Zero Robotics Autonomous Space Capture Challenge Manual

v1.3

1 Introduction

1.1 Conventions

Vectors

All vectors in this document are denoted with a bold face font. Of special note is the position vector $\mathbf{x} = \left[\begin{array}{ccc} x & y & z \end{array}\right]^T$ not to be confused with the state vector $\mathbf{X}$ (see blow).

Zero Robotics Attitude

A reduced attitude representation is available to simplify the process of attitude control. The Zero Robotics attitude specifies a pointing direction $\hat{\mathbf{n}}_{ZR} = \left[\begin{array}{ccc} n_x & n_y & n_z \end{array}\right]^T$ for the Velcro (-x) face of the satellite. The API function *ZRSetAttitudeTarget()* orients the satellite to point its Velcro face in the selected direction[1]. No control is provided for the vehicle's rotation about the pointing direction. More advanced users may prefer to use quaternions for attitude control (see next section).

Quaternion Attitude (Advanced)

Internally, SPHERES uses a quaternion to represent rotations. The quaternion represents a transformation from the global coordinate frame to the body frame of the satellite by rotating through an angle θ about a unit vector axis $\hat{\mathbf{n}} = \left[\begin{array}{ccc} n_x & n_y & n_z \end{array}\right]^T$. A quaternion is represented as a vector of length four, with the following properties

$$\begin{aligned} \mathbf{q} &= \left[\begin{array}{cc} \sin\frac{\theta}{2}\cdot\hat{\mathbf{n}} & \cos\frac{\theta}{2} \end{array}\right]^T \\ \|\mathbf{q}\| &= 1 \end{aligned}$$

In our convention $\mathbf{q} = \left[\begin{array}{cccc} q_1 & q_2 & q_3 & q_4 \end{array}\right]^T$ with the scalar part, $\cos\frac{\theta}{2}$, as the fourth term. Please note that the ZR attitude representation specifies the pointing direction for the **-x** face of the satellite, so an identity (no rotation) quaternion $\mathbf{q} = \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \end{array}\right]^T$ corresponds to a ZR attitude of $\hat{\mathbf{n}}_{ZR} = \left[\begin{array}{ccc} -1 & 0 & 0 \end{array}\right]^T$.

Simple cases of the challenge should be possible with either attitude representation, but using quaternions will allow more advanced motion than the ZR attitude. The ZR API provides the function *ZRSetQuatTarget()* to specify a quaternion attitude target, and several math functions are available for performing calculations with them. See the ZR API documentation for details.

State Vectors

Two types of state vectors are available to determine the satellite's full state (position, velocity, orientation, and attitude rates). A 13-element SPHERES state

$$\mathbf{X}_{SPHERES} = \left[\begin{array}{cccc} \mathbf{x} & \dot{\mathbf{x}} & \mathbf{q} & \omega \end{array}\right]^T$$

may be retrieved with the API function *ZRGetMySphState()* and uses a quaternion attitude. Alternatively, the Zero Robotics state

$$\mathbf{X}_{ZR} = \left[\begin{array}{cccc} \mathbf{x} & \dot{\mathbf{x}} & \hat{\mathbf{n}} & \omega \end{array}\right]^T$$

[1] Note that calling *ZRSetAttitudeTarget()* will point the satellite in a specified *direction*, not at a specified *location*.

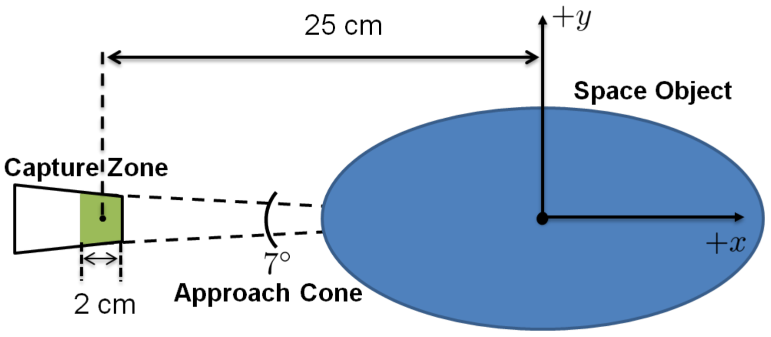

Figure 1: Capture Zone Positioning

may be retrieved with the API function *ZRGetMyZRState()* and uses the ZR attitude. A SPHERES state can be converted to ZR state with the function *ZRSpheresToZR()*.

2 Challenge Statement

2.1 Objective

The Autonomous Space Capture Challenge consists of synchronizing rotational and translational motion with a tumbling space object, thereby setting up the conditions to "capture" it. The challenge specifically focuses on producing a control algorithm that minimizes the propellant cost to capture the object. Competitors will also identify the most challenging docking conditions by specifying several parameters of the space object's motion (see Section 2.3). To complete the challenge, the Tender must:

1. Maneuver to a *Capture Zone* located 25 ± 1 cm along the -x axis in the 7° *Approach Cone* of the space object (see Figure 1). Section 3 provides calculations for determining if the Tender is in the capture zone
2. Align for capture by orienting the -x axis of your satellite within $\pm 2.5°$ of the space object's -x axis (see Figure 2)
3. Stay within the capture zone for 5 seconds with a relative velocity of less than 5 mm/s

while avoiding the following constraints:

1. The Tender must maintain a 30 cm collision avoidance distance from the center of the space object except when in the approach cone (see Figure 3). The approach cone ends at the boundary of the capture zone at 24 cm from the object.
2. Docking must occur while the centers of both the Tender and the space object are within the Object Capture Area. The boundaries are shown in Figure 4. It is important to note that the absolute position of the tender within the test volume will have a high uncertainty. See section 2.4.
3. The Tender must complete the challenge without running out of a virtual tank of propellant. Each time the Tender fires a thruster, a counter records the total time it is open. An allocation of 30 thruster-seconds is allowed for completing the challenge. The total propellant remaining in thruster-seconds is available through the API function *ACGetFuelRemaining()* and is displayed in the visualization.
4. The Tender must complete the capture maneuver within a time period of 210 seconds.

For initial testing it is possible to disable the capture constraints by setting the ZR game variable *DisableConstraints* to 1 in the simulation settings dialog.

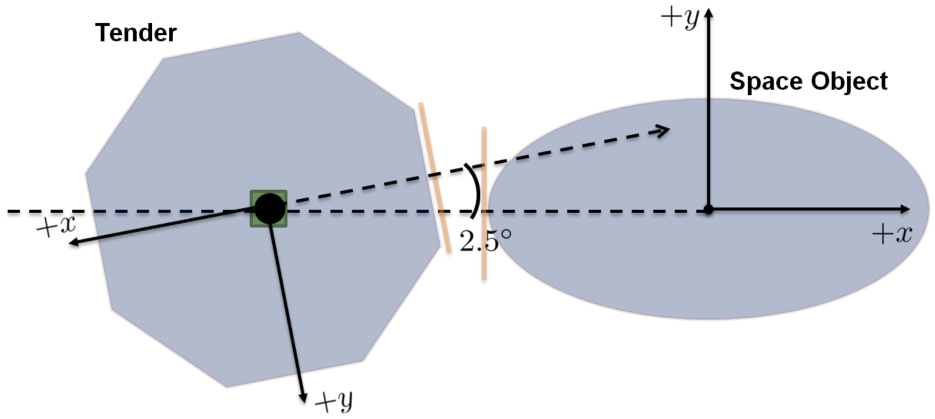

Figure 2: Capture Zone Alignment

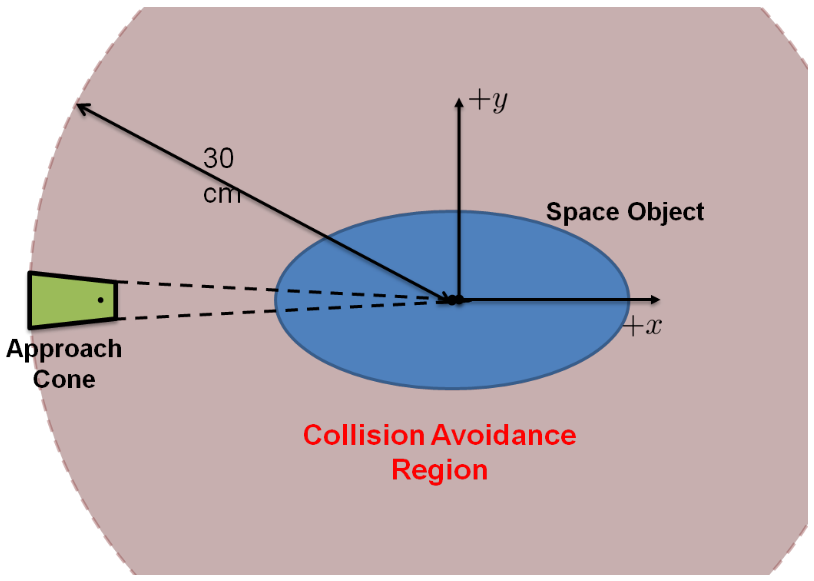

Figure 3: Collision Avoidance Region and Avoidance Cone

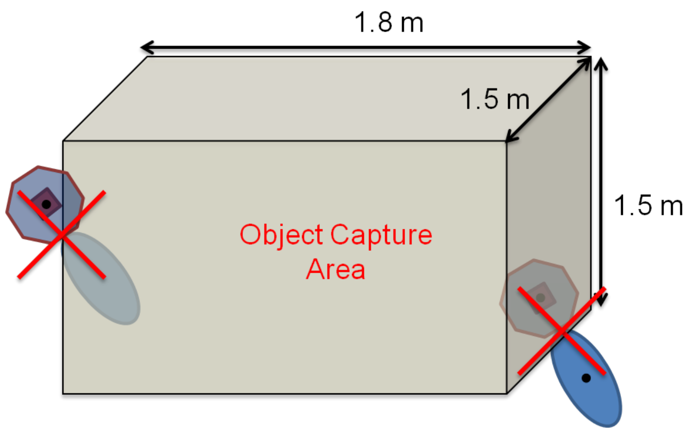

Figure 4: Capture Area

2.2 Tender Initial Conditions

Both satellite competitors will be initialized in the same virtual location in the test volume. The initial position will be on the $-y$ axis:

$$\mathbf{x}_0 = \left[\begin{array}{ccc} 0.0 & -0.6 & 0.0 \end{array}\right]^T$$

The initial orientation starts with the docking port facing the space object side of the volume with the Velcro face of the Tender along the Y axis. This corresponds to a ZR attitude of

$$\mathbf{n}_0 = \left[\begin{array}{ccc} 0 & 1 & 0 \end{array}\right]^T$$

and a quaternion attitude of

$$\mathbf{q}_0 = \left[\begin{array}{cccc} 0.7071 & -0.7071 & 0 & 0 \end{array}\right]^T.$$

The Tender will have no initial velocity or rotation rate with respect to the global coordinate frame.

2.3 Space Object Parameters

A key part of the challenge will be selecting a set of parameters that initialize the state of the space object and affect its motion. Table 1 describes the configurable parameters. The initial position for the object is specified in only the $(x,\, z)$ plane, and the y position is constant:

$$y_0 = 0.6m$$

To initialize the parameters, the API function *ACSetObjectParams()* must be called in *ZRInit()* with an *ObjectParams* data structure. Also, note the limits placed on the parameters. The limits are intended to keep the motion within the thrust capabilities of the SPHERES satellites, though some combinations of parameters may result in impossible scenarios. If any user-supplied parameters exceed the specified limits or the parameters are not specified, the Tender's controls will be disabled, and the submission will be scored as if it did not complete the challenge.

The initial attitude of the space object is specified as a quaternion. If you prefer to use the ZR attitude to specify the orientation, use the API fucntion *ZRAttVec2Quat()* to create a quaternion from the ZR attitude vector.

(Updated 2012-04-14) The initial rotation axis direction specified by ω_0 will be perturbed randomly by up to 8 degrees. For testing purposes the perturbation can be disabled by setting the game variable *DisableSpinAxisNoise* to 1.

2.4 Relative State Information

In contrast to previous Zero Robotics competitions, the state information provided for the space object will be expressed in a coordinate frame centered on the tender's position. This relative state information models the information that would be available to a tender spacecraft when capturing a space object at close proximity using a LIDAR or camera-based sensor. To further emphasize the importance of using relative state information, the position states provided by the *ZRGetMySphState()* and *ZRGetMyZRState()* API functions will be intentionally distorted with additional random noise. The position information will be sufficient to avoid the walls of the volume, but the information should not be used for capturing the object.

Since the objective of the challenge is to develop an independent control system to dock to the target, the API functions ZRGetOther*State() will not return a state for the other player.

2.5 Scoring

Scoring Calculation

During the scoring process, submissions run in head-to-head matches against the top performing projects on the competition leaderboard. For each pairing, the scoring system runs matches with the players as both SPH1 and SPH2, and both players use the space object parameters specified by SPH1. If SPH1 does not specify parameters, the parameters from SPH2 are used, and SPH1 is not scored. Both competitors are initialized in the same positions

Parameter	Description	Limit	Units
I_x, I_y, I_z	Principal moments of inertia	$\frac{I_{max}}{I_{min}} \leq 18$ $I_x + I_y \geq I_z$ $I_y + I_z \geq I_x$ $I_z + I_x \geq I_y$	$kg - m^2$
$\mathbf{v}_0$	Initial velocity	$\|\mathbf{v}_0\| \leq 0.15$	$\frac{m}{s}$
x_0, z_0	Initial position in the (x, z) plane	± 0.75	m
$\mathbf{q}_0$	Initial quaternion orientation	$\|\mathbf{q}\| = 1$	-
ω_0	Initial angular velocity	$\|\omega_0\| \leq 0.16$ $\|\omega_0\| \geq 0.03$	$\frac{rad}{s}$

Table 1: Configurable capture object parameters

and perform the same capture challenge with the same object parameters simultaneously. The final score for the match will be the *difference* in propellant consumed between the two players.

$$\begin{aligned} score_1 &= propUsed_2 - propUsed_1 \\ score_2 &= propUsed_1 - propUsed_2 \end{aligned}$$

It is important to note that although the two competitors in a match will always be performing the same challenge and have identical satellites, the two satellites may be affected by random perturbations in different ways, resulting in small or even large variations in score. This is fully intended as part of the challenge and reflects uncertainties in the satellite dynamic and sensing models. The best performing solutions will be those that prove to be robust to these variations and a wide variety of object parameters.

Partial Completion

If only one competitor completes the challenge in the allotted time without violating constraints, the score for the successful tender (+) is automatically set to the maximum 5 points, and the score for the unsuccessful tender (-) is set to -5.

$$\begin{aligned} score_+ &= 5 \\ score_- &= -5 \end{aligned}$$

As an extra incentive for attempting to complete the challenge, if the unsuccessful tender manages to reach the capture zone for at least one second, and the relative fuel consumption between the satellites is within 1 unit, the unsuccessful tender will receive 0 points instead of -5.

$$\begin{aligned} score_+ &= \text{as above} \\ score_- &= 0 \end{aligned}$$

If neither satellite completes the challenge their scores will both be set to 0.

Entering a Submission

Scores are calculated whenever a competitor enters a *submission*. To enter a submission, follow these steps:

1. Click on the team management link for your team under the Resources → Team Management page (www.zerorobotics.org/we robotics/my-teams)
2. From the menu in the left column, select the *Submissions* button
3. Select the *Submit Project* or *Update Project* button and choose a project to be scored
4. After clicking *Submit* the project will be compiled and checked for code size limitations. If compilation is successful, the project will be queued for scoring.

The scoring process may take up to 15-30 minutes to complete depending on server loads. Attempting to submit another project during the scoring process will result in a notification indicating that the scoring process is still running. If you do not see an option to submit a project, please contact support.

The Competition Leaderboard

The leaderboard will be visible on the ZRASCC tournament detail page after the first submissions are made. The leaderboard operates as follows:

1. When a new submission is entered, a batch simulation is started against the top players in the current leaderboard standings.
2. After each match, the scores of both players are updated. This means the top leaderboard positions are constantly defending against challenges from new submissions.
3. Position on the leaderboard is determined by the average score over all the matches the submission has participated in. If a new match is completed between the same players as an old match, the old score is deleted and replaced with the new match results.

Match visualization results are available by clicking on a ranked player.

2.6 Test Result Codes

When simulations complete in the IDE, they return a single number that indicates the outcome of a test. The following numbers indicate the outcomes of a ZRASCC simulation.

Result Code	Description
10	Capture Completed Successfully
20	Timeout
30	Constraints Violated
40	Out of Propellant
50	Object Parameters not Specified

3 Math Reference and API Functions

3.1 Space Object Motion

The space object is modeled as a rigid body translating and rotating through space with no external forces or torques. The body angular rates of the object follow the Euler equations for rigid body rotation as shown in Equation 1.

$$\begin{aligned}\dot{\omega}_x &= \frac{I_y - I_z}{I_x}\omega_y\omega_z \\ \dot{\omega}_y &= \frac{I_z - I_x}{I_y}\omega_z\omega_x \\ \dot{\omega}_z &= \frac{I_x - I_y}{I_z}\omega_x\omega_y \qquad (1)\end{aligned}$$

The attitude of the object is represented with a quaternion and is propagated according to the following equation

$$\begin{aligned}\dot{\mathbf{q}} &= \frac{1}{2}\boldsymbol{\Omega}\mathbf{q} \\ \boldsymbol{\Omega} &= \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \qquad (2)\end{aligned}$$

The center of the capture zone moves as if rigidly attached to the space object at a distance of 25cm along the body axis as shown in Equation 4. Equation 4 describes the global position of the capture zone given the position of the object, and Equation 5 shows the global velocity of the capture zone taking into account the velocity of the object and its rotational velocity.

$$\begin{aligned}\mathbf{r}_{cap} &= \begin{bmatrix} -0.25 & 0 & 0 \end{bmatrix}^T \qquad (3) \\ \mathbf{x}_{cap} &= \mathbf{x}_{obj} + \mathbf{R}_{b2g}\mathbf{r}_{cap} \qquad (4) \\ \dot{\mathbf{x}}_{cap} &= \dot{\mathbf{x}}_{obj} + \mathbf{R}_{b2g}\left(\omega_{obj} \times \mathbf{r}_{cap}\right) \qquad (5)\end{aligned}$$

The matrix $\mathbf{R}_{b2g}$ converts vectors in the body frame of the object into vectors in the global coordinate frame, and the vector ω_{obj} represents the body frame rotation rates. The attitude target for the capture zone is the space object attitude rotated 180° about the body Z axis. The target can be calculated by post-multiplying the object quaternion by the rotation quaternion $\mathbf{q}_{rot} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T$ as shown in Equation 6. The body rotation rates of the capture zone are likewise rotated by 180°to match the orientation of the tender.

$$\begin{aligned}\mathbf{q}_{cap} &= \mathbf{q}_{obj} \times \mathbf{q}_{rot} \qquad (6) \\ \omega_{cap} &= \mathbf{R}_{rot}\omega_{obj} \qquad (7)\end{aligned}$$

Several utility functions are provided to simplify the task of predicting the motion of the satellite and the capture zone. Full specifications for the functions are provided in the API documents.

ACGetObjectState() Retrieves the SPHERES state of the space object relative to the position of the tender. The (x, y, z) directions are aligned with the global coordinate frame. Attitude is with respect to the the global coordinate frame and rotation rates are with respect to the body frame of the space object.

ACPredictState() Integrates the equations of motion of the object forward from the supplied initial state

ACGetCaptureState() Given a SPHERES state of the space object, determines the full SPHERES state $\mathbf{X}_{cap} = \begin{bmatrix} \mathbf{x}_{cap} & \dot{\mathbf{x}}_{cap} & \mathbf{q}_{cap} & \omega_{cap} \end{bmatrix}^T$ of the center of the capture zone

3.2 Approach Cone

There are two steps to determining if the the Tender is in the capture cone of the space object.

1. Determine which direction the approach cone is pointing in the global frame. The unit vector attitude contained in the ZR state vector provides a quick way of retrieving the direction because it points in the same direction as the cone.
$$\hat{\mathbf{n}}_{cone} = \hat{\mathbf{n}}_{object}$$

2. Calculate the relative unit vector $\hat{\mathbf{r}}$ between the space object and the Tender using their state vectors. Taking the dot product with the unit vector $\hat{\mathbf{n}}_{cone}$ results in the cosine of the angle between the two vectors. If the angle is within the half-cone angle of 2.4° the the Tender is within the approach cone.

$$\begin{aligned} \mathbf{r} &= \mathbf{x}_{tender} - \mathbf{x}_{object} \\ \frac{\mathbf{r}}{\|\mathbf{r}\|} \cdot \hat{\mathbf{n}}_{cone} \geq \cos(2.4^\circ) \quad &\Rightarrow \quad \text{in cone} \end{aligned}$$

The API function *ACInCone()* performs this check when provided a SPHERES state vector for both the Tender and the space object.

3.3 Capture Zone

In addition to the checks in Section 3.2, the capture zone requires the distance to the space object to be within 25 ± 1 cm and the relative velocity within 5 mm/s.

$$\begin{aligned} 0.24 \leq \|\mathbf{r}\| \leq 0.26 \quad & AND \\ \|\dot{\mathbf{r}}\| \leq 0.005 \quad & \Rightarrow \quad in\,capture \end{aligned}$$

Additionally, the relative attitude between the two objects must be within 2.5°. Using the ZR state vector, calculate the dot product between the attitude vector of the Tender and the attitude of the space object.

$$-\hat{\mathbf{n}}_{cone} \cdot \hat{\mathbf{n}}_{tender} \geq \cos(2.5^\circ) \Rightarrow \text{aligned}$$

Note the change of sign now that the axes are anti-aligned. The API function *ACInCapture()* will perform these checks provided a SPHERES state vector for both the Tender and the space object. There is no requirement on the rotation of the satellite about the pointing direction to allow for use of ZR attitude or quaternion attitude.

4 Problem Add-Ons and Adjustments

Several additional problem add-ons have been identified to increase the difficulty of the capture challenge by adding more realistic features. At the end of each week, the Zero Robotics team will evaluate the progress of the competitors and may choose to release one or more add-ons for the upcoming week. The changes are intended to promote more advanced solutions and re-invigorate the solution space once the competitors begin to converge on an equilibrium. Problem simplifications will also be considered if required.

4.1 Pointing Constraint

Many concepts for autonomous capture rely on a LIDAR or camera sensor to track the object during approach. This add-on will require the Tender to be pointed at the space object to retrieve the state from *ACGetObjectState()*.

4.2 State Noise (ACTIVE 2012-04-04)

The level of noise on the global state of the satellite may be increased to discourage its use for position control or lowered to allow for better maneuvering around the boundary constraints.

4.3 Approach Cone

The angle of the approach cone may be decreased to provide more challenging conditions for the final synchronization.

4.4 Propellant Constraint

The propellant allocation may be decreased or increased depending on the performance of the competitors. In general, it is expected that the allocation will only be adjusted if there is a significant difference between the default allocation and the amount required for demonstrated capture scenarios.

4.5 Non-Completion Penalty

The fuel penalty for not completing the capture may be adjusted to better match the scale of scores in the competition.

4.6 Minimum Object Rotation Rate (ACTIVE 2012-04-14)

The rotation rate of the object $\|\omega_0\|$ must be at least 0.03 rad/s. This add-on will encourage a more challenging capture scenario.

5 Example Code and Standard Player

A basic competitor has been created to demonstrate the use of the API functions. It is available as a standard player to compete against under the username 'zrascc' and example code for the player is available in the API documentation. The player does not avoid the collision or boundary constraints, so consider setting "DisableConstraints" when testing the player initially.

Change Log

Version	Description	Author	Date
1.0	Initial Release	jgkatz	2012-03
1.01	Added clarifying note about disabled ZRGetOther*State()	jgkatz	2012-04
1.1	Updated scoring description for partial completion. Activated state noise add-on.	jgkatz	2012-04
1.2	Modified description of leaderboard scoring to reflect new double-match system	jgkatz	2012-04
1.3	Simplified bonus for partial completion and activated new add-on for rotation axis perturbation	jgkatz	2012-04