PATH PLANNING FOR VISUAL SERVOING AND NAVIGATION USING CONVEX OPTIMIZATION

Abdul Hafez Abdul Hafez,[*] Anil K. Nelakanti,[**] and C.V. Jawahar[***]

Abstract

Exploiting visual cues to control systems for robotic applications is a promising idea. Practically, they are usable only if the end result accedes to the restrictions of the specific environment. These restrictions are like the limited field of view of the camera and physical constraints of the workspace of the robot. Hence, there is a need for a general framework that can be used adaptively across various environments. We develop such an algorithmic framework that is flexible to accommodate various kinds of constraints and generate a solution that is optimal in the sense of the considered error measure. We perform a constrained optimization on the error in a convex domain considering all the necessary constraints using convex optimization techniques and further extend it to non-convex domains. We utilize branch-and-bound algorithm to divide the problem of optimizing over a range of rotations into simpler problems and solve for the optimal rotation. We demonstrate the performance of the algorithm by generating control signals in a simulated framework for visual servoing and in a real-world for robot navigation.

Key Words

Visual control, convex optimization, robotic vision

1. Introduction

Vision-based control for robotics draws inferences from visual cues in the environment to generate appropriate control signals for robot motion [1]. The goal is to generate control signals while taking into consideration various possible restrictions or constraints that are introduced by the environment. Restrictions could be in the form of mechanical constraints of the robot, physical constraints by various obstacles in the workspace, limitations of the sensor like the field of view of the camera, limitations of the designed control algorithm like local minima problem or view planning for active reconstruction. The solution to this problem is visual path planning strategy [2]. This solution is a path that satisfies all the necessary conditions instead of designing the control algorithm [3], [4]. This path is the optimal path with respect to the given restrictions and objectives. This paper aims at achieving this goal of finding the optimal path through the constrained region using convex optimization techniques.

Recently, Kazemi *et al.* proposed in [3], [5] a planner that explores the camera state space (*i.e.*, a space of camera poses and velocities) for permissible trajectories by iteratively extending a search tree in this space and simultaneously tracking these trajectories in the robot configuration space (*i.e.*, joint space). The planned camera trajectories are then projected into the image space to obtain desired feature trajectories. Searching for global path planning methods that satisfy the given constraints of the sensor, robot and environment has recently motivated the utilization of convex optimization in visual path planning methods.

Optimization techniques aimed at finding the optimal path with respect to various cost functions and constraints such as distance from the image boundary, straightness and length of the camera/robot path and smoothness of the path and energy [2]. Most of them are not suitable for real-time path planning and do support the representation of only a few constraints [6]. Moreover, most optimization techniques are local that lead to incomplete path [4]. The most common path planning optimization method is the projected gradient (potential field method) [6]. It suffers from the local minima problem when the repulsive and attractive filed are equals. In contrast, our method is formulated as a convex optimization problem. Hence, it ensures the globality of the solution and supports the representation of unlimited number of constraints as well.

The initial attempt to the visual motion planning as a convex optimization problem is presented in [7], [8]. However, these methods return a path that satisfies a few constraints such as joint limits and workspace but not necessarily the shortest possible one. In this paper, we use convex optimization to plan the path that ensures the global optimality, *i.e.*, the shortest possible path for which various constraints are met. We build further on [7],

[*] Computer Engineering Department, Hasan Kalyoncu University, Sahinbey, Gaziantep, Turkey; e-mail: abdul.hafez@hku.edu.tr
[**] Indian Institute of Technology (BHU), Varanasi, India; e-mail: anil.nelakanti.cse@iitbhu.ac.in
[***] International Institute of Information Technology IIIT-Hyderabad, India; e-mail: jawahar@iiit.ac.in

which attempted to perform path planning as a convex optimization process where the whole path was planned by optimizing in the translation space.

Convex optimization [9], most importantly, does not have the pitfall of local minima assuring that the achieved solution always happens to be the most optimal. The advantages of optimality and efficiency make the convex optimization highly preferable when compared to the traditional methods of modelling and optimization. Application of convex optimization for computer vision has been extensively explored and surveyed in [10]. A wide variety of vision problems have been reformulated as convex optimization problems and solved efficiently [11], [12]. In this work, we follow the approach of [11], [12] in using the branch-and-bound algorithm to search in solution space for the most appropriate rotation within a tolerance limit. Optimality of the solution is also guaranteed following the proof from [11], [12] (also see [9]). The proof is outlined in Section 3.

The contribution of this paper is a convex optimization-based visual path planning algorithm that produces an optimal camera and image trajectory subject to sets of visibility and workspace constraints with application to robot visual navigation and control. The translational part of the camera pose is optimized by directly solving a convex optimization problem. The rotational part is optimized by solving a set of convex feasibility problems with the help of a branch-and-bound search algorithm running over the rotation space. In this work, (i) we show a general framework for path planning involving multiple constraints, both in the image space and physical workspace, (ii) we specifically extend the work of [7] to optimize for poses along the path over the set of rotations and (iii) we evaluate and demonstrate our proposed method on visual servoing and robot navigation tasks with camera visibility constraints.

2. Problem Formulation and Modelling

Path planning comprises the computation of the intermediate poses between the initial and desired poses. Each of these intermediate camera poses is calculated by an optimization process subject to necessary constraints. The end result should be a path through the constrained region from the initial to the destination pose that is optimal in the sense of considered error measure. Figure 1 depicts the planning strategy followed in this paper, which is mathematically formulated below.

Let the points in the feature set to be used as cues be denoted by M, which are tracked along the path by the camera. The camera pose vector $\mathcal{X}_i$ describes both the position and the orientation of the camera. The pose vector $\mathcal{X}_i = (r_i \;\; t_i)^T$ is a 6×1 vector where $t_{i(3\times1)}$ gives the translation and $r_{i(3\times1)} = (\alpha_i, \beta_i, \gamma_i)^T$ gives the rotation, with α being roll, β being pitch and γ being yaw of the camera. Let the initial camera frame be F_0 and the destination camera frame F_* and F_i any arbitrary intermediate camera frame on the path of the camera.

We aim at constructing a path, which is in the form of a sequence of camera poses

$$\mathcal{L} = \{\mathcal{X}_i \mid i = 1, \dots, C-1\} \qquad (1)$$

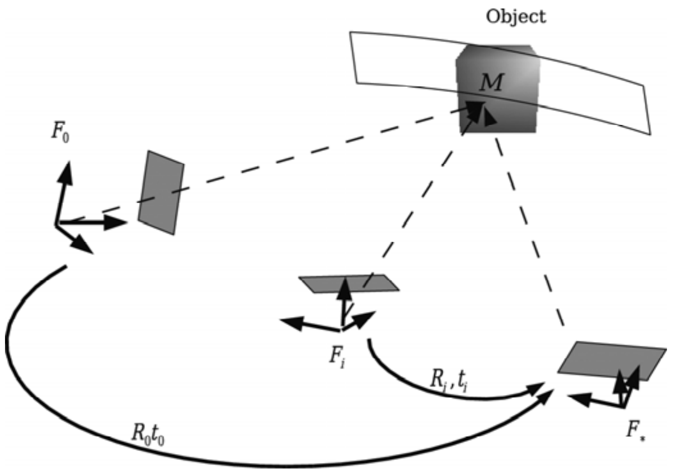


Figure 1. An illustration of the path planning process. M is a set of points belonging to the target object. The frame F_i is the frame attached to the camera at arbitrary intermediate pose $\mathcal{X}_i$ that belongs to the path $\mathcal{L}$ between frames F_0 and F_*.

such that it is optimal in the sense of the given error measure, while satisfying the given constraints. Beginning with F_0 over various iterations F_i it should gradually converge to F_*. Along with the initial and destination poses the path consists of $C+1$ steps.

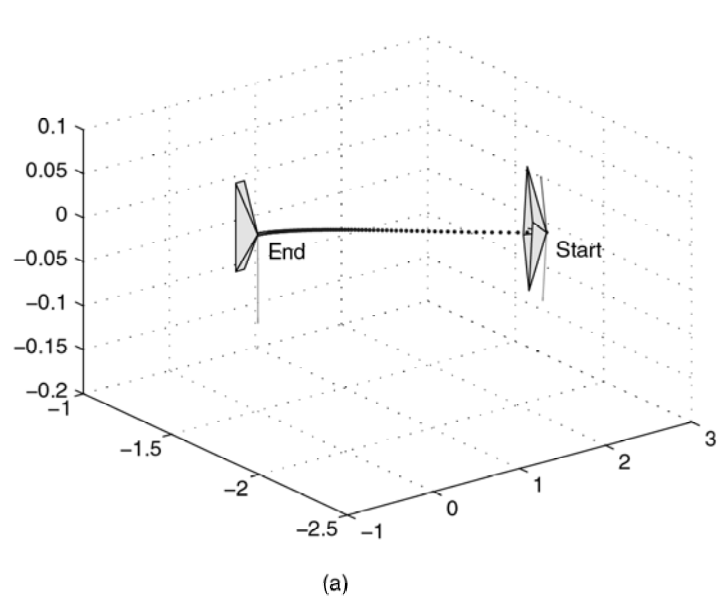
2.1 Trajectory in the Cartesian Space

Let $\mathcal{L} = \{\mathcal{X}_i\}_{i=1}^{(C-1)}$ describe a trajectory or an ordered sequence of $(C-1)$ intermediate camera poses $\mathcal{X}_i$ in the 3D Cartesian space. In each iteration the camera moves towards its destination pose in terms of its translation and rotation parameters. Each camera frame F_i is defined by transformation $[R_i, t_i]$, where t_i is the translation vector and R_i is rotation matrix (computed from roll, pitch and yaw (α_i, β_i, γ_i) angles of the camera) with respect to a reference frame F. The resulting path $\mathcal{L}$ made by putting together the result of each iteration should minimize an error measure.

Consider the shortest possible path between the initial and destination poses. The shortest possible path would obviously be a straight line path $\mathcal{L}_S = \{\mathcal{X}_i^S = (t_i^S \;\; r_i^S) \mid i = 1, \dots, C-1\}$ between the two poses. The iterative process that generates intermediate poses for the straight line path $\mathcal{L}_S$ can be given by a simple linear equation

$$\mathcal{X}_i^S = \mathcal{X}_{i-1}^S + \eta \frac{\mathcal{X}_*^S - \mathcal{X}_{i-1}^S}{\|\mathcal{X}_*^S - \mathcal{X}_{i-1}^S\|}, \quad \eta \geq 0 \qquad (2)$$

However, the straight line path is not always acceptable due to various constraints as described in Section 2.3. Figure 2(a) shows a straight line path from an initial to a destination pose generated by (2), and Fig. 2(b) shows the corresponding image trajectory where feature points leave the camera field of view, which is not acceptable for various applications such as robot localization. As in this case if the straight line path does not fall completely in



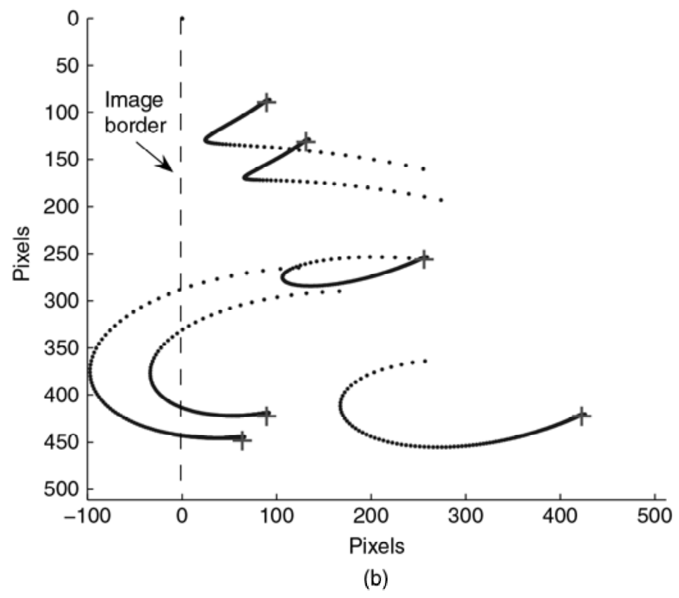


Figure 2. The unconstrained path: (a) a straight line path and (b) the corresponding image trajectory.

the constrained region, an alternative path needs to be planned by minimizing an error function.

For this paper we consider the deviation in the translation space from the straight line path between the initial and destination poses as the error function. However, any meaningful convex function might be used in this framework for a measure of error in the general case.

2.2 Trajectory in the Image Space

The image trajectory is defined at each intermediate pose $\mathcal{X}_i$ by the image projections $\{m_j^i = [p_j^i, q_j^i]^T\}_{j=1}^N$. These are the perspective projections of the N different 3D points $\{M_j = [X_j, Y_j, Z_j]^T\}_{j=1}^N$ in the object frame to the image at the pose $\mathcal{X}_i$. The projection to the image space is done using a perspective camera model with internal parameters $\mathcal{K}$ as follows:

$$m_j^i = \mathcal{K} T_o^i \, M_j$$

where the transformation T_o^i is defined by the rotation R_o and translation t_o between the object frame F_o and the current camera one F_i. In our algorithm, the current transformation T_o^i is a function of the current planned pose $\mathcal{X}_i$ of the camera with respect to the reference frame.

The trajectory in the image space is given by the sequence $\{s^i \mid i = 1, \ldots, C-1\}$. Each s^i is the feature $2N$-vector, *i.e.*,

$$s^i = \left[(m_1^i)^T, \ldots, (m_N^i)^T\right]^T$$

However, it is important that the image trajectory does not contain any large discontinuities, which is ensured by limiting the variation of the feature points across consecutive frames so that the corresponding camera motion is also small.

2.3 Constraints in Path Planning

2.3.1 Visibility Constraints

Visibility of the features is necessary for various applications in different ways. Some applications require the whole object to remain in camera field of view at all times. For a $m \times n$ image, the jth image point $(p_j(\nu) q_j(\nu))^T$ is constrained to lie m_0 pixels from the border along the width of the image and n_0 pixels along the height. These form the following linear constraints:

$$\begin{aligned} p_j(\nu) \geq m_0, \quad -p_j(\nu) + (m - m_0) \geq 0 \\ q_j(\nu) \geq n_0, \quad -q_j(\nu) + (n - n_0) \geq 0 \end{aligned} \tag{3}$$

As shown in Fig. 3, the margins from the corresponding image boundaries are represented in the figure by the margins m_0, n_0 respectively. This problem is a convex quadratic problem and can be solved efficiently using SOCP techniques.

In the above equations, if optimization is done only over translation parameters keeping rotation fixed, then $\nu = (t)$. If rotation is also allowed to vary, then $\nu = (t, \alpha, \beta, \gamma)$.

2.3.2 Workspace and Mechanical Constraints

Every workspace has a number of obstacles that the robot has to avoid for proper navigation. They could be simple planar constraints of the form $\Pi(\nu) \geq a$ like the boundaries of the pathway that do not affect the convexity of the domain of the robot. The same applies for mechanical constraints such as the length of the arm of the robot limiting the reachable points in space. Similarly, a camera fixed on the top of a mobile robot has restrictions on its motion like it does not move in the direction perpendicular

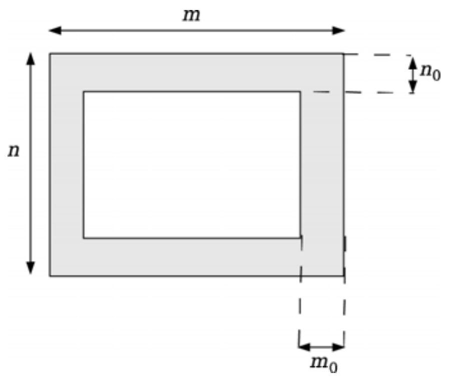


Figure 3. The image space with a resolution of $m \times n$. The shaded region represents the visibility margin of the features with dimensions $m_0 \times n_0$.

to the ground. Such information can be translated to constraints on the free variables in the optimization. They can be any set of convex constraints in general. In some complex cases, however, where the constraint is non-convex we will need to break the domain, say S, into convex regions $\{S_a \mid a = 1, 2, \ldots, A\}$ with $\bigcup_{a=1}^{A} S_a = S$ and reformulate the problem appropriately such that all the constraints are convex in each of the considered domains. The algorithm is then run independently for each of the subsets S_a.

Algorithm 1. Summary of the proposed algorithm

```
Input: Initial and desired camera poses $\mathcal{X}_0$ and $\mathcal{X}_*$, 3D information of the model M and the set of convex constraints $C(\nu)$.
Initialize $\mathcal{L}, \mathcal{L}_S, \mathcal{L}_I, P$ to empty sets.
for $i \in \{1, \ldots, C-1\}$ do
    Find the next straight line camera pose $\mathcal{X}_i^S$ using (2);
    $\mathcal{L}_S \leftarrow \mathcal{L}_S + \{\mathcal{X}_i^S\}$.
    Invoke the branch and bound procedure.
    a. Solve for $\mathcal{X}_i^I$ as in minimization problem in (6);
       $\mathcal{L}_I \leftarrow \mathcal{L}_I + \{\mathcal{X}_i^I\}$; initialize $P = \{D\}$ with eight
       cuboid $D^j$, where $\bigcup_j D_j = D$. $P = \{D\}$.
    b. For each $D^j \in P$, branch the search space $D^j$ into
       smaller disjoint pieces $\sum_k D_k = D^j$; and remove it
       from the considered rotation space $P \leftarrow P - \{D^j\}$.
    c. For each $D_k$ solve the feasibility problem described
       in (8).
       If infeasible then remove $D_k$; else $P \leftarrow P + \{D_k\}$;
       update bounds of $t_i$.
    d. If required accuracy is reached then
           return $\mathcal{X}_i = (t_i, r_i)$
       else go to step 4-b.
end for
$\mathcal{L} \leftarrow \mathcal{L} + \{\mathcal{X}_i\}$.
Output: $\mathcal{L} = \{\mathcal{X}_i \mid i = 1, 2, \ldots, C-1\}$.
```

2.3.3 Smoothness Constraint

The smoothness of the camera path in the Cartesian space can be enforced by forcing the variation of the corresponding image points to be small as discussed in Section 2.2. Let $m_i(\nu) = \left(\frac{p_i(\nu)}{z_i(\nu)}\ \frac{q_i(\nu)}{z_i(\nu)}\right)^T$ be the image of a point after moving the camera, which was initially at $m_i^0 = (p_i^0\ q_i^0)^T$. The variation of the image point can be limited to δ pixels by the following second-order cone constraint

$$\|m_i(\nu) - m_i^0\| \leq \delta \tag{4}$$

For simplicity of notation we use $\mathcal{C}(\nu)$ to represent the set of all constraints on the system.

3. Convex Optimization Path Planning Algorithm

The proposed algorithm aims at generating the optimal path w.r.t. the given constraints, *i.e.*, the set of poses $\mathcal{L} = \{\mathcal{X}_i\}_{i=1}^{(C-1)}$. This path can be mathematically described for each pose $\mathcal{X}_i$ with error function described in Section 2.1 and constraints from Section 2.3 in the following way:

$$\begin{gathered}\mathcal{X}_i = \arg\min_{\nu} \|t_i - t_i^S\| \\ \text{subject to } \mathcal{C}(\nu) \text{ with } \nu = (t_i^S, \alpha_i^S, \beta_i^S, \gamma_i^S)\end{gathered} \tag{5}$$

Here, t_i is the translation of the straight line path produced using (2), while ν is the set of variables over which the objective is optimized subject to the constraint set $\mathcal{C}$.

This problem would have been a convex formulation had translation variables been the only free variables, *i.e.*, $\nu = (t_i^S)$. As the rotation variables are also allowed to vary freely this problem is non-convex in ν and hard to solve by normal convex optimization algorithms. The details of the branching procedure and the bounding function are described in the next section.

3.1 Overview of the Algorithm

We initially present an overview of the proposed algorithm and then get into its details in the subsequent subsections. The algorithm, as mentioned earlier, attempts to find an alternative to the unconstrained straight line path so that the final path lies in the constrained region. The whole algorithm can be simply put into three steps as follows:

1. First, we generate a straight line path as given by (2) and depicted in Fig. 2.

$$\mathcal{L}_S = \{\mathcal{X}_i^S \mid i = 1, 2, \ldots, C-1\}$$

2. We then initialize each of the poses in $\mathcal{L}_S$ by a pose in the constrained region by which we get the path of initialization:

$$\mathcal{L}_I = \{\mathcal{X}_i^I \mid i = 1, 2, \ldots, C-1\}$$

In this path, each pose has the same rotation as in the unconstrained path $\mathcal{L}_S$ but with some change in the translation.

3. Finally we optimize the path $\mathcal{L}_I$ to achieve the optimal path $\mathcal{L}$ of (1). This is done by searching the rotation space using the branch-and-bound algorithm to find a rotation with minimal translation in pose.

The algorithm hence traces from $\mathcal{L}_S \rightarrow \mathcal{L}_I \rightarrow \mathcal{L}$. We now present the details of each of the latter two steps while the first step is already explained in Section 2.1.

We use optimality to mean the shortest path to the destination from the current position without violating any of the given constraints. The resulting path could be different from a straight line path depending on the constraint set. The optimality of the procedure can be proven following the work of [11], [12]. The proof is outlined below. For detailed proof readers are referred to [11], [12].

3.1.1 Outline of the Optimality Proof

We initialize a straight line path that might not satisfy the necessary constraints. We then update each pose along the path for an improved estimate of rotation and translation such that the constraints are met. Search for best rotation is complicated by the complex nature of the Lie Algebraic manifold. Following [11], [12], we use a parameterization of the rotation matrices that can be described by a cuboid. Each point in the cube is a possible candidate for the solution. We define a bound on the smallest possible error from among all rotations within any given cuboid. We use this bound to prune the search space by recursively branching the cuboid of rotations. All cuboids with minimum error above our required error tolerance are removed from further search. This process is repeated until we find a required solution for all the poses. The described procedure completely searches the space to find a solution within the specified tolerance limit. Hence, for a sufficiently small error, it will deliver the most optimal solution.

3.2 Initializing the Path with Translational Constrained Poses

The process of initialization for the path can be done by picking any acceptable pose point in the feasible region for each of the points in $\mathcal{L}_S = \{\mathcal{X}_i^S\}_{i=1}^{(C-1)}$. This is done by using the method proposed in [7]. It generates each of the pose points of $\mathcal{L}_I = \{\mathcal{X}_i^I\}_{i=1}^{(C-1)}$ solving the following minimization over t,

$$\begin{aligned} \mathcal{X}_i^I &= (t_i^I \; r_i^I), \qquad (6)\\ \text{where } & (r_i^I) = (r_i^S) \text{ and } t_i^I \text{ is solution of:}\\ t_i^I &= \arg\min_t \|t - t_i^S\|\\ \text{subject to } & \mathcal{C}(\nu) \text{ with } \nu = (t, \alpha_i^S, \beta_i^S, \gamma_i^S) \end{aligned}$$

Note that the minimization is only over the translation t. This is to say that the rotation angles are fixed and optimization is done on translation parameters alone. This is a convex optimization problem and can be solved using convex optimization techniques. The sketch in Fig. 4 explains the process. All the feature points can be seen from the camera pose $\mathcal{X}_0$, but when the camera moves to its current pose $\mathcal{X}_i$ some of the points on the object move out of camera's field of view. Hence, the translation t_i^S are corrected to t_i^I to get the alternative camera pose from which all the feature points are visible.

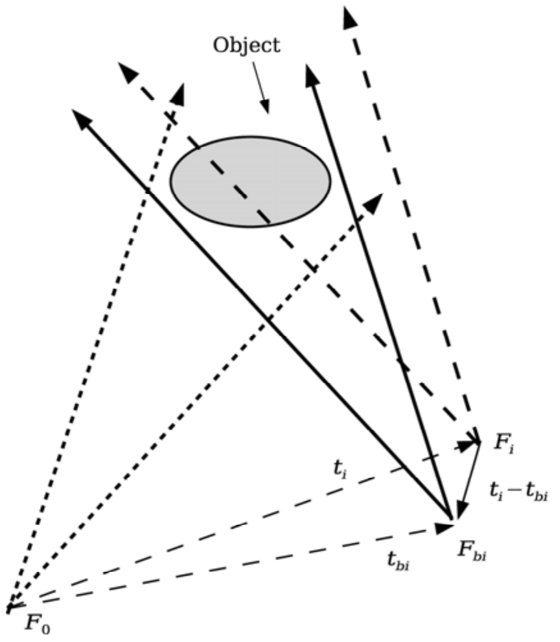


Figure 4. Camera cones that keep the features visible. The frame F_i is attached to the camera at a pose $\mathcal{X}_i^S$ from the unconstraint path. In the Initialized path the frame F_i is replaced by the frame F_{bi}, which is attached to pose $\mathcal{X}_i^I$. One may note that frames F_i and F_{bi} have the same rotations and different translations.

3.3 Searching the Rotation Space using Branch and Bound Algorithm

The problem described in (5) being non-convex we use branch-and-bound algorithm for further optimization over both translation and rotation variables. The space of rotation makes a cuboid D of dimensions $(\alpha_u - \alpha_l) \times (\beta_u - \beta_l) \times (\gamma_u - \gamma_l) = \Delta\alpha \times \Delta\beta \times \Delta\gamma$, where the subscripts u and l indicate the corresponding upper and lower limits, respectively. We need to solve for each $\mathcal{X}_i = (t_i, r_i)$ of (1) by searching the rotation space for better bounds on our variables till we acquire sufficient accuracy.

The actual problem described in (5) can now be rewritten for a space of rotation S as a feasibility problem of the following form:

$$\begin{aligned} \text{Find} \quad & t_i, r_i \in S \qquad (7)\\ \text{s.t.} \quad & \|t_i - t_i^S\| \le \epsilon\\ \text{subject to } & \mathcal{C}(\nu) \text{ with } \nu = (t_i, r_i) \end{aligned}$$

where ϵ is the tolerance limit on the deviation in the translation values. In this manner, we find a pose whose translation is the nearest to t_i^S and whose rotation belongs to the rotation space $S = D$. Ideally, this problem needs to be solved for each of the points r in the rotation space $S = D$ till the required accuracy is achieved, which is

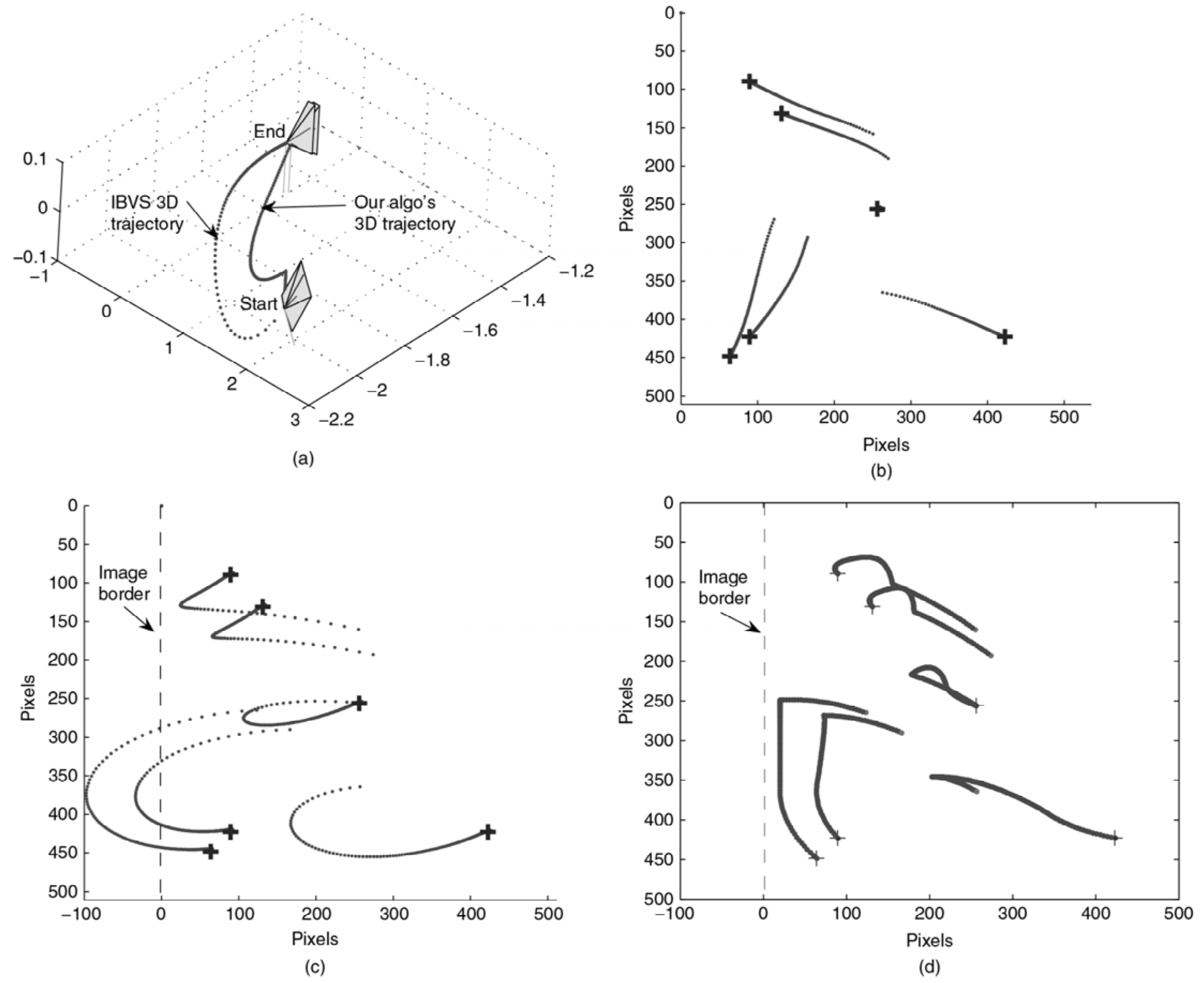


Figure 5. (a) The camera trajectory using our algorithm and using IBVS algorithm. (b) The image trajectory using IBVS. (c) The image trajectory when the camera path is straight line. (d) The image trajectory using our path planning algorithm. Target positions of the image features are marked by "+".

practically impossible. So, we modify the above problem to something that has the same solution as the above one but is still computationally tractable.

Over the given rotation space D the branch-and-bound algorithm is applied by dividing the rotation space D into eight cuboids, then checking on the feasibility of problem given in (7) with respect to translation t and rotation r defined at the centre of each cuboid. This feasibility means there is a better bound on our variables.

The problem that can explain whether a better bound exists than the current one for the problem described in Equation (7) in a given search space S as used in step (3) in section 3.1 is as follows:

$$\begin{aligned} \text{Find} \quad & t_i, r = \overline{S} \\ \text{s.t.} \ \ & \|t_i - t_i^S\| \leq (\bar{\epsilon}) \\ \text{subject to } & \overline{\mathcal{C}}(\nu) \text{ with } \nu = (t) \end{aligned} \tag{8}$$

$\overline{S}$ is the value of r corresponding to the centre of the cuboid S and $\overline{C}(\nu)$ is the set of constraints $\mathcal{C}(\nu)$ modified appropriately. In addition $\bar{\epsilon}$ is a similar modification in the corresponding norm constraint.

This problem can be solved similar to the problem in (6) because the only varying variable is t, as r is fixed at the centre of the cuboid, $\overline{S}$. We need to rewrite constraints $\mathcal{C}(\nu)$ such that the problem described in (8) is feasible for all cases where problem given in (7) is feasible. The branch-and-bound algorithm is summarized in Algorithm 1.

As we are looking only at rotation values fixed at the centre of the search space to ascertain if the whole space D^j possibly has any solution, we need to accommodate for the extreme case of the solution existing at one of the corners of D^j. This is done by making appropriate relaxations in the constraints to reflect this possibility. Let the search space S have edges of lengths $(\Delta\alpha, \Delta\beta, \Delta\gamma) = \Delta r$. Then, a function of the form $f(\nu) \geq 0$ in $\mathcal{C}(\nu)$ is rewritten as

$f(\nu) + \frac{1}{2}\frac{\partial f(\nu)}{\partial r}\Delta r \geq 0$ in $\overline{\mathcal{C}}(\nu)$. This can be done with all the constraints presented in Section 2.3. This is particularly simple for visibility and workspace constraints that are linear functions. However, the smoothness constraint is a second-order function and can be relaxed further as

$$\|m_i(\nu) - m_i^0\| \leq (\delta + \rho) \tag{9}$$

where ρ is the maximum possible variation in pixels, which resulted from the maximum possible rotation variation Δr.

However, the solution is the optimal rotation value that provides a constraint satisfaction with shortest translation vector. This produces a path that is as near to the straight line path as possible.

4. Experimental Evaluation

We demonstrate the flexibility of the algorithm, which makes it applicable to problems in various domains. We discuss application of the algorithm to image-based visual servoing IBVS. We then discuss the performance of our algorithm when used for autonomous robot navigation.

4.1 Visual Servoing

Visual servoing is the use of visual information to close the control loop in robot control system [13]. Visual Servoing has a control part and a feature tracking part. The feature tracking part is necessary to measure the image points of the features in the current frame, which are used in planning the appropriate trajectory. Recent visual servoing algorithms such as [14] proposed to merge both stages into one. Path planning methods that deal them independently are quite efficient at handling various complex situations. The constraints address the feature tracking part but as our algorithm returns only the path to be tracked we further need to generate control signals such that the robot traces the actual path. We use an IBVS controller for the control part of the servoing process. The usual complexities and problems that occur with IBVS controls do not show up in such small motions.

First, we show a positioning task when it is carried out using image-based servo control [15], where the error between the current image features s_i and the image features in the destination image s_* is regulated to zero in the image space. The image trajectory is almost a straight line subject to errors in the internal camera parameters (see Fig. 5(b)) while the 3D camera trajectory as shown in Fig. 5(a) is a complex path to trace. When unconstrained path planning or PBVS algorithms like [16] are used the 3D camera trajectory is a straight line but some feature points move out of the camera's field of view as in Fig. 5(c).

Results from our algorithm are also shown in Fig. 5. One may clearly see that our algorithm outperforms the previous two algorithms: IBVS and PBVS. The camera trajectory as shown in Fig. 5(a) is a straight line when the constraints are satisfied (see Fig. 5(d)) and deviates from the straight line only when constraints could have been violated. The straight line portion of the camera trajectory belongs to the first step of the algorithm where a straight

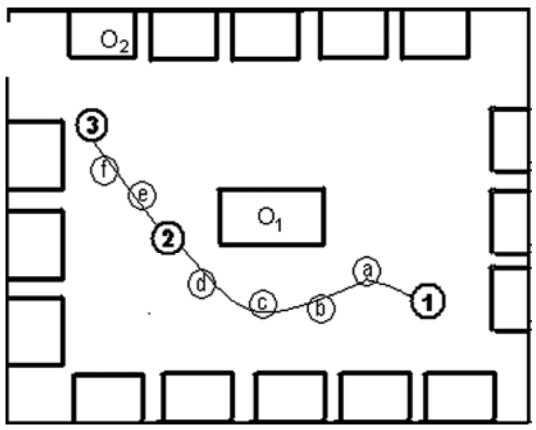


Figure 6. Top view of the 3D path between the selected two key images for the visual navigation example. The camera is mounted on mobile robot. We assume the height of the camera is constat along the path. Pose of the camera at each of the nodes while capturing the key frames in Fig. 7(a–f).

line camera path is generated. In second and third stages of the algorithm the camera's position is modified to the nearest position that satisfies the set of constraints. In other words, we can say that our algorithm, after planning the straight line path, searches for the nearest camera pose to the straight line path that satisfies the set of constraints. Should the straight line path completely lie in the constrained region, the algorithm would return the same result as [16] and produce a perfectly straight camera trajectory.

4.2 Visual Navigation

Autonomous vision-based navigation is the ability of a robot to move one position to another with the aid of the visual information. The robot's knowledge about the environment is available as a visual knowledge base, *i.e.*, as a set of key images in both appearance-based and model-based approaches. This information is used for localization and navigation.

We carried out our experiments by manually building a map of the environment (see Fig. 6). Travelling from from one end of the room to the other end while tracking an object within the room has been set as the goal of the robot. Hence, the room is the domain of the robot.

The domain of the robot is clearly non-convex with no single object that can be tracked from node 1 on one end to node 3 on the other end. Hence, the space has been split into two convex sets as described in Section 2.3 with nodes 1 and 2 making the first set and nodes 2 and 3 making the second set each of which is convex. Node 2 serves as a link between the two sets. A set of key images is made from images of the objects O_1 and O_2 captured from the nodes (see Fig. 7). The tracked feature points of both objects are shown on black patches in all the images. The robot used has a camera mounted on its top with three degrees of freedom: two for position on the floor plane and one for pan angle of the camera. The restriction that

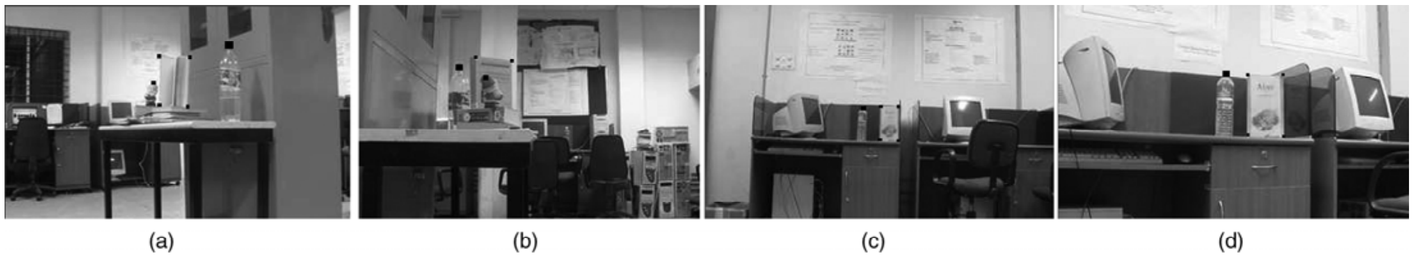


Figure 7. The object O_1 is shown in (a) and (b). It consists of the corners of a book, a toy and the tip of a bottle. The object O_2, shown in (c) and (d), consists of the four corners of the book and tip of another bottle. All the corresponding features are marked with black patches. The key frames shot from each of the nodes are shown. As node 2 is the linking node it captures images of both O_1 and O_2: (a) image of O_1 from node 1; (b) image of O_1 from node 2; (c) image of O_2 from node 2; and (d) image of O_2 from node 3.

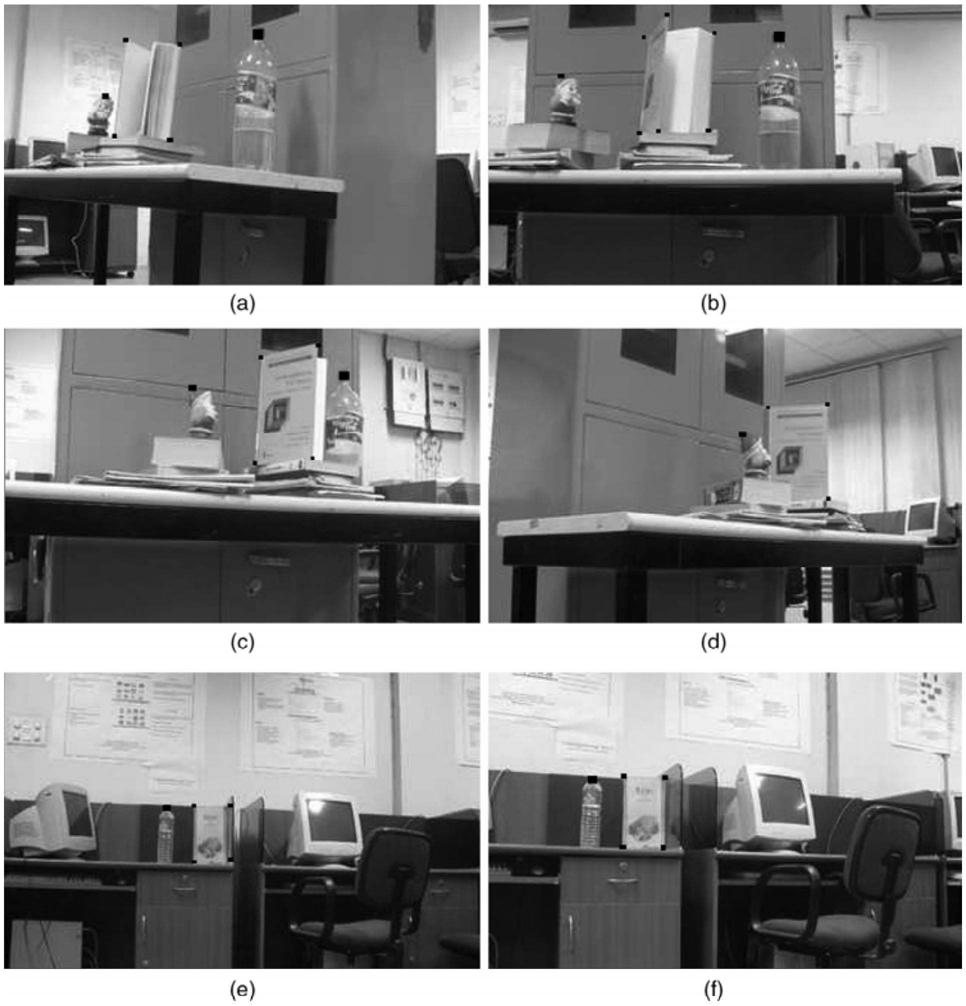


Figure 8. Images of the objects captured while the camera traversed the designed path. The corresponding camera positions on the path are shown in Figure 6; (a) O_1 from a; (b) O_1 from b; (c) O_1 from c; (d) O_1 from d; and (e) O_2 from e. (f) O_2 from f. In each of the above images the objects are completely visible while the robot traverses through the constrained region. All the corresponding features are marked with black patches.

the camera height remains fixed has also been used as a constraint. Visibility and smoothness constraints are used as mentioned in Section 2.3.

We then ran the robot between the nodes on the path generated using the proposed algorithm to traverse from node 1 to node 3 through node 2. Alternative positions were solved for each position lying outside the constrained region by optimizations performed using optimization tools SeDuMi [17] and YALMIP [18]. Figure (6) shows a top view of the path generated by our algorithm. The navigation problem is divided into two: (1) from node 1 to node 2 and (ii) from node 2 to node 3. Images captured at different positions on the path while traversing are shown in Figure (8) with the corresponding camera positions marked in Figure (6) by letters a, b, c, d, e and f. Evidently, the objects that are being tracked are completely visible all along the path for proper localization of the robot and minimize mechanical errors while travelling along the designated path. One can note that the path starts as straight line from node 1 towards node 2. It starts deviating from the straight line path at position a to satisfy the visibility, and other workspace and mechanical constraints.

5. Conclusion

In this paper we have shown that the proposed framework considers the generic case and is promising for practical applications. It is suitable for different systems with variable structure while leaving us with choice on accuracy and computational complexity. A trade-off can be found between accuracy and complexity as the later is inversely proportional to the selected accuracy. However, results shown good accuracy in association with the real application like hardware and sensors' errors. This is due to many aspects. One of them is the path is finally produced in the image space, and image-based visual servo control law is used, which is know as robust odometry and sensory errors, to track the path.

However, the branch-and-bound algorithm discussed here is computationally expensive and can slow some real-time applications. In such cases, we will need to investigate more efficient ways to quickly eliminate irrelevant parts of space efficiently. This is a future direction that is worthy of investigation.

References

[1] A.H. Abdul Hafez, Visual servo control by optimizing hybrid objective function with visibility and path constraints, *Journal of Control Engineering and Applied Informatics*, *16*(2), June 2014, 120–129.

[2] M. Kazemi, K. Gupta, and M. Mehrandezh, Path-planning for visual servoing: A review and issues, in G. Chesi and K. Hashimoto (eds.), *Visual servoing via advanced numerical methods* (LNCI, Springer, 2010), vol. 401, ch. 11, 189–208.

[3] M. Kazemi, K.K. Gupta, and M. Mehrandezh, Randomized kinodynamic planning for robust visual servoing, *IEEE Transactions on Robotics*, *29*(5), 2013, 1197–1211.

[4] Y. Mezouar and F. Chaumette, Path planning for robust image-based control, *IEEE Transactions on Robotics and Automation*, *18*(4), Aug 2002, 534–549.

[5] M. Kazemi, K.K. Gupta, and M. Mehrandezh, Path planning for image-based control of wheeled mobile manipulators, *IROS*, 2012, 5306–5312.

[6] Y. Mezouar and F. Chaumette, Optimal camera trajectory with image-based control, *International Journal of Robotics Research*, *22*(10–11), 2003, 781–804.

[7] A.H. Abdul Hafez, A. Nelakanti, and C.V. Jawahar, Path planning approach to visual servoing with feature visibility constraints: A convex optimization based solution, *IEEE/RSJ International Conf. on Intelligent Robots and Systems, IROS'07*, San Diego, CA, October 2007.

[8] G. Chesi, Visual servoing path planning via homogeneous forms and LMI optimizations, *IEEE Transaction on Robotics*, *25*(2), February 2009, 281–291.

[9] S. Boyd and L. Vandenberghe, *Convex optimization* (Cambridge University Press, New York, NY: 2004).

[10] R. Hartley and F. Kahl, Optimal algorithms in multiview geometry, in Y. Yagi, S.B. Kang, I.S. Kweon, and H. Zha (eds.), *Computer vision – ACCV 2007, ser.* (LNCS, Springer, 2007), vol. 4843, 13–34.

[11] R.I. Hartley and F. Kahl, Global optimization through rotation space search, *International Journal of Computer Vision*, *82*(1), 2009, 64–79.

[12] R. Hartley and F. Kahl, Global optimization through searching rotation space and optimal estimation of the essential matrix, *Proc. 10th IEEE International Conf. on Computer Vision, ICCV'07*, Rio de Janeiro, Brazil, October 2007.

[13] A.H. Abdul Hafez and C.V. Jawahar, Visual servoing by optimization of a 2D/3D hybrid objective function, *IEEE Int. Conf. on Robotics and Automation, ICRA'07*, Roma, Italia, April 2007.

[14] A.H. Abdul Hafez, S. Achar, and C. Jawahar, Visual servoing based on gaussian mixture models, *IEEE International Conference on Robotics and Automation, 2008. ICRA 2008.* IEEE, 2008, pp. 3225–3230.

[15] S. Hutchinson, G. Hager, and Cork, A tutorial on visual servo control, *IEEE Transactions on Robotics and Automation*, *12*(5), October 1996, 651–670.

[16] Y. Mezouar and F. Chaumette, Model-free optimal trajectories in the image space: Application to robot vision control, *IEEE International Conf. on Computer Vision and Pattern Recognition, CVPR'01*, vol. 1, Kaui, Hawaii, December 2001, 1115–1162.

[17] J. Sturm, Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, *Optimization Methods and Software*, 1999, vol. 11–12, 625–653, special issue on Interior Point Methods (CD supplement with software).

[18] J. Lfberg, Yalmip: A toolbox for modeling and optimization in MATLAB, in *Proc. of the CACSD Conf.*, Taipei, Taiwan, 2004. [Online]. Available: http://control.ee.ethz.ch/joloef/yalmip.php.

Biographies

Abdul Hafez Abdul Hafez is an assistant professor at Hasan Kalyoncu University, Gaziantep, Turkey. He completed his Ph.D. in Computer Science and Engineering (2008) from Osmania University, jointly with IIIT Hyderabad, India. His present areas of research include robotic vision, machine learning and vision-based control.

Anil K. Nelakanti has a bachelor's degree in computer science from IIIT Hyderabad, a master's degree from Institut National Polytechnique de Grenoble and a doctorate degree from Universite Pierre et Marie Curie.

C.V. Jawahar is a professor at IIIT Hyderabad, India. His areas of research include robotic and computer vision, machine learning and document image analysis.