_\tau$, parametrized by parameter λ. Thus we have a reference forward direction $\mathbf{u}_f = \frac{\partial \mathbf{R}_\tau}{\partial \lambda}\big|_{\mathbf{r}_c^*}$ and a reference normal direction $\mathbf{u}_n = \nabla_s f_T|_{\mathbf{r}_c^*}$. Then we define the following two objective functions to be maximized,

$$\sigma_1 = {}^G\mathbf{u}_f \ \cdot \ \tilde{R}(g) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\sigma_2 = {}^G\mathbf{u}_n \ \cdot \ \tilde{R}(g) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

b. We desire that the joint angles remain well within their working range or they don't assume too much extreme values so that the legs assume very odd configurations. That's why we defined a *normal* configuration of the legs in *section 4.2*. Hence we now define another objective to be maximized,

$$\sigma_3 = \frac{1}{1 + \|(\theta - \theta^{nor})K\|}$$

where, θ and θ^{nor} are 4×3 matrices whose elements are $\theta_{i,j}$ and $\theta_{i,j}^{nor}$ respectively (note that $\theta_{i,j}$ are obtained from Ψ by inverse kinematics); K is a 3×3 scaling matrix which we chose to be the identity matrix; and $\| \cdot \|$ is a suitable matrix norm.

Thus we define the final combined objective to be *minimized*,

$$\sigma = -{\sigma_1}^{\gamma_1} {\sigma_2}^{\gamma_2} {\sigma_3}^{\gamma_3}$$

where, γ_1, γ_2 and γ_3 are suitable weight factors.

7.3 Finding a point on a particular leaf of $\mathbb{S}_0$ such that the center of mass is close to a desired point on a trajectory in the physical space of the robot — the *soft persuasion* strategy:

This problem is very similar to the previous one, except that now we relax the constraint that the center of mass of the robot should coincide with a given $\mathbf{r}_c^*$ (the superscript '*' denotes that this is a given/constant quantity). As a consequence we add three mode dimensions to our search space which correspond to the position of the center of mass. This undoubtedly increases the computational cost for this algorithm. Moreover we have the additional set of inequalities defining the force constraints. Thus once again, if $\Psi = \{g, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4\}$ defines a particular point in the configuration space, the optimization problem can defined as:

Given/constants: The trajectory τ that is to be followed, along with its parametrization, is given $\mathbf{R}_\tau(\lambda)$. A given point in the space is $\mathbf{r}_c^*$ (note that here we relax the condition that $\mathbf{r}_c^*$ is a point on the given trajectory. *Section 8.2.1* describes a particular method for finding a good $\mathbf{r}_c^*$.) and one of our objectives will be to keep the center of mass of the robot as close as possible to this point. Moreover all the footholds, $\mathbf{r}_i = \mathbf{r}_i^*$, $i = 1, 2, 3, 4$ are known. The footholds remain fixed in the particular leaf of $\mathbb{S}_0$. Moreover as before, the *force carrying feet* are mentioned. Say the index of the foot which will not be used to compute the forces be i_f. Hence this optimization problem can be represented completely by similar notation as before:

$$\Psi_{optimum} = \Theta_2(\mathbf{r}_c^*, \tau, \mathbf{r}_1^*, \mathbf{r}_2^*, \mathbf{r}_3^*, \mathbf{r}_4^*, i_f) \tag{8}$$

Search space: The search space consists of the elements of the 6×1 vector $\tilde{\varepsilon}(g)$. They basically represent the position and orientation of the robot's body fixed frame. Thus we have a 6 dimensional search space.

Bounds on search space: Since we keep the twist angle corresponding to g between $-\pi$ and $+\pi$, the last three search space variables should lie in $[-\pi, \pi]$. That is, each of the elements of $\widetilde{\vartheta\omega}(g)$ should lie in $[-\pi, \pi]$.

Equality constraints: As before, there isn't any proper equality constraint. However for better convergence of *fmincon*, we impose a constraint by specifying the imaginary part of the joint angles returned by inverse kinematics to be equal to zero.

Inequality constraints: The first three sets of inequality constraints are the same as before. In addition we have 2 force constraints:

a. $| \widetilde{\vartheta\omega}(g) | \leq \pi$

b. $\min_{\mathbf{r} \in P} f_T(\mathbf{r}) \geq \delta_c$

c. $\theta_{i,j}^{min} \leq \theta_{i,j} \leq \theta_{i,j}^{max}$; $i = 1, 2, 3, 4$; $j = 1, 2, 3$

d. The forces are computed as mentioned in *section 5.2*. The summations are performed over $i \neq i_f$. The first set of force inequalities specify that

none of the normal component of forces at the relevant foot-tips should be negative. Thus,

$$\mathbf{F}_i \cdot \mathbf{n}_i > 0,\ i \neq i_f$$

e. The second set of force inequalities specify that the force system at the foot-tips should lie within the friction cone. Thus,

$$\mid \mathbf{F}_i - (\mathbf{F}_i \cdot \mathbf{n}_i)\mathbf{n}_i \mid < \mu(\mathbf{F}_i \cdot \mathbf{n}_i),\ i \neq i_f$$

However, under certain circumstances we may not want to impose the last two inequality constraint, i.e. the force inequality constraints. The simple reason being the fact that the force system solved here is just a particular solution and does not represent the actual forces. Hence whenever we don't want to impose inequalities *d.* or *e.*, we will mention that while using this algorithm. However we will include them unless explicitly mentioned not to do that.

Optimization objectives: In addition to the optimization objectives mentioned in the previous section, we have one additional objective to keep the center of mass as close to $\mathbf{r}_c$ as possible. Thus the objectives are,

a. We define reference forward direction $\mathbf{u}_f = \frac{\partial \mathbf{R}_\tau}{\partial \lambda}\big|_{\lambda_{\mathbf{r}_c^*}}$ (where $\lambda_{\mathbf{r}_c^*}$ represents a point on the trajectory τ *close* to the point $\mathbf{r}_c^*$) and reference normal direction $\mathbf{u}_n = \nabla_s f_T|_{\mathbf{r}_c^*}$. Then,

$$\sigma_1 = {}^G\mathbf{u}_f \ \cdot\ \tilde{R}(g) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\sigma_2 = {}^G\mathbf{u}_n \ \cdot\ \tilde{R}(g) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

b. $\theta_{i,j}$ are obtained from Ψ using inverse kinematics. We define,

$$\sigma_3 = \frac{1}{1 + \|(\theta - \theta^{nor})K\|}$$

c. We define the objective to be *maximized* that will keep the center of mass of the robot as close as possible to $\mathbf{r}_c^*$,

$$\sigma_4 = \frac{1}{1 + |\tilde{\mathbf{v}}(g) - \mathbf{r}_c^*|}$$

d. And finally we define an objective that will ensure that the forces at the foot-tips remain well within the friction cone. Thus we wish to maximize

$$\sigma_5 = \min_{i \neq i_f} \left(\frac{\mathbf{F}_i \cdot \mathbf{n}_i}{|\mathbf{F}_i|} \right)$$

However it can be noted here that since the forces that we calculated are just a particular solution, at times we may not wish to use this objective.

Thus we define the final combined objective to be *minimized*,

$$\sigma = -{\sigma_1}^{\gamma_1}{\sigma_2}^{\gamma_2}{\sigma_3}^{\gamma_3}{\sigma_4}^{\gamma_4}{\sigma_5}^{\gamma_5}$$

where, $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ and γ_5 are suitable weight factors.

7.4 Finding a point in a new leaf of $\mathbb{S}_{i_f} \cap \mathbb{S}_0$, $i_f \neq 0$ searching along a leaf of $\mathbb{S}_{i_f}$ — the *foot placement* algorithm:

Once again this is a nonlinear, non-convex optimization problem with well defined inequality & equality constraints and search space, and some freedom in choosing the optimization objectives. Here the physical picture is we are given three footholds for the robot on which the weight of the robot is distributed, and the g^* (the '*' denotes that it is a given/fixed parameter for this problem) defining the position & orientation of robot body is also given. So now we have the freedom to move the other leg. Our goal will be to choose a foothold for this leg on the terrain surface, while keeping g^* and other footholds fixed. Thus if the index of the foot that is to be placed is i_f, then we are essentially searching for a suitable point in $\mathbb{S}_0 \cap \mathbb{S}_{i_f}$ lying on the particular leaf of $\mathbb{S}_{i_f}$ defined by the other 3 footholds. But since we have fixed g, our search space will be a sub-manifold of the leaf. Essentially our search space will be a 3 dimensional sub-manifold of the leaf. Thus, if $\Psi = \{g, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4\}$ defines a particular point in the configuration space, the optimization problem can defined as:

Given/constants: The trajectory τ that is to be followed, along with its parametrization, $\mathbf{R}_\tau(\lambda)$ is given. $g = g^*$ is given, and we denote the different components derivable from g^* by $\mathbf{r}_c^* = \tilde{\mathbf{v}}(g^*)$, $R^* = \tilde{R}(g^*)$, etc. And three of the footholds are given as $\mathbf{r}_i = \mathbf{r}_i^*$, $i \neq i_f$. Hence this optimization problem can once again be represented completely as,

$$\Psi_{optimum} = \Theta_3(g^*, \tau, \{\mathbf{r}_i^*\}_{i \neq i_f}, i_f) \tag{9}$$

Search space: The search space is simply the components of the vector $\mathbf{r}_{i_f}$. That means we are searching only for a foothold. It can be noted that the boundaries of the search space, $\mathbf{r}_{i_f}$ is rather complex. To simplify the problem we can instead choose our search space to be $\theta_{i_f,j}$, $j = 1, 2, 3$. Although these two spaces are not homeomorphic, we can use either of them without much difficulty. Since it is easier to define the boundaries in the θ space, we choose $\theta_{i_f,j}$, $j = 1, 2, 3$ as our search space. The transformation to the $\mathbf{r}_{i_f}$ space can be obtained using the Direct/Forward Kinematics described in *section 5.1.1.*

Bounds on search space: The search space $\theta_{i_f,j}$, $j = 1, 2, 3$ has very simple linear boundaries,

$$\theta_{i_f,j}^{min} \leq \theta_{i_f,j} \leq \theta_{i_f,j}^{max}; \quad j = 1, 2, 3$$

Equality constraints: Since the i_f^{th} foot should touch the terrain surface, we have the following equality constraint,

$$f_T(\mathbf{r}_{i_f}) = 0$$

Inequality constraints: Essentially there isn't any, since we have already defined the bounds on our search space. However for ensuring that the foothold is not chosen at a place which is very *steep*, we impose a limit on the slope of the part of the terrain surface where the foot is to be placed. We define $\cos\bar{\theta} = \mid \hat{\mathbf{g}} \cdot \mathbf{n}_{i_f} \mid$,

where $\hat{\mathbf{g}}$ is an unit vector along $\mathbf{g}$. Thus we impose the inequality constraint,

$$\tan\bar{\theta} \leq \mu$$

Optimization objectives: Presently we have 3 different objectives:

a. We define a *target foothold* for the i_f foot that is to be placed. Let the position vector of the target foothold be $\mathbf{r}_{i_f}^t$. There may be different ways for finding a $\mathbf{r}_{i_f}^t$. Presently we have a very crude way of defining a target foothold. *Section 8.2.2* gives a particular method of finding this $\mathbf{r}_{i_f}^t$. Thus now we define the first objective to be maximized,

$$\sigma_1 = h_m\left(\left|M(\mathbf{r}_{i_f} - \mathbf{r}_{i_f}^t)\right|\right)$$

where M is a scaling tensor which may be a function of $(\mathbf{r}_{i_f} - \mathbf{r}_{i_f}^t)$, and $h_m : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ is a monotonically decreasing function. Presently we are using a heuristic approach for choosing M and h_m, the details about which will not be discussed right now.

b. It is preferable to place the foot on a *flat* part of the terrain. By *flat* we mean that the normal to the surface should remain aligned with $-\mathbf{g}$. Thus we add the following objective to be maximized,

$$\sigma_2 = 1 - \hat{\mathbf{g}} \cdot \mathbf{n}_{i_f}$$

where $\hat{\mathbf{g}}$ is basically an unit vector along the direction of $\mathbf{g}$.

c. And finally we have an objective to be maximized that helps to keep the joint angles close to the *normal* configuration joint angles,

$$\sigma_3 = \frac{1}{1 + \left|(\theta_{i_f} - \theta_{i_f}^{nor})K\right|}$$

where, θ_{i_f} and $\theta_{i_f}^{nor}$ represents the vector whose j^{th} elements are $\theta_{i_f,j}$ and $\theta_{i_f,j}^{nor}$ respectively. K is again a 3×3 scaling matrix for which we are presently using the identity matrix.

Thus we define the final combined objective to be *minimized*,

$$\sigma = -{\sigma_1}^{\gamma_1}{\sigma_2}^{\gamma_2}{\sigma_3}^{\gamma_3}$$

where, γ_1, γ_2 and γ_3 are suitable weight factors.

7.5 Initial placement:

This algorithm is not required for the case of actual experimentation with a quadruped robot. However in simulation we need to perform this extra step to make sure that the robot is initially placed on one of the allowed strata. Thus essentially we need to search in the 18 dimensional configuration space to find an initial placement for the robot. Thus this is an extremely computationally expensive step that needs to be performed at the beginning of a *local planning* process. We can however reduce the dimensionality of the search space by

specifying a $\mathbf{r}_c$ close enough to the terrain surface such that a feasible solution can be found. As before, let $\Psi = \{g, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4\}$ defines a particular point in the configuration space. Then,

Given/constants: $\tilde{\mathbf{v}}(g) = \mathbf{r}_c^*$ is given. Moreover the trajectory τ along with its parametrization is known, $\mathbf{R}_\tau(\lambda)$. But none of the footholds are known. Hence this optimization problem can be represented completely by the following notation:

$$\Psi_{optimum} = \Theta_4(\mathbf{r}_c^*, \tau) \tag{10}$$

Search space: The search space consists of the elements of the 3×1 vector $\widetilde{\vartheta\omega}(g)$ and the joint angles of the feet $\theta_{i,j}$; $i = 1, 2, 3, 4$; $j = 1, 2, 3$.

Bounds on search space: Since we keep the twist angle corresponding to g between $-\pi$ and $+\pi$, we have $\widetilde{\vartheta\omega}(g) \in [-\pi, \pi]$. Moreover, $\theta_{i,j}^{min} \leq \theta_{i,j} \leq \theta_{i,j}^{max}$; $i = 1, 2, 3, 4$; $j = 1, 2, 3$.

Equality constraints: Since the foot-tips should touch the terrain surface, we impose the equality constraints $f_T(\mathbf{r}_i) = 0$, $i = 1, 2, 3, 4$. The $\mathbf{r}_i$ are computed using Forward Kinematics.

Inequality constraints: Following are the sets of inequality constraints:

a. $\mid \widetilde{\vartheta\omega}(g) \mid \leq \pi$

b. $\min_{\mathbf{r} \in P} f_T(\mathbf{r}) \geq \delta_c$

c. $\theta_{i,j}^{min} \leq \theta_{i,j} \leq \theta_{i,j}^{max}$; $i = 1, 2, 3, 4$; $j = 1, 2, 3$

d. $\mathbf{F}_i \cdot \mathbf{n}_i > 0$; $i = 1, 2, 3, 4$

e. $\mid \mathbf{F}_i - (\mathbf{F}_i \cdot \mathbf{n}_i)\mathbf{n}_i \mid < \mu(\mathbf{F}_i \cdot \mathbf{n}_i)$; $i = 1, 2, 3, 4$

Note that here we consider all the four feet for computation of the forces. That means the $\sum_i$ used under *section 5.2* for the computation of forces run from $i = 1$ to 4. This assumption may be satisfactory for the purpose of initial placement. However as before, we may choose not to impose the force inequality constraints.

Optimization objectives: The following are a few objective functions that we have used for this algorithm:

a. As before, we define reference forward direction $\mathbf{u}_f = \frac{\partial \mathbf{R}_\tau}{\partial \lambda}\big|_{\mathbf{r}_c^*}$ and reference normal direction ${}^G\mathbf{u}_n = \nabla f_T|_{\mathbf{r}_c^*}$. Then we define the following two objective functions to be maximized,

$$\sigma_1 = {}^G\mathbf{u}_f \cdot \tilde{R}(g) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\sigma_2 = \mathbf{u}_n \cdot \tilde{R}(g) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

b. For keeping the joint angles close to the *normal* values, we define the following objective to be maximized,

$$\sigma_3 = \frac{1}{1 + \|(\theta - \theta^{nor})K\|}$$

where, θ and θ^{nor} are 4×3 matrices whose elements are $\theta_{i,j}$ and $\theta^{nor}_{i,j}$ respectively; K is a 3×3 scaling matrix which we chose to be the identity matrix; and $\| \cdot \|$ is a suitable matrix norm.

Thus we define the final combined objective to be *minimized*,

$$\sigma = -{\sigma_1}^{\gamma_1}{\sigma_2}^{\gamma_2}{\sigma_3}^{\gamma_3}$$

where, γ_1, γ_2 and γ_3 are suitable weight factors.

7.6 Algorithm for searching along a trajectory:

This algorithm essentially uses the algorithms in *section 7.2* or *section 7.3* to perform a search by varying the values of $\mathbf{r}_c$ that those two algorithms need as *given parameters*. This problem may be represented as,

$$\begin{aligned} \Xi &= \{(\Psi, \lambda) \mid \Psi = \Theta_k(\mathbf{R}_\tau(\lambda), \tau, \mathbf{r}_1^*, \mathbf{r}_2^*, \mathbf{r}_3^*, \mathbf{r}_4^*, i_f) \text{ returns a feasible solution } \} \\ &= \Omega_k\left(\tau, \mathbf{r}_1^*, \mathbf{r}_2^*, \mathbf{r}_3^*, \mathbf{r}_4^*, i_f\right) \end{aligned} \tag{11}$$

We use a binary search technique to perform this search operation. Here $k = 1$ or 2 depending on whether we are using a *hard* or a *soft* persuasion method. Hence basically we need to alter the values of λ and determine which points on the trajectory are attainable by the center of mass of the robot. However it can be noted that for $k = 2$, i.e. the *soft* persuasion method, the value of λ will not determine the existence of a feasible solution. Hence essentially the use of this search algorithm will make sense only if we are making use of the *hard* persuasion method, i.e. $k = 1$. Thus we will always assume $k = 1$ unless explicitly told to do otherwise.

8 Algorithms for *local planning*:

In the following subsections we will discuss a few algorithms that we had been using for the purpose of planning the motion along a given trajectory in the physical space of the robot. All the algorithm are primarily based on a heuristic approach. The basic strategy behind designing these algorithms is that during the motion we mostly try to move along leaves of $\mathbb{S}_0$. When on a leaf of $\mathbb{S}_0$ we try to move such that the distance from the *goal* Ψ is reduced as much as possible. Once it is no more possible to move along a particular leaf of $\mathbb{S}_0$ because of presence of discontinuities, we shift to another leaf via a leaf of $\mathbb{S}_i$, $i \neq 0$. We make this transition as quick and minimal as possible, however always trying to move along a direction which takes the robot closer to the *goal* Ψ. A typical snippet of a path in the lower dimensional representation of the configuration manifold (as discussed under *section 1.1*) may look something as below:

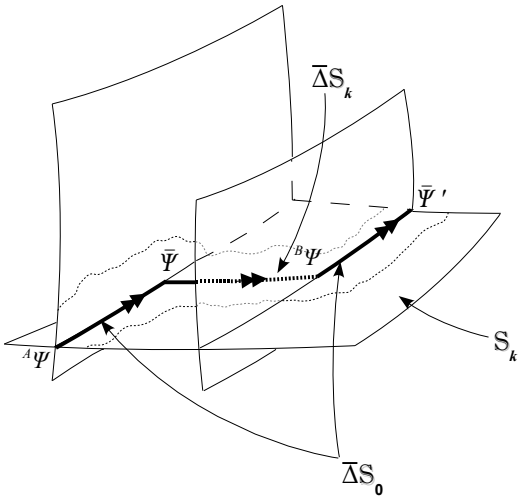


Fig. 6: A snippet of a typical path in the configuration manifold - a reduced dimensional representation

The above figure shows the transition from one leaf of $\mathbb{S}_0$ to another via a leaf of $\mathbb{S}_k$, $k \neq 0$. We try to keep the part $\bar{\Psi} - {}^B\Psi$ as short and simple as possible, but the primary motion takes place in this part. On the other hand the portions ${}^A\Psi - \bar{\Psi}$ and ${}^B\Psi - \bar{\Psi}'$ are essential to reach one leaf of $\mathbb{S}_{i_1}$ from one of $\mathbb{S}_{i_2}$, but primary motions don't take place along them. The dotted boundaries show the discontinuities in the strata. Hence we need to take care that the robot stays inside these boundaries.

There are primarily two approaches that we have used in planning a path as mentioned above.

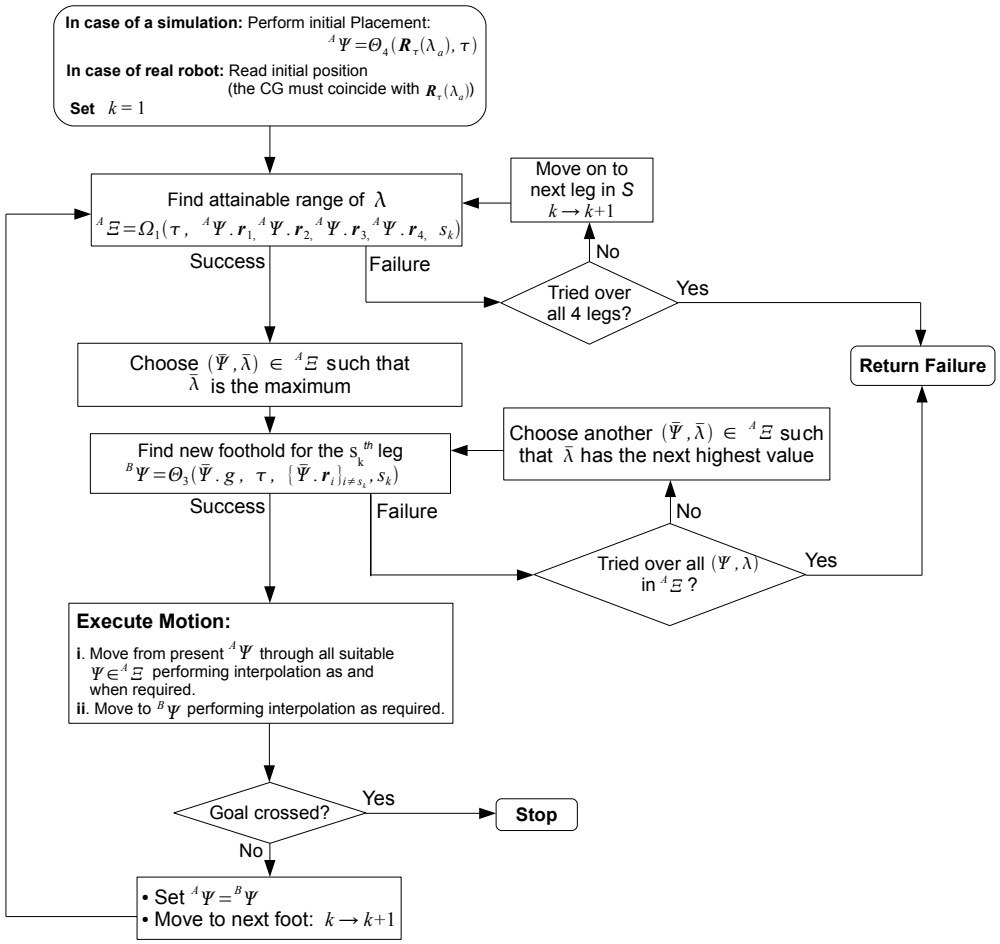
i. The first approach as described under *section 8.1* uses a binary search technique to find points on the leaves of $\mathbb{S}_0$.

ii. Whereas in the second approach we find only a few distinct key points in the manifold (like the points $\bar{\Psi}$ or ${}^B\Psi$ in *figure 6*) and then rely completely on the interpolation algorithm (described under *section 7.1*) to find the trajectories.

8.1 An algorithm for *local planning* using search in $\mathbb{S}_0$ and the *hard persuasion* strategy:

This is the most basic and primitive algorithm that we have used for the local planning. In this algorithm we make use of the search procedure that we defined under *section 7.6*. The flowchart below gives the overall idea about the algorithm. There may be a few finer aspects that are not reflected in the flowchart but has been implemented in practice.

The trajectory τ along with its parametrization is given. Points on the trajectory are given by $\mathbf{R}_\tau(\lambda)$, where $\lambda \in [\lambda_a, \lambda_b]$. A gait step sequence is given, $S = \{s_1, s_2, \cdots\}$.



8.2 An algorithm for *local planning* using interpolation and the *soft persuasion* strategy:

The basic concept behind this algorithm is that we primarily rely on the interpolation procedure of *section 7.1* for planning the motion. We just find a few key points in the manifold that we must pass through, and let the interpolation algorithm take care of constructing the path.

In addition to this we make a modification on the algorithm in the previous section (*8.1*) by introducing a notion of *gait step queue*. This helps us in dealing with failures of Θ_1 or Θ_2 in a more efficient manner. A *gait step queue* is a list if the indices of feet, quite similar to *gait step sequence (S)* mentioned in the previous section. The only difference being that this is not a fixed list and the items in the list change dynamically during the runtime. In the queue we have the notions of pushing, popping and swapping of elements.

Before we present the flowchart we would like to discuss two short algorithms for calculating the $\mathbf{r}_c^*$ used in the algorithm of *section 7.3*, and for calculating $\mathbf{r}_{i_f}^t$ used in the algorithm of *section 7.4*.

8.2.1 Method for finding a suitable $\mathbf{r}_c^*$ for using in Θ_2 (algorithm of *section 7.3*):

An evident choice for $\mathbf{r}_c^*$ will be an arbitrary point ahead on the trajectory, so that the robot's center of mass moves as much forward as possible. However there are certain reasons because of which this many not be what we would like to have in the actual experiments. In our model we make assumption of a quasistatic system, which in real experiments is not true even when we take much care. This results in changes in the boundaries/discontinuities in the workspace induced by the force inequalities. Moreover we need to account for errors and inexact measurements in the actual experiments. Thus to ensure that the robot is sufficiently away from these uncertain workspace boundaries, we need to choose $\mathbf{r}_c^*$ accordingly. It is to be noted here that since our model cannot predict how the boundaries will behave, we can't have a proper way of determining the desired $\mathbf{r}_c^*$. Thus we use a rather heuristic approach.
Given a Ψ and an index i_f of a foot, we find the $\mathbf{r}_c^*$ as,

$$\mathbf{r}_c^* = \Lambda_c(\Psi, i_f) \tag{12}$$

Where Λ_c is essentially a 2-step algorithm. In the the first step we compute $\mathbf{r}_c^+ = \sum_{i \neq i_f} \Psi.\mathbf{r}_i$. In the second step we *lift up* the point from the terrain surface to a distance of d_c from the terrain surface. In case of a *simple terrain* (a terrain that can be represented with the global x and y coordinates as parameters and $z = z_T(x, y)$) this is done rather trivially. We just increase the z coordinate of ${}^G\mathbf{r}_c^+$ by a value of d_c and hence obtain $\mathbf{r}_c^*$.

8.2.2 Method for finding a suitable $\mathbf{r}_{i_f}^t$ for using in Θ_3 (algorithm of *section 7.4*):

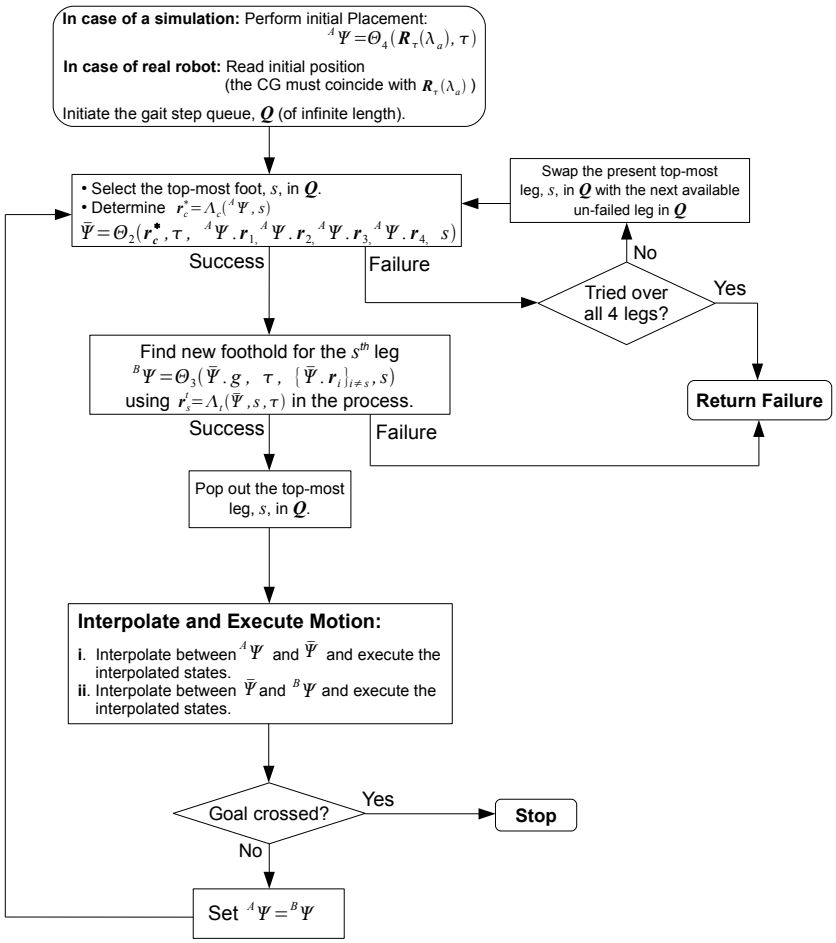
As mentioned under section 7.4, $\mathbf{r}_{i_f}^t$ is a target foothold for the foot placement procedure. The details of this method will not be discussed here. But the basic concept is that, given a Ψ and the i_f, we find a point on the trajectory τ which is close to $\Psi.g.\mathbf{v}$. Then we compute the *normal position* of the foot-tip of leg i_f

had the robot's center of mass coincided with that point on the trajectory. (By *normal position* we mean the configuration when the joint angles are all equal to $\theta_{i,j}^{nor}$, i.e. the normal configuration.) Then we just choose $\mathbf{r}_{i_f}^t$ to be a point ahead of this *normal position* of i_f foot-tip in the direction of the trajectory τ. This problem may be represented as

$$\mathbf{r}_{i_f}^t = \Lambda_t(\Psi, i_f, \tau) \tag{13}$$

8.2.3 The flowchart:

As before, the trajectory τ along with its parametrization is given. Points on the trajectory are given by $\mathbf{R}_\tau(\lambda)$, where $\lambda \in [\lambda_a, \lambda_b]$. A gait step queue, Q, is given.



Comments on this algorithm:

i. Since we use the *soft persuasion* procedure, this algorithm has greater probability of success compared to the one in section 8.1.

ii. Using the notion of a gait step queue enables us to keep trying with a particular leg again and again in case of failure. This is contrary to the algorithm of section 8.1 where we had to wait for one complete gait cycle to retry a leg that failed in the previous cycle.

iii. Although we don't use a failure handling procedure for the foot placement step, Θ_3, we can note that the probability of failure of Θ_3 is quite low. However an absence of failure handling procedure definitely makes it less robust.

iv. Since the calculation of $\mathbf{r}_c^*$ does not involve the information about the trajectory, basically we don't perform any significant motion towards our goal using Θ_2. The main movement towards goal is performed using Θ_3, i.e. the foot placement procedure.

8.3 Creating a feedback system:

We have recently implemented a feedback step in our algorithm in order to account for the deviations of the actual robot from our model due to the quasi-static assumption and also to account for the errors that creep into the system. For the actual experiment we make use of the MOCAP system [6] to obtain the state information of the real robot. Then we just add a simple additional step in the main loop of the flowchart of *section 8.2.3* to update the ${}^A\Psi$. In fact in place of the step *'Set* ${}^A\Psi = {}^B\Psi$*'* we use a step to read ${}^A\Psi$, the state of the actual robot.

The study on stability or convergence of this feedback procedure is beyond the scope of the present work. However if the errors and deviations are small, we note that the ${}^B\Psi$ at the end of a particular cycle will be almost same as the state of the robot that is being read into ${}^A\Psi$. Thus even with the feedback the algorithm will work similar to the working of the algorithm of section 8.2.3 on our constructed model.

8.4 An algorithm involving pre-computation and feedback correction:

The optimization processes using MATLAB's *fmincon* is extremely computationally expensive. While it is possible to run all the previously mentioned algorithms in real time on the robot, the slow movement of the robot may not be very presentable. Hence we are devising a method by which the states of the robot for moving along a given trajectory will be precomputed in a simulation, and then we'll execute all the states all together on the actual robot. But to account for the errors that may cause deviation of the robot from the simulated states we perform small corrections based on the feedback from the MOCAP [6] system. We are in the process of development of this method.

8.5 A robust algorithm for *local planning* using recursive techniques for dealing with failure situations:

We haven't yet implemented this algorithm, but a supposed high potential of this algorithm makes it worth of discussion.

As evident from the algorithm of *section 8.1*, it is capable of dealing with a few situations where Ω_1 or Θ_3 returns failure. Similarly the algorithm of *section 8.2.3* is capable of dealing with failures of Θ_2. Under such failure situations it just tries to move on to the next foot or trace back in the list of Ψ's that the search algorithm returned.

However this method of dealing with failure is not robust and there may be situations when the algorithm returns total failure. Hence we need a technique that will be able to trace back the steps as much required when the algorithm encounters a failure. It should have the ability to trace back as many times as required till it finds a feasible solution for the step where it got stuck.

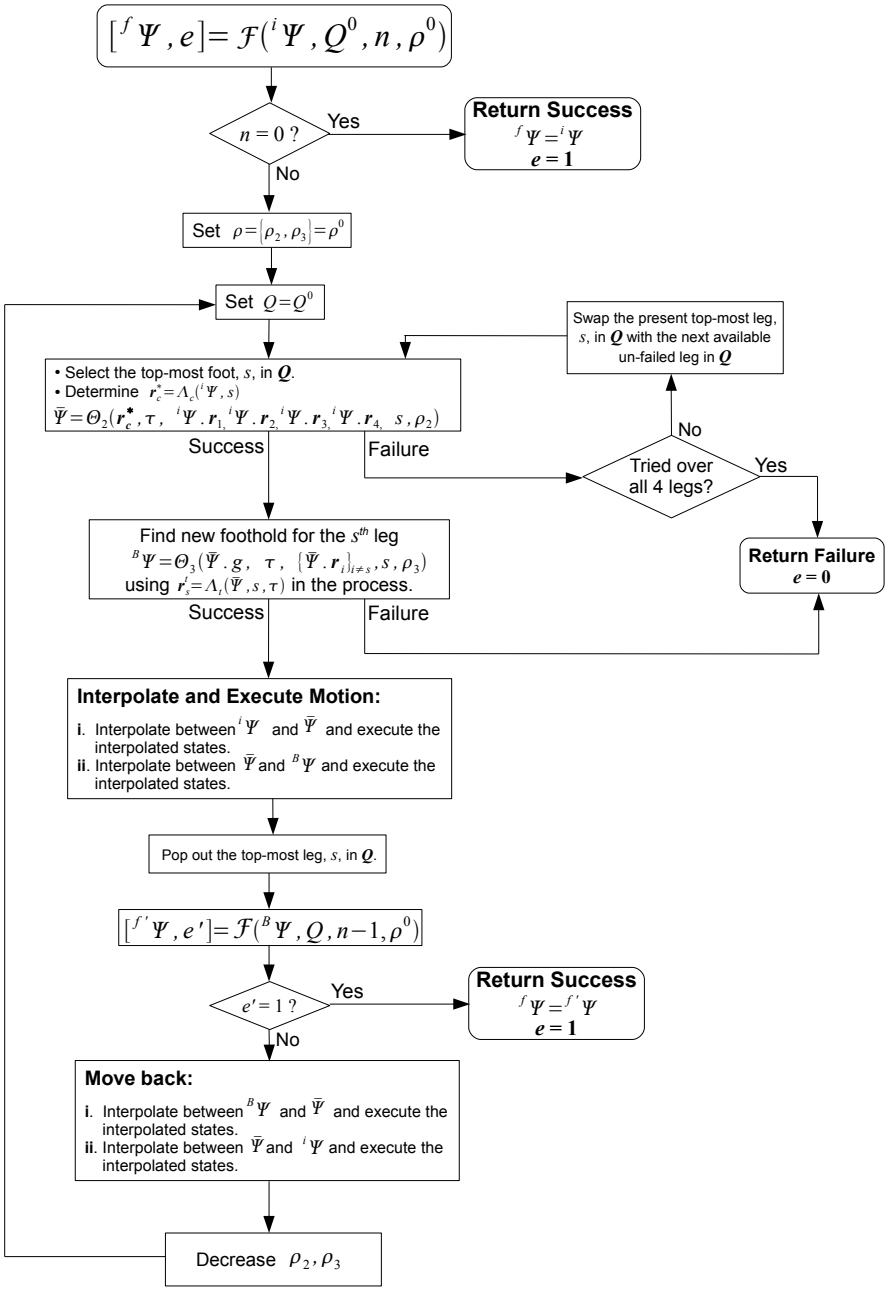
The implementation of such an algorithm is possible efficiently only using a recursive technique. We haven't yet constructed a formal scheme for such an algorithm, but have the basic outline. We'll discuss in brief about the outline using a schematic flowchart. The details of the algorithm are yet to be developed.

For using in this algorithm, we modify the optimization problems Θ_2 and Θ_3 slightly by adding an extra parameter to the input arguments of these problems. Thus the optimization problems in equations (8) and (9) are now represented as $\Theta_2(\cdots, \rho_2)$ and $\Theta_3(\cdots, \rho_3)$, where ρ_2 and ρ_3 are the parameters and can assume positive values. The parameters will possibly be used in the objective functions of the respective problems so that altering the parameter will result in different solution to the problem. However that should not change the state of existence of a feasible solution. A high value of ρ_i, $i = 2, 3$ will mean very ambitious movements, while a low value will imply short and minimal movement.

We define the recursive algorithm in a functional form,

$$[{}^{f}\Psi, e] = \mathcal{F}({}^{i}\Psi, Q, n, \rho) \tag{14}$$

This function is supposed to start with the robot state ${}^{i}\Psi$ and move for n steps (by a *step* we mean the combined process of body movement Θ_2 and foot placement Θ_3 in a particular cycle of our algorithm), executing the states all within itself, and finally return the final state of the robot, ${}^{f}\Psi$. A gait step queue, Q, is also supplied. $\rho = \{\rho_2, \rho_3\}$ is the parameter vector that is supplied. On success it will return $e = 1$, else will return $e = 0$. Moreover there is a given trajectory τ, the information about which is available globally. The schematic structure of $\mathcal{F}$ is given in the next page.



Discussions:

i. Quite evidently this is a high computationally expensive process, like any other recursive processes, specially if too many failures are encountered.

ii. Being a recursive algorithm it has the ability of *tracing back* steps and hence is more robust towards failure situations.

9 The *global planning* — planning an optimum trajectory:

We haven't yet implemented this part of the algorithm. However once the *local planning* is completed successfully, the *global planning* will be a comparatively easier issue. We can always calculate the cost associated with a given trajectory. using some suitable criteria. Let us denote the cost associated with a trajectory τ by $\mathcal{C}(\tau)$. The exact functional form of $\mathcal{C}$ is yet to be worked out.

Let's define a particular trajectory τ_0 that connects the initial point to the goal point in the physical space of the robot. Let its parametrization be given by $\mathbf{R}_{\tau_0}(\lambda)$, $\lambda \in [\lambda_a, \lambda_b]$. Then any arbitrary trajectory τ, passing through the initial and goal points may be represented by the following parametrization,

$$ {}^{G}\mathbf{R}_{\tau}(\lambda) = {}^{G}\mathbf{R}_{\tau_0}(\lambda) + \sum_{p=1}^{\infty} \left(\sin\left(\frac{p\pi(\lambda - \lambda_a)}{\lambda_b - \lambda_a} \right) \begin{bmatrix} A_{p1} \\ A_{p2} \\ A_{p3} \end{bmatrix} \right) \tag{15} $$

Then our problem of global planning reduces to the following optimization problem,

$$ \mathcal{C}(\tau_{optimum}) = \min_{A_{pq} \in \mathbb{R}} \mathcal{C}(\tau) \tag{16} $$

Thus we basically search over different trajectories to find the one whose cost is the minimum. As quite evident, this problem will be extremely computationally expensive and will not be feasible in real time. However we may pre-compute the optimum trajectory and make the robot follow it after that.

10 Simulations and Experiments:

We have implemented the algorithms in MATLAB and performed simulations on an exact model of the Little Dog robot [6]. Below is a screenshot of our simulated model.

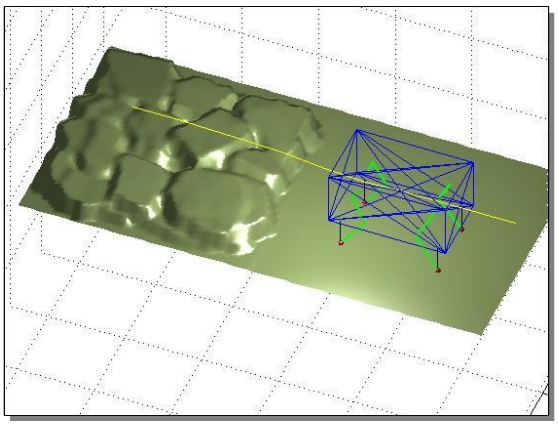


Fig. 7: Simulated model of the Little Dog Robot in MATLAB

We have started working with the actual Little Dog Robot and have made it walk on flat surfaces using our algorithms with satisfactory success. Presently

we are working to make it walk on rough terrains. We have implemented the feedback correction system discussed under *section 8.4* successfully. We have also attained some success in making the robot walk on one of the rough terrains. However we still need to do quite a lot of work to make the performance of the feedback correction system better and more robust.
The latest videos and updates about the works in progress can be found at http://www.seas.upenn.edu/∼subhrabh/nonWebsite/LittleDog/index.html.

11 Conclusions — Achievements, Drawbacks and Future works:

We have so far done a systematic study of the stratified configuration manifold of a quadruped robot. We have defined the manifold including its strata and their foliations completely using proper notations, equations and inequalities. Moreover we have designed a scheme for planning motion along the leaves of the foliations of the strata.

One major drawback of the optimization algorithms is that they are extremely computationally expensive. Although the local planning may be possible to be implemented in real time using powerful processors, the global planning can not be done in real time with the present processing power of available computers. Moreover as mentioned in the previous section, our assumption of quasistatic model is quite idealistic. To make the algorithm work better we need to incorporate the dynamic terms in the equations.

Hence at present we have quite a few tasks in hand that we would like to implement in the recent future. Below is a short list of them:

i. Complete and implement the algorithms mentioned under *section 8.4* and *section 8.5*.

ii. Gain more success in making the robot walk on rough terrains in actual experiments.

iii. Perform the *global planning* as described under *section 9*.

iv. Include the dynamic terms in the force equations so that we no longer restrict ourselves to the quasistatic model. This will essentially increase the dimensionality our search space (since now the velocities will also be included as state variables) and will induce more constraints in our search processes.

12 Acknowledgement:

I would like to thank Prof. Vijay Kumar, School of Engineering & Applied Sciences, GRASP Laboratory, University of Pennsylvania, for his valuable guidance and advices which has made this work possible. I would also like to thank Dr. Sachin Chitta, GRASP Laboratory, University of Pennsylvania, for his kind help and advices.

13 References

[1] S.Hirose, *A Study of Design and Control of a Quadruped Walking Vehicle*, The International Journal of Robotics Research, vol. 3, no. 2, pp. 113-120 (1984).

[2] V.R.Kumar, K.J.Waldron, *Force Distribution in Closed Kinematic Chains*, IEEE Journal of Robotics and Automation, vol. 4, no. 6, pp. 657-664 (1988).

[3] V.Kumar, K.J.Waldron, *Force Distribution in Walking Vehicles*, Transaction of the ASME Journal of Mechanical design, vol. 112, pp. 90-99 (1990).

[4] Sachin Chitta, *Dynamics and Control of Modular Locomotion Systems*, Doctoral Thesis, University of Pennsylvania (1999-2004).

[5] John William Goodwine, Jr., *Control of Stratified Systems with Robotic Applications*, Doctoral Thesis, California Institute of Technology (1997).

[6] *LittleDog Robot 1.0 User Guide*, Boston Dynamics (2006).

[7] Richard M. Murray, Zexiang Li, S. Shankar Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press LLC (1994).