Vision-Based Tracking and Trajectory Generation for Robotic Capture of Objects in Space

Guy Rouleau*, Siddharth Verma† and Inna Sharf‡
McGill University, Montreal, Quebec, H3A 2K6, Canada

and

Éric Martin§
Canadian Space Agency, St-Hubert, Quebec, J3Y 8Y9, Canada

In this paper, we describe the development of vision-based algorithms for determining the motion of a moving object and generating trajectory for a robotic manipulator to intercept the object. The ultimate goal of this work is to develop autonomous algorithms for robotic grasping of objects in space. This problem arises in several applications, including on-orbit servicing of satellites and removal of space debris. The proposed methods have been implemented and tested in simulation and are currently being implemented on an experimental facility. The facility is based on a novel concept for experimental evaluation of robotic capture of free-floating objects, in particular, to use a small helium airship to emulate a free-floating object. The paper presents a brief overview of the main components of the facility, describes the vision-based motion estimation, trajectory generation and redundancy resolution algorithms implemented thus far and presents simulation results demonstrating the performance of these algorithms.

Nomenclature

\[ a = \text{Polynomial coefficient} \]
\[ a_t = \text{Predicted target position polynomial coefficient} \]
\[ b = \text{Polynomial coefficient} \]
\[ b_t = \text{Predicted target position polynomial coefficient} \]
\[ c = \text{Polynomial coefficient} \]
\[ c_t = \text{Predicted target position polynomial coefficient} \]
\[ d = \text{Polynomial coefficient} \]
\[ e = \text{Polynomial coefficient} \]
\[ f = \text{Polynomial coefficient} \]
\[ g = \text{Polynomial coefficient} \]
\[ h = \text{Polynomial coefficient} \]
\[ I = \text{Identity matrix} \]
\[ J = \text{Manipulator Jacobian Matrix} \]
\[ J^\# = \text{Manipulator Jacobian pseudoinverse} \]
\[ J_1 = \text{Jacobian Matrix of the primary task space} \]
\[ J_2 = \text{Jacobian Matrix of the secondary task space} \]
\[ k = \text{Cost function projection operator} \]
\[ k_l = \text{Joint limit cost function shaping parameter} \]
\[ k_{TR} = \text{Task reconstruction stopping gain} \]

* Master Thesis Student, Mechanical Engineering, McGill University.
† Doctoral Student, Mechanical Engineering, McGill University.
‡ Associate Professor, Mechanical Engineering, McGill University
§ Robotics Simulation Engineer, Canadian Space Agency

American Institute of Aeronautics and Astronautics
\( k_{TR2} \) = Task reconstruction escaping gain
\( n_m \) = Constant manipulability surface normal vector
\( \Omega_B \) = Position of the CG of the object.
\( p \) = 3D position of the marker
\( \dot{p} \) = Velocity of the marker
\( \ddot{p} \) = Acceleration of the marker
\( m \) = Position of CG of the three markers
\( q \) = Manipulator joint angle vector
\( \dot{q} \) = Manipulator joint velocity vector
\( q_{min} \) = Minimum joint angle vector
\( q_{max} \) = Maximum joint angle vector
\( t \) = Time
\( t_r \) = Manipulator trajectory time
\( t_{search} \) = Time at which target state is predicted
\( \tau \) = Normalized time
\( s \) = Normalized manipulator trajectory
\( x_0 \) = Initial position
\( x_d \) = Final Position
\( \dot{x} \) = Task velocity vector
\( \dot{x}_{max} \) = Maximum velocity
\( \ddot{x}_{max} \) = Maximum acceleration
\( w \) = Scalar cost function
\( w_j \) = Joint limit cost function
\( w_m \) = Manipulability measure
\( \dot{x}_1 \) = Primary task velocity vector
\( \dot{x}_2 \) = Secondary task velocity vector
\( \dot{x}_{TR} \) = Reconstructed end-effector task
\( z \) = Arbitrary vector
\( \Omega \) = Cross product matrix of the angular velocity vector
\( \dot{\Omega} \) = Cross product matrix of the angular acceleration vector
\( \omega \) = Angular velocity vector
\( \dot{\omega} \) = Angular acceleration vector

1. Introduction

A. Background

Robotic manipulators have been used in the space environment for over twenty years. Their functions include deploying and retrieving satellites from orbit and more recently, assembling the Space Station. Over the past several years, much discussion has taken place on in-orbit maintenance and servicing of satellites\(^1\) and spacecrafts. Space scientists have also proposed to employ robotic arms to collect space debris. In all these operations, a robotic manipulator is required to grasp (or grapple) an object that is passively ‘floating’, is under active attitude control or tumbling in an un-cooperating manner. Development and verification of vision-based algorithms for these tasks is the main subject of the present paper.

Our long-term objective for this research is to develop autonomous algorithms for a robotic manipulator to grasp an unknown object, moving or spinning in space. In the short term, we are tackling the following two sub-problems:

1) To determine the kinematic motion variables of the object moving in space from a camera mounted on a robot in a eye-in-hand formation;
2) To use these motion variables to plan and generate an interception path for the robot to capture the object. This problem includes resolving the redundancy of the manipulator.
Vision-based Kinematic Variables Estimation

There is a large body of literature on pose estimation of an object and some of these techniques use vision-based sensors for pose computation\(^2,3,4\). The problem of motion estimation in particular, involving estimation of kinematic variables such as translational velocity and angular velocity, has not been addressed extensively. One of the original works in the area of vision-based 3D estimation of motion variables has been reported\(^5,6\), in which the authors solved only for 3D coordinates of the moving feature points. A technique for estimation of structure and kinematics of an object undergoing smooth 3D motion is reported\(^7\). As this technique uses a monocular camera, the structure and translational variables are scaled along the depth axis, which makes this technique unsuitable for 3D object handling operations. Another technique is presented\(^8\) in, which uses a stereo-camera-based Kalman filter for efficient recursive estimation of 3-D motion variables using a kinematics model-based approach. However, the models were limited by constant angular velocity and constant acceleration assumptions. The technique\(^9\) presents architecture for estimating dynamic state and geometric shape of the object using multiple cooperating range sensors. This architecture is not useful in our research as it is not feasible for us to collect data from multiple cooperating range sensors, which are always around the object.

Trajectory Generation

Trajectory generation for the interception of moving objects is a well-studied problem\(^10,11,12\). In such tasks, the time window where it is possible to intercept the target is limited. This limitation brings the need for an online and near time-optimal trajectory generation to improve the chances of success for the interception. One way of solving this problem involves minimizing the difference between the state of the robot and the state of the object as is the case in operations of guided missiles\(^13\). This approach is reliable for objects with random motion or when the target is very close, but for a general case, it is not necessarily time-optimal. An alternate approach to the interception problem employs a prediction, planning and execution (PPE) strategy. Here, the motion of the target is predicted, then an optimal rendezvous point is planned and finally executed. To be more efficient, this approach can be used in an active way (APPE), where the three steps are repeated in order to compensate for variations in the target motion. The APPE has been used extensively\(^12,14\) and shows satisfactory results.

In the APPE approach, the determination of the rendezvous point depends on the algorithm used to generate the trajectory of the manipulator. In our case, we focus on real-time algorithms, which lead to near time-optimal trajectory, taking into account kinematics limitations of the robot. The target interception problem is close to the pick-and-place operation since they both involve the blending of two points by a smooth interpolation. The difference is that the target interception implies non-stationary initial and final conditions. A solution to this problem consists of using one single fifth-order polynomial\(^14\) or a seventh-order polynomial if jerk continuity is required. For short trajectories this polynomial gives satisfactory results, but when the distance to be traveled is long, this trajectory is far from being time-optimal, since it reaches the maximum velocity at one single instant during the trajectory. A combination of fifth-order polynomials\(^15\) and linear segments is employed to obtain a jerk bounded trajectory with the desired start and end point conditions.

Redundancy Resolution

To respect the real-time constraint of the vision-based target interception, redundancy resolution algorithms taking into account mechanical and computational limitations have been considered. Liegeois\(^16\) proposed to utilize redundancy to avoid mechanical joint limits. As demonstrated for different tasks\(^17\), a potential function that takes larger values as the manipulator approaches its limits can be defined. The gradient of that function can then be projected onto the nullspace of the main task Jacobian to produce a self-motion that will monotonically decrease the potential function without affecting the main task---so-called Gradient Projection Method (GPM). Using another approach, Chang and Dubey\(^18\) have proposed a method based on a weighted least norm solution for a redundant robot. This method does not use directly the self motion to maximize the distance of the joints from their limits, but dampens any motion in the direction of the limits.

In Refs. [17, 19], the concept of task priority is presented and used to prioritize the position relative to orientation. The resulting subtask decomposition is interesting because the reachable workspace of the first manipulation variable is increased by allowing the incompleteness of the secondary subtask. A task reconstruction method\(^20\) is presented to avoid singularity. This method changes the original path close to singularities to a path which insures that a desired level of manipulability is maintained. When combined with the task priority formulation, a specific subtask can be reconstructed to achieve a specific goal.
Few groups around the world have experimental facilities to verify vision-based motion estimation and trajectory generation for objects moving freely in three-dimensional space, targeting specifically the space robotic applications. One of the difficulties is how to emulate a free-floating object in terrestrial environment. Examples of existing facilities for space robotics research include the WATFLEX facility at University of Waterloo, the Aerospace Robotics Laboratory at Stanford University, the Neutral Buoyancy Research Facility at University of Maryland and the Space Telerobotics test-bed at Carnegie Mellon. In these facilities, the weightless environment of space is simulated by using either a flat horizontal surface to confine the motion of the robot and the object to be captured or, a neutral buoyancy water tank or, a gravity compensation system to support the robotic arm and the object. The first method, adopted by several research groups in North America and Japan, is limited to two-dimensional testing and therefore precludes full 3D validation of the dynamics, mission planning and control strategies. The neutral buoyancy water tank has the potential for high-fidelity emulation of weightlessness but is very costly to implement, and, understandably, is a rare facility. A mechanical gravity compensation system involves passive compensation via counter-weight mechanisms and a system of cables and pulleys, as well as additional compensation in software.

The algorithms described in the present paper will be implemented and tested on a novel facility recently constructed at McGill University designed specifically for studying robotic grasping of objects in space. The key component of the facility is a small (5-ft diameter) indoor helium airship employed to emulate a floating object. The facility also includes a six-degree-of-freedom light-duty manipulator arm, moving on a linear 3-m track. The robot is equipped with a stereo ‘eye-in-hand’ vision system to enable execution of the vision-based algorithms that are being developed.

B. About the paper

Following the introduction, we briefly describe the aforementioned experimental facility for studying robotic capture of objects in space. This is followed in Section III by the description of kinematic motion variables estimation of the object motion for trajectory planning for object capture. Next, we explain how this information is used to generate an interception path for the manipulator to capture the target, taking into account the redundancy of the robot. The presentation of the algorithms in these sections includes their descriptions and implementation in simulation. The algorithms are currently being implemented on the facility, with our immediate goal to grasp a known grapple fixture with well-defined visual features on the airship hovering near the robot end-effector.

II. Facility Overview

We present a brief description of the facility constructed at McGill University over the past two years. The reader is referred to referred for a more detailed description.

A. Robot and Controller

As noted earlier, the test-bed includes a CRS 465 six-dof robot with the following specifications: harmonic drive transmission, nominal payload of 2 kg, tip position repeatability of ± 0.05 mm, reach of 711 mm horizontally, outward from base joint axis and vertical reach of 1041 mm from the base.

To meet the objectives of the research planned with the facility, the robot was procured with a 3m linear track. Placing the robot on a track provides several benefits and significantly enhances the capabilities of the system by increasing the robot workspace and adding a redundant degree of freedom. A photograph of the robot on the track is included in Figure 1.

The robot was procured from Quanser with an open architecture controller, allowing full access to the encoder counters, ADC’s, DAC’s and digital I/O via Quanser’s Q8 data acquisition board. The real-time controller for the robot has been developed using RT-LAB from Opal-RT Technologies Inc. This software provides access to the Q8 board from Simulink, in which different control schemes can be rapidly developed, compiled into real-time code and distributed to multiple computation nodes for execution.
The robot position is controlled with independent PID joint controllers that employ conditional anti-windup logic. The gains for the current controller were tuned individually for each joint and the entire controller is currently running at a sample time of 0.0005 seconds. Input commands to the controller can be specified either directly as joint commands or as Cartesian commands fed through the robot’s inverse kinematics to produce the appropriate joint commands. If the set-points are sent to the controller at a slower rate than the controller sample time, as is the case in our visual servoing application, the resulting robot motion will in effect be a sequence of step responses. The trajectory generator described in Section V is used to smooth the motion between these incoming set points.

B. Helium Balloon

The airship used in the facility is a custom design spherical airship of 5 ft in diameter (see Figure 2). The airship was designed to meet three principal requirements: 1) it must closely emulate a free-floating object which requires it to be neutrally buoyant and balanced; 2) the airship must carry a grapple fixture, initially, a simple design and ultimately more sophisticated designs, weighing up to 1 lb; 3) it has to be capable of performing 6-dof motions, to emulate, for example, a spin-stabilized satellite or a spacecraft out of control. The main components of the airship are: a) a 5-ft diameter spherical bladder bag for maximum lift of 1.9 kg; b) six identical propellers mounted in custom-made nacelles in a symmetrical arrangement on the sphere; c) six speed controls for the propellers. These perform two main functions: signal conditioning and amplification of the control signal; d) the battery to power the propellers and electronics; e) an aluminum grapple fixture.

A CAD model of the airship was created for the purpose of laying out components on the airship, keeping account of its mass and inertia properties and for optimizing its mass distribution. A special procedure has been designed to balance the airship, i.e., to move the center of mass of the airship as close as possible to the geometric center of the sphere, as well as to make it neutrally buoyant.

C. Robot Camera

The eyes of the robot are the BumbleBee Firewire stereo camera from Point Grey Research Inc. The camera is mounted to the last joint of the robot in a position shown in Figure 3.

The camera consists of two, 640x480 pixels, Sony grayscale, 1/3” progressive scan CCDs and ships pre-calibrated for lens distortions and camera misalignments. The algorithm described in Section 3 has been developed to interface with the vision library provided by Point Grey. The position of the camera relative to the robot hand has been calibrated by using the technique described in [1].

D. System Architecture

All vision computations, including object identification, feature extraction and motion estimation of the airship, will be performed on the dedicated hardware, and the extracted vision data sent to the robot controller. This allows the robot controller to run at a much higher frequency without being burdened with the computationally intensive vision-processing task. The robot control PC and the vision PC communicate via Ethernet using the low overhead UDP/IP protocol.
III. Vision-based Kinematic Variables Estimation

As alluded earlier, the kinematic motion variables are computed using images captured from a BumbleeBee stereovision sensor placed in eye-in-hand formation on the robot end-effector and these parameters will be used for trajectory planning and autonomous capture of the airship. The kinematic parameter estimation problem is basically divided into two parts, namely, the vision-based estimation of 3D coordinates of the markers on the airship and computation of kinematics motion parameters of the airship from the position of the markers. For the current task, white circular markers are placed on the surface of the balloon to bootstrap the parameter estimation process. The balloon is assumed to be a perfectly spherical rigid body. To date, the images have been acquired from a stationary camera only and this work will be expanded to a moving camera in the near future.

A. Vision-based marker extraction

Initially, the cameras acquire a pixel-based background model of the empty background. Image subtraction is used to detect any object that is moving in the scene. Every incoming image from the camera is subtracted from the background model. The synthetic image obtained from one of the camera views is shown in Figure 4(a). If the subtracted image is completely black, it shows no change in the scene, while if white patches are present in the subtracted image, a new object has appeared in the scene.

Once the object is identified, the image subtraction process is stopped. The pixel position of the Center of Gravity (CG) of the white patch is assumed to be the CG of the object and is used to identify the object in the following frame. The image intensity of the object in the current frame is computed as an average over a 3 x 3 image kernel at the CG position of the object in the subtracted frame. The current image is segmented by subtracting pixel intensity at each pixel with the calculated intensity of the object. The object is assumed to be of same color throughout, therefore, the image is segmented by making all the pixels that have color difference of less than 5 pixels as 1 (white) and making all the other pixels as 0 (black). After the intensity subtraction, the image has many patches/areas of white color against a black background, as shown in Figure 4(b).

The technique of sequential labeling\textsuperscript{33} is used to identify and label all the blobs in the resultant image. A blob is a collection of connected pixels that have similar image intensity and sequential labeling is a technique for identifying these blobs by labeling each pixel in the image based on the pixel intensity of the 8 neighboring pixels. At the end of the labeling process, each pixel in the image is assigned a unique label number. The pixels having the same label number are addressed as a blob and statistical properties such as area and CG of each blob are calculated. The CG of the object computed during image subtraction operation, is used to compare the Euclidian distance between all the remaining blobs obtained in sequential labeling process. As there could only be one blob, which would represent the balloon, all the blobs that have Euclidian distance higher than the least distance blob are deleted.

Once the object is identified, the algorithm searches for manually placed markers on the object (the markers assumed circular and white.) As the markers are of different color than the object, they will appear as holes on the segmented object image, as shown in Figure 4(b). The next task is to identify the position of the markers on the object. To identify the markers, which are only on the object, the rest of the background should be neglected during the image processing. Therefore, the positions of the holes in the image are calculated and copied to a new completely white image. The image of holes against a white background is shown in Figure 4(c). The positions of the holes in the image are computed by first sorting the coordinates of the object blob along rows and then searching along the rows to find gaps in pixel coordinates.

![Figure 4 (a): Synthetic image of the scene, which otherwise would be captured by the camera. (b): Image showing object in foreground as white and background as black. (c): Image containing only the holes positions.](image-url)
Once the new image containing only the holes is obtained (Figure 4(c)), it is processed to identify and label each marker as a blob using the sequential labeling technique\(^{34}\). As there are many small objects mounted on the airship, a method is needed to remove all the blobs that do not represent the manually placed circular markers. The blobs representing circular markers are segregated from the other blobs using simple image correlation technique\(^{34}\). Once the circular markers are identified in one image, a similar process is performed for the second camera image to identify the same circular markers from the second image. An image disparity-based 3D reconstruction technique is then used to compute the 3D positions of the CG of each marker from the 2D position of the CG of the marker for each image.

B. Kinematic variables estimation

Kinematic variable estimation for the airship, i.e., the calculation of the translational velocity (acceleration) of its center and the angular velocity (acceleration) of the airship, requires computing positions, velocities and accelerations of three points on the balloon with respect to an inertial reference frame\(^{8,35}\). To this end, we first compute the 3D positions of three non-collinear points/markers using image processing, as described in Section 3A. The point velocities and acceleration of the same three points are calculated by employing second order finite differences\(^{36}\), as follows

\[
\ddot{p}(t) = \frac{3 \times p(t_k) - 4 \times p(t_{k-1}) + p(t_{k-2})}{2 \times (t_k - t_{k-1})}
\]

where \(p(t_k)\) represents the position of a marker at time \(t_k\). The acceleration values can be computed similarly using the second order finite differences with the velocity values obtained from the above equation.

![Figure 5: Schematic of the kinematic model showing all the coordinate frames](image)

The position, velocity and acceleration of the three markers are expressed as \(p_j(t), \dot{p}_j(t)\) and \(\ddot{p}_j(t)\), respectively, where \(j = 1, 2, 3\), as shown in Figure 5. From\(^{34}\), these velocities and accelerations can be expressed as

\[
\ddot{p}_j - \dot{m} = \Omega (p_j - m)
\]

\[
\ddot{p}_j - \dot{m} = (\Omega + \Omega^2)(p_j - m)
\]
where \( m \) is the centroid of the three marker positions, and, \( \dot{m} \) and \( \ddot{m} \) are averages for the corresponding velocities and accelerations.

Then, the angular velocity vector \( \omega \) and \( \dot{\omega} \) of the airship can be calculated as

\[
\omega = D^{-1} \text{vect}(\dot{P})
\]

\[
\dot{\omega} = D^{-1} \text{vect}(\ddot{P} - \Omega^2 P)
\]

where \( \text{vect}(\Omega) \) is \( \omega \), matrix \( D = \frac{1}{2}[\text{trace}(P)1 - P] \), matrix \( P = [p_1 - m \quad p_2 - m \quad p_3 - m] \) and we obtain matrix \( \dot{P} \) and \( \ddot{P} \) by differentiating matrix \( P \) with respect to time.

Translational velocity of the center of the airship is given by

\[
\dot{p}_{O_{o}}(t) = \dot{p}_{m}(t) + \omega(t) \times (O_{o}(t) - m(t))
\]

where, \( O_{o}(t) \) is the position of the center of the balloon, which is calculated using position of any four markers on the surface and the velocity of point \( m \) is given by:

\[
\dot{p}_{m}(t) = \dot{p}_{i}(t) + \omega(t) \times (m(t) - p_{i}(t))
\]

Similarly the translational acceleration of the center of the object is calculated as

\[
\ddot{p}_{O_{o}} = \ddot{m} + (\Omega + \Omega^2)(p_{O_{o}} - m)
\]

All computations in the above are carried out in the camera reference frame and the resulting kinematics of the airship must be transformed to the inertial frame, to be useful for vision-based interception described in the next sections. The required transformation matrix can be obtained by performing eye-in-hand robot camera calibration, as described in [2].

IV. Vision-based interception

The overall task of intercepting a moving target is divided in the following steps:

Step 1. Tracking

The goal of this phase is to analyze the motion of the target. To achieve this task, the only requirement is to keep the target in the field of view. This is done by orienting the camera to position the target in the center of the image, without changing the camera position. In that phase, only the position information from the vision system is used since the orientation measurement is not sufficiently accurate at far distance.

Step 2. Far Range Approach

The far range approach brings the robot close to the target in order to obtain accurate orientation measurements. In this phase, the camera is oriented to position the target in the center of the image while moving to reduce the distance from the target.

Step 3. Close Range Approach

The close range approach begins when the end-effector is close enough to the target to obtain very accurate measurements of both position and orientation. The goal of this phase is to bring the camera directly in front of the target to exactly match its motion.

Step 4. Grasping

When the camera is very close to the target, it may be impossible to position all the visual features in the field of view of the camera. Therefore, we assume that the motion of the target is continuous and reduce the distance to zero by extrapolating the previous measurements.

For each of the above steps, we need to execute trajectory generation and redundancy resolution, as described in the following sections.
V. Trajectory generation

In this section, an online algorithm for Cartesian trajectory generation for the robot is presented. A predictor of the target motion based on kinematics estimation from Section III to determine an interception point is presented first, followed by a trajectory generation algorithm. The latter is directly applicable to Cartesian translation and small rotations of the robot end-effector. Because of singularities and gimbals lock, trajectory generation for arbitrary orientation is a more complex problem and will be addressed in a future study.

A. Prediction

For the final phases of the interception, the close range approach and grasping, the window of opportunity to intercept the target can be extended if the target motion can be predicted. Using the data obtained from the vision system and the robot kinematics, the predicted target trajectory can be defined in an inertial reference frame by a second-order polynomial in time:

\[ p(t) = at^2 + bt + c, \]  

Then a one-dimensional search is executed on the polynomial to find an optimal time for interception. Starting at \( t = t_{\text{search}} = 0 \), the minimum time (\( t_r \)) needed by the robot to reach the predicted state of the target at time \( t_{\text{search}} \) is computed and the two values are compared. At the first possible interception, both values will be equal (\( t_{\text{search}} = t_r \)). Depending on the trajectory generator, the aforementioned time \( t_r \) needed by the robot to reach the target is computed differently. For a trajectory with non-stationary initial or final state, the problem of finding \( t_r \) requires a numerical solution, implemented as a standard iterative search procedure.

B. Motion Generation

As specified\(^3\), the motion of a robotic mechanical system should be, as a rule, as smooth as possible. In other words, abrupt changes in position, velocity, acceleration and jerk should be avoided. In the case studied here, a Cartesian motion must be planned between an initial non-stationary pose and a final non-stationary pose estimated by a vision system. The initial condition arises from the fact that each new trajectory is generated and begins while the robot is already in motion. At the meeting point the final condition is non-stationary because the robot has to match the state of a moving target.

The method proposed\(^3\) for pick-and-place operations has been adapted to the problem considered here. The Cartesian trajectory is generated from a 7th order interpolation of the form:

\[ s(\tau) = a\tau^7 + b\tau^6 + c\tau^5 + d\tau^4 + e\tau^3 + f\tau^2 + g\tau + h \]

This polynomial is defined for \( 0 \leq \tau \leq 1 \) with \( \tau = t/t_r \) and \( t_r \) is the duration of the trajectory. The mapping between the interpolating polynomial and the Cartesian space is defined by the following:

\[ x(t) = s(\tau) \quad \dot{x}(t) = \frac{\dot{s}(\tau)}{t_r} \quad \ddot{x}(t) = \frac{\ddot{s}(\tau)}{t_r^2} \quad \dddot{x}(t) = \frac{\dddot{s}(\tau)}{t_r^3} \]

Based on the mapping defined by the previous equation and on the desired initial and final conditions on the Cartesian trajectory, the coefficients of the polynomial can be computed:

\[ h = x_0 \quad g = \dot{x}_0 t_r \quad f = \frac{\ddot{x}_0 t_r^2}{2} \quad e = \frac{\dddot{x}_0 t_r^3}{6} \]

The remaining coefficients can be found by solving a system of 4 equations, thus completing the definition of the desired trajectory:

\[
\begin{bmatrix}
\alpha \\
\beta \\
\gamma \\
\delta
\end{bmatrix} = \begin{bmatrix}
-20 & 10 & -2 & 1/6 \\
70 & -34 & 6.5 & -0.5 \\
-84 & 39 & -7 & 0.5 \\
35 & -15 & 2.5 & -1/6
\end{bmatrix}
\begin{bmatrix}
x_f \\
\dot{x}_f \\
\ddot{x}_f \\
\dddot{x}_f \\
\end{bmatrix} = \begin{bmatrix}
e + f + g + h \\
(3e + 2f + g)t_r \\
(6e + 2f)t_r^2 \\
(6e)t_r^3
\end{bmatrix}
\]

For non-stationary initial or final condition, insuring that the generated trajectory will not exceed the maximum velocity and acceleration reachable by the robot is possible only with an iterative search algorithm. This iterative
search procedure has not been implemented to date, but a ‘stationary’ approximation is used instead. In particular, for stationary initial and final conditions, the maximum velocity and acceleration ($\hat{p}_{\text{max}}$ and $\ddot{p}_{\text{max}}$) are directly related to the trajectory duration $t_r$ by:

$$\hat{x}_{\text{max}} = \frac{35}{16t_r}, \quad \ddot{x}_{\text{max}} = \frac{84\sqrt{3}}{25t_r}$$  \hspace{1cm} (14)

These two equations can be used to compute the minimum trajectory time required to avoid exceeding a maximum velocity or acceleration. Using the above approximation instead of the correct value leads to some error, however, this error is continuously reduced as the manipulator approaches the target. In the near future, we will implement an algorithm to traverse long distances faster by combining polynomials and constant velocity segments. In that case, the manipulator will travel for a longer period at constant velocity instead of reaching its maximum velocity only for an instant along the trajectory.

VI. Redundancy Resolution

As discussed in Section II, the robotic manipulator employed in our facility is kinematically redundant because of the presence of the track (7th) axis. Accordingly, in this section we describe how joint trajectories of the manipulator are computed to achieve the desired end-effector trajectory. Several methods for resolving the kinematic redundancy are presented briefly, with their application to the different phases of the interception task.

A. Close Range Approach and Grasping

This phase of the interception task requires controlling the 6 DOF world coordinates of the end-effector in order to match the motion of the target. Therefore, one degree of freedom remains and the self-motion of the manipulator can be used to optimize a scalar function ($w$). Examples of commonly used cost functions can be found17, 38 where individual cost functions are combined to keep the manipulator away from joint limits, increase its manipulability or avoid obstacles. To use the self-motion in this way, joint velocities ($\dot{q}$) are computed based on the desired end-effector velocity ($\dot{x}$) and the Jacobian matrix ($J$) using the gradient projection method (GPM):

$$\dot{q} = J^\# \dot{x} + (I - J^\# J) \left( -k \frac{\partial w(q)}{\partial q} \right)$$  \hspace{1cm} (15)

where $J^\#$ is the Jacobian pseudoinverse and $k$ is the cost function projection operator.

As shown17, when $\dot{x} = 0$ the previous equation guarantees monotonic decrease of the potential function $w(q)$ and for $\dot{x} \neq 0$ the second term still tends to reduce $w(q)$. During the final stages of the interception task, we want to achieve two goals simultaneously. To maximize the chances of the robot grasping the balloon, we need to stay far from singularities and from joint limits. For the latter, the following cost function can be used:

$$w_j(q) = 1 - e^{-k \prod_{i=1}^{n} (q_i - q_{i,\text{min}})(q_{i,\text{max}} - q_i)}$$  \hspace{1cm} (16)

Therefore, $w_j(q) = 1$ when all joints are in the middle of their range and $w_j(q) = 0$ at the joint limits while the shape of the function is determined by the parameter $k_j$.

To avoid robot singularities, or equivalently, to maintain its manipulability, several choices for the cost function are available. The most commonly used measure was proposed by (Yoshikawa 1985) and is

$$w_m(q) = \sqrt{\det(J(q)J(q)^T)}$$  \hspace{1cm} (17)

Another interesting manipulability index is the condition number of the manipulator Jacobian. Regardless of the particular choice, both indexes are combined to define the cost function to be minimized.

$$w(q) = w_j(q)w_m(q)$$  \hspace{1cm} (18)
With the above, we have a general framework for determining the motion of the seven joints of the manipulator to generate a desired end-effector trajectory while optimizing (in the sense of the particular cost function) the configuration of the manipulator. Figure 6 shows how the reachable workspace of our robot on a track can be extended by using GPM method applied to joint limit avoidance. On the left side, the minimum norm solution is used and the manipulator rapidly reaches the extremity of the track ($x=-2m$). In the right plot, the GPM method takes advantage of the redundancy to keep all joints away from their limit before failing to follow the prescribed end-effector translation.

B. Far Range Approach and Tracking

When the manipulator is far from the target, the redundancy resolution options are more interesting. Since the chances of the manipulator colliding with the target are low, the end-effector does not necessarily have to follow a 6 DOF trajectory relative to the target, as described in the previous section. A suitable approach for resolving the redundancy to meet the task requirements is to use the idea of task priority. In the following, we first describe a method to prioritize the tasks of 6DOF trajectory tracking. Subsequently, a task reconstruction algorithm is presented to insures the desired level of the cost function.

Division of the task

The end-effector task is divided in two subtasks with different levels of priority. When the target is outside the workspace of the manipulator, our only goal is to keep the target in the field of view of the camera. In that case it is important to align the camera with the target, but the positioning of the camera is less important. By resolving the redundancy in this way, the positioning task is realized only if it does not interfere with the orientation task. To implement this redundancy resolution scheme, the task velocity vector and the Jacobian matrix are partitioned in their rotational and translational components ($\hat{x}_1, \hat{x}_2$ and $J_1, J_2$). Both tasks are solved separately and then the solution of the secondary task can be projected onto the nullspace of the first task as proposed:

$$\dot{q} = J_1^\# \ddot{x}_1 + J_2^\# \left(\ddot{x}_2 - J_2 J_1^\# \dot{x}_1\right) + \left(I - J_2^\# J_2\right) z$$

where

$$\hat{J}_2 = J_2 \left(I - J_1^\# J_1\right)$$

and $z$ is an arbitrary vector. If an exact solution exists, then both tasks are accomplished, but otherwise, the solution will be the one minimizing $\|\ddot{x}_2 - \hat{J}_2 \dot{\phi}\|$ in the least-squares sense.

In a general task priority formulation is presented. Using this task priority formulation, it is possible to imagine scenarios where different tasks are achieved simultaneously with different order of priority. The question of which task should have the highest priority will be investigated. Different combinations have already been implemented.
using Matlab/Simulink. Many other combinations will be implemented and all methods will be tested and compared experimentally during the next months.

**Task reconstruction**

Methods presented previously can contribute to avoiding singularities and physical joint limits of the robot, but they cannot insure these objectives. To enable a tracking system that can follow a target anywhere in its workspace, a higher degree of robustness is required. A task reconstruction method\(^{20}\) is presented which involves the creation of a sphere around singular points. The manipulator is constrained to move only outside or on the surface of the sphere and cannot move through it. Mathematically, the task reconstruction is achieved by modifying the task velocity vector to obtain:

\[
\dot{q} = J^T (\dot{x} - \dot{x}_{TR})
\]

(21)

where the parameter modifying the task velocity vector is defined as

\[
\dot{x}_{TR} = \frac{1 - \text{sign}(\dot{\mathbf{c}} \cdot \mathbf{n}_m)}{2} k_{TR1}(\mathbf{n}_m \mathbf{n}_m^T)\ddot{\mathbf{c}} + k_{TR2} \mathbf{n}_m
\]

(22)

In the above, \(n_m\) is the constant manipulability surface normal, \(k_{TR1}\) is the task reconstruction stopping gain, \(k_{TR2}\) is the task reconstruction escaping gain. Notice that in equation (21), the self-motion has been neglected since it is decoupled from the end-effector motion. In\(^{20}\), a general framework for task reconstruction inside a multiple subtasks architecture is presented. The feature of reconstructing only the low priority task is now being implemented in our facility. In Figure 7, we illustrate the effect of the task reconstruction method on manipulability. Here, a constant velocity straight line driving the manipulator on a path near a singularity is the desired task velocity vector. In reconstructing the task a desired manipulability level is maintained, avoiding the large joint velocities and tracking errors that would have otherwise occurred near singularities.

**VII. Conclusions**

In this paper, we described the vision-based approach to target motion estimation, trajectory generation and robot control for the task of robotic grasping of a moving target. The intended application of these methods is for autonomous robotic capture of objects in space. The algorithms presented have been verified in simulation and are currently being implemented on a laboratory facility. This test-bed consists of a robotic arm moving on a track and equipped with a stereo camera, as well as a helium balloon employed to emulate a free-floating object. Experimental results with the test-bed will be reported in the near future.

**Acknowledgments**

Funding for this work was provided by the Canadian Space Agency, NSERC and PRECARN through the CSA-IRIS Cooperation in Space Telerobotics and Telemedicine Research program.

**References**


