MATLAB® Support Package for Parrot® Drones

Reference

MATLAB®

R2021b

MathWorks®

How to Contact MathWorks

Latest news:	`www.mathworks.com`
Sales and services:	`www.mathworks.com/sales_and_services`
User community:	`www.mathworks.com/matlabcentral`
Technical support:	`www.mathworks.com/support/contact_us`
☎ Phone:	508-647-7000

✉ The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

MATLAB® Support Package for Parrot® Drones Reference

Trademarks

Patents

Revision History

March 2019	Online only	New for Version 19.1.0 (R2019a)
April 2019	Online only	Revised for Version 19.1.1 (R2019a)
May 2019	Online only	Revised for Version 19.1.2 (R2019a)
September 2019	Online only	Revised for Version 19.2.0 (R2019b)
March 2020	Online only	Revised for Version 20.1.0 (R2020a)
September 2020	Online only	Revised for Version 20.2.0 (R2020b)
March 2021	Online only	Revised for Version 21.1.0 (R2021a)
September 2021	Online only	Revised for Version 21.2.0 (R2021b)

Contents

1 Functions

2 Objects

1

Functions

abort

End flight of Parrot drone

Syntax

```
abort(parrotObj)
```

Description

`abort(parrotObj)` instantaneously ends the flight of Parrot drone, represented by `parrotObj`, by shutting down all the motors. Use `takeoff` to begin a new flight . This is a blocking call. In other words, MATLAB blocks the command line until the current command runs to completion.

Examples

Abort Parrot Drone

Connect to a Parrot drone

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is flying, abort the flight of the drone instantaneously by shutting down all the motors.

```
abort(parrotObj)
```

Input Arguments

`parrotObj` — Parrot drone connection
`parrot` object

Parrot drone connection object, specified as a `parrot` on page 2-5 object.

See Also

`land` | `takeoff` | `flip` | `turn`

Introduced in R2019a

closePreview

Close Parrot drone FPV camera preview window

Syntax

```
closePreview(cameraObj)
```

Description

`closePreview(cameraObj)` closes the preview window of the Parrot drone camera, specified as `cameraObj`. You can close the preview at any time using the `closePreview` function. If you do not explicitly close the preview, it closes when you clear the camera object.

Examples

Close FPV Camera Preview

Connect to a Parrot Mambo FPV drone over a wireless network.

```
parrotObj = parrot('Mambo');
```

Connect to the FPV camera

```
cameraObj = camera(parrotObj,'FPV')
```

```
cameraObj = 
          camera with properties:

                    Name: "FPV"
              Resolution: "640x360"
```

Preview the image from the camera

```
preview(cameraObj);
```

The preview window opens and displays live video stream from your camera. The banner of the preview window shows the camera URL. The lower portion of the window shows the timestamp in seconds, resolution, and frame rate in frames per second.

Close the preview

```
closePreview(cameraObj);
```

Input Arguments

`cameraObj` — Parrot drone camera connection
`camera` object

Parrot drone camera connection object, specified as a `camera` on page 2-2 object.

See Also

`preview` | `camera` | `snapshot`

Introduced in R2019a

flip

Flip Parrot drone in specified direction

Syntax

```
flip(parrotObj,direction)
```

Description

`flip(parrotObj,direction)` flips the Parrot drone, represented by `parrotObj`, in the specified direction. This is a blocking call. In other words, MATLAB blocks the command line until the current command runs to completion.

Note The Parrot drone does not flip if the battery level is low. Check the drone's battery level before you use the function.

Examples

Move Parrot Drone in Specified Direction

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, flip the drone forward.

```
flip(parrotObj,'forward');
```

Input Arguments

`parrotObj` — Parrot drone connection
`parrot` object

Parrot drone connection object, specified as a `parrot` on page 2-5 object.

`direction` — Direction to flip drone
`'forward'` | `'right'` | `'left'` | `'back'`

The direction in which the Parrot drone flips, specified as a character vector.

See Also

`turn` | `takeoff` | `land` | `abort`

Introduced in R2019a

land

Land Parrot drone

Syntax

```
land(parrotObj)
```

Description

`land(parrotObj)` initiates the gradual landing of Parrot drone, represented by `parrotObj`, from the current position and ends the drone flight. Use `takeoff` to begin a new flight. This is a blocking call. In other words, MATLAB blocks the command line until the current command runs to completion.

Examples

Land Parrot Drone

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot is flying, initiate the landing of the drone.

```
land(parrotObj)
```

Input Arguments

`parrotObj` — Parrot drone connection
`parrot` object

Parrot drone connection object, specified as a `parrot` on page 2-5 object.

See Also

`takeoff` | `abort` | `flip` | `turn`

Introduced in R2019a

move

Move Parrot drone in all six directions

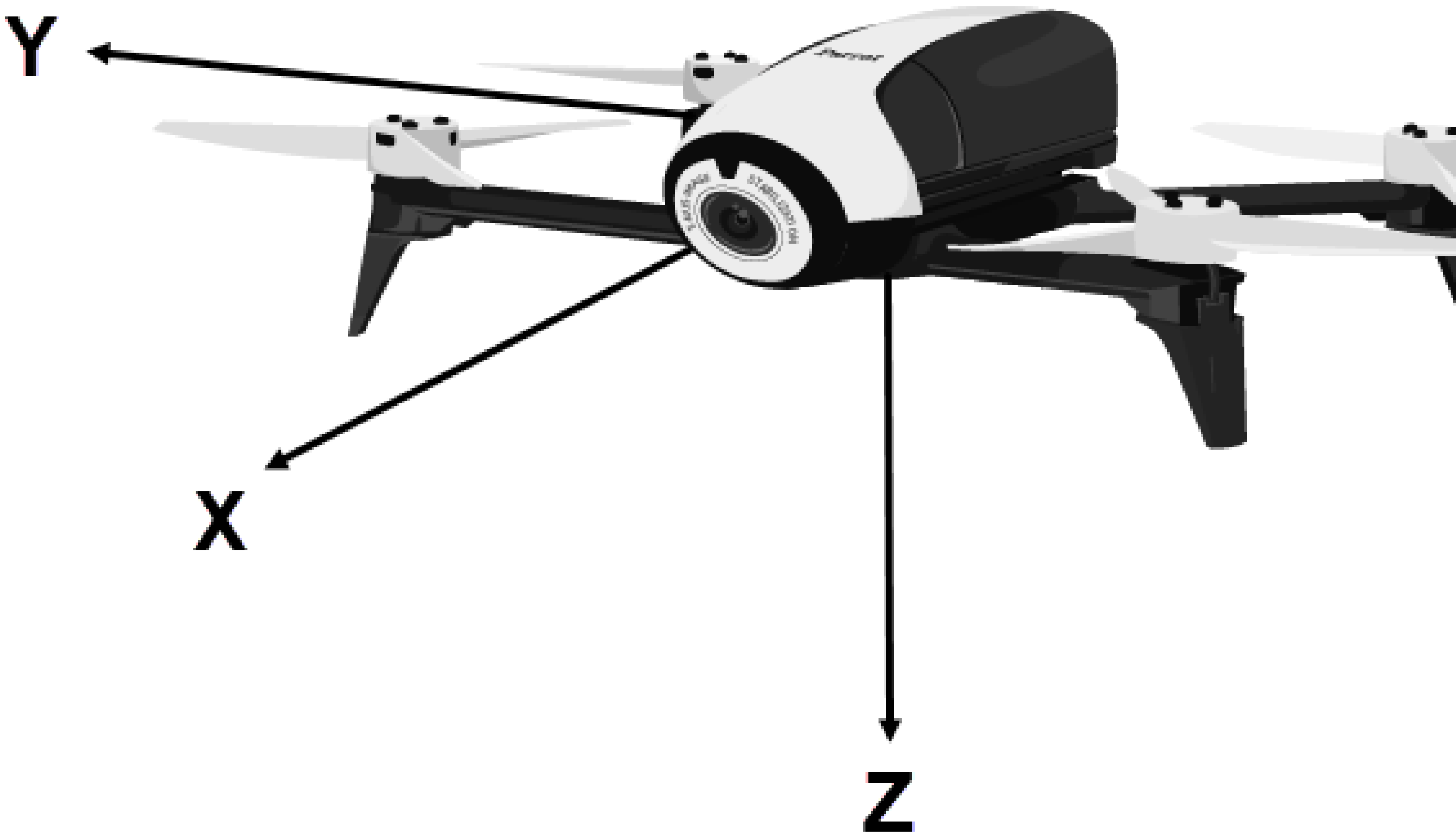
Syntax

```
move(parrotObj,Name,Value)
move(parrotObj,duration,Name,Value)
```

Description

`move(parrotObj,Name,Value)` moves the Parrot drone, represented by `parrotObj`, in all six directions based on attitude angles, rotation speed, and vertical speed specified as `Name,Value` pair arguments.

The following schematic shows the quadcopter axis characteristics :



The quadcopter body axis is centered in the center of gravity.

- The x-axis starts at the center of gravity and points in the direction along the nose of the quadcopter.
- The y-axis starts at the center of gravity and points to the right of the quadcopter.
- The z-axis starts at the center of gravity and points downward from the quadcopter, following the right-hand rule.

`move(parrotObj,duration,Name,Value)` moves the Parrot drone, represented by `parrotObj`, in all six directions based on attitude angles, rotation speed, and vertical speed specified as `Name,Value` pair arguments for the duration specified as `duration`.

Examples

Move Parrot Drone Diagonally Forward

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone diagonally forward with a `Pitch` and `Roll` of -10 degrees and +10 degrees respectively for the default duration of 0.5 seconds.

```
move(parrotObj,'pitch',deg2rad(-10),'roll',deg2rad(10));
```

Move Parrot Drone Diagonally Backwards For a Specified Duration

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone diagonally backwards for 2 seconds with a Pitch and Roll of +10 degrees and +10 degrees respectively

```
move(parrotObj,2,'pitch',deg2rad(10),'roll',deg2rad(10));
```

Input Arguments

parrotObj — Parrot drone connection

parrot object

Parrot drone connection object, specified as a parrot object.

duration — Duration in seconds

0.5 seconds (default) | positive real scalar

The time for which the Parrot drone moves in the specified direction for the specified time. The default value is 0.5 seconds.

Data Types: double

Name-Value Pair Arguments

Specify at least one comma-separated pair of Name,Value arguments. Name is the argument name and Value is the corresponding value. Name must appear inside quotes. You can specify several name and value pair arguments in any order as Name1,Value1,...,NameN,ValueN

Example: move(parrotObj,3,'VerticalSpeed',-2);

Pitch — Movement of drone along x-axis

0 radians (default)

The pitch angle in which the Parrot drone moves along the x-axis, specified in radians. The drone moves forward, if the Pitch is negative. If the Pitch is positive, the drone moves backwards.

The following Pitch values are valid for the Parrot drones

Parrot Drone	Pitch value
Parrot Mambo	-0.436 rad (-25degrees) to 0.436 rad (25 degrees)
Parrot Bebop 2	-0.6109 rad (-35degrees) to 0.6109 rad (35 degrees)

Example: move(parrotObj,3,'Pitch',deg2rad(2));

Data Types: double

Roll — Movement of drone along y-axis

0 radians (default)

The roll angle in which the Parrot drone, moves along the y-axis, specified in radians. The drone moves to the left if the Roll is negative. If the Roll is positive, the drone moves right.

The following Roll values are valid for the Parrot drones

Parrot Drone	Roll value
Parrot Mambo	-0.436 rad (-25degrees) to 0.436 rad (25 degrees)
Parrot Bebop 2	-0.6109 rad (-35degrees) to 0.6109 rad (35 degrees)

Example: `move(parrotObj,3,'Roll',deg2rad(2));`

Data Types: `double`

`RotationSpeed` — Speed of drone around z-axis

0 rad/s (default)

The speed at which the Parrot drone moves around the z-axis, specified in radians/s. The drone moves in the counterclockwise direction if the `RotationSpeed` is negative. If the `RotationSpeed` is positive, the drone moves in the clockwise direction.

The following `RotationSpeed` values are valid for the Parrot drones

Parrot Drone	RotationSpeed value
Parrot Mambo	-pi rad/s (-180 deg/s) to +pi rad/s (180 deg/s)
Parrot Bebop 2	-3.4907 rad/s (-200 deg/s) to 3.4907 rad/s (200 deg/s)

Example: `move(parrotObj,3,'RotationSpeed',deg2rad(2));`

Data Types: `double`

`VerticalSpeed` — Speed of the drone along z-axis

0 m/s (default)

The speed at which the Parrot drone moves along z-axis, specified in m/s. The drone ascends, if the `VerticalSpeed` is negative. If the `VerticalSpeed` is positive, the drone descends.

The following `VerticalSpeed` values are valid for the Parrot drones

Parrot Drone	VerticalSpeed value
Parrot Mambo	-2 m/s to +2 m/s.
Parrot Bebop 2	-6 m/s to +6 m/s.

For Parrot Mambo drones, the valid values for `VerticalSpeed` is between -2 m/s to +2 m/s.

Example: `move(parrotObj,3,'VerticalSpeed',-2);`

Data Types: `double`

See Also

`moveback` | `movedown` | `moveforward` | `moveleft` | `moveright` | `moveup`

Introduced in R2019a

moveback

Move Parrot drone backwards

Syntax

```
moveback(parrotObj)
moveback(parrotObj,duration)
moveback(parrotObj,duration,tilt)
```

Description

`moveback(parrotObj)` moves the Parrot drone backwards.

`moveback(parrotObj,duration)` moves the Parrot drone backwards for the specified time.

`moveback(parrotObj,duration,tilt)` moves the Parrot drone backwards for the specified time at a specified angle.

Examples

Move Parrot Drone Backwards

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone backwards for the default duration of 0.5 seconds.

```
moveback(parrotObj);
```

Move Parrot Drone Backwards for Specified Duration

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone backwards for 5 seconds.

```
moveback(parrotObj,5);
```

Move Parrot Drone Backwards for Specified Duration and Angle

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone backwards for 5 seconds at an angle of 0.5236 radians.

```
moveback(parrotObj,5,deg2rad(30));
```

Input Arguments

`parrotObj` — Parrot drone connection
`parrot` object

Parrot drone connection object, specified as a `parrot` object.

`duration` — Duration in seconds
`0.5` (default) | positive real scalar

The time for which the Parrot drone moves back, specified in seconds.

Data Types: `double`

`tilt` — Absolute value of angle at which drone moves
`0.052` (default) | positive real scalar

The absolute value of the pitch angle at which the Parrot drone moves, measured in radians.

Data Types: `double`

See Also

`move` | `moveforward` | `parrot` | `takeoff`

Introduced in R2019a

movedown

Move Parrot drone down

Syntax

```
movedown(parrotObj)
movedown(parrotObj,duration)
movedown(parrotObj,duration,speed)
```

Description

`movedown(parrotObj)` moves the Parrot drone, represented by `parrotObj`, downwards.

`movedown(parrotObj,duration)` moves the Parrot drone, represented by `parrotObj`, down for the specified time.

`movedown(parrotObj,duration,speed)` moves the Parrot drone, represented by `parrotObj`, down for the specified time at a specified speed.

Examples

Move Parrot Drone in the Downwards

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone downwards for the default duration of 0.5 seconds.

```
movedown(parrotObj);
```

Move Parrot Drone in the Downwards for Specified Duration

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone down for 2 seconds.

```
movedown(parrotObj,2);
```

Move Parrot Drone in the Downwards for Specified Duration and Speed

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone down for 2 seconds at a speed of 1 m/s.

```
movedown(parrotObj,2,1);
```

Input Arguments

`parrotObj` — Parrot drone connection

`parrot` object

Parrot drone connection object, specified as a `parrot` object.

`duration` — Duration in seconds

`0.5` (default) | positive real scalar

The time for which the Parrot drone moves down, specified in seconds.

Data Types: `double`

`speed` — Vertical speed at which drone moves

`0.2` (default) | positive real scalar

The absolute value of vertical speed at which the Parrot drone moves down, specified in m/s. The default value is 0.2m/s.

Data Types: `double`

See Also

`move` | `moveup` | `parrot`

Introduced in R2019a

moveforward

Move Parrot drone forward

Syntax

```
moveforward(parrotObj)
moveforward(parrotObj,duration)
moveforward(parrotObj,duration,tilt)
```

Description

`moveforward(parrotObj)` moves the Parrot drone, represented by `parrotObj`, forward.

`moveforward(parrotObj,duration)` moves the Parrot drone, represented by `parrotObj`, forward for the specified time.

`moveforward(parrotObj,duration,tilt)` moves the Parrot drone, represented by `parrotObj`, forward for the specified time at a specified angle.

Examples

Move Parrot Drone Forward

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone forward for the default duration of 0.5 seconds.

```
moveforward(parrotObj);
```

Move Parrot Drone Forward for Specified Duration

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone forward for 5 seconds.

```
moveforward(parrotObj,5);
```

Move Parrot Drone Forward for Specified Duration and Angle

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone forward for 5 seconds at an angle of 0.7854 radians.

```
moveforward(parrotObj,5,deg2rad(45));
```

Input Arguments

`parrotObj` — Parrot drone connection

`parrot` object

Parrot drone connection object, specified as a `parrot` object.

`duration` — Duration in seconds

`0.5` (default) | positive real scalar

The time for which the Parrot drone moves forward, specified in seconds.

Data Types: `double`

`tilt` — Absolute value of angle at which drone moves

`0.052` (default) | positive real scalar

The angle at which the Parrot drone moves forward, specified in radians.

Data Types: `double`

See Also

`move` | `moveback` | `parrot` | `takeoff`

Introduced in R2019a

moveleft

Move Parrot drone left

Syntax

```
moveleft(parrotObj)
moveleft(parrotObj,duration)
moveleft(parrotObj,duration,tilt)
```

Description

`moveleft(parrotObj)` moves the Parrot drone to the left.

`moveleft(parrotObj,duration)` moves the Parrot drone to the left for the specified time.

`moveleft(parrotObj,duration,tilt)` moves the Parrot drone to the left for the specified time at a specified angle.

Examples

Move Parrot Drone Left

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone to the left for the default duration of 0.5 seconds.

```
moveleft(parrotObj);
```

Move Parrot Drone Left for Specified Duration

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone to the left for 5 seconds.

```
moveleft(parrotObj,5);
```

Move Parrot Drone Left for Specified Duration and Angle

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone to the left for 5 seconds at an angle of 0.0873 radians.

```
moveleft(parrotObj,5,deg2rad(5));
```

Input Arguments

`parrotObj` — Parrot drone connection

`parrot` object

Parrot drone connection object, specified as a `parrot` object.

`duration` — Duration in seconds

`0.5` (default) | positive real scalar

The time for which the Parrot drone moves to the left, specified in seconds.

Data Types: `double`

`tilt` — Absolute value of angle at which drone moves
`0.052` (default) | positive real scalar

The angle at which the Parrot drone moves to the left, specified in radians.

Data Types: `double`

See Also

`move` | `moveright` | `parrot` | `takeoff`

Introduced in R2019a

moveright

Move Parrot drone right

Syntax

```
moveright(parrotObj)
moveright(parrotObj,duration)
moveright(parrotObj,duration,tilt)
```

Description

`moveright(parrotObj)` moves the Parrot drone, represented by `parrotObj`, to the right.

`moveright(parrotObj,duration)` moves the Parrot drone, represented by `parrotObj`, to the right for the specified time.

`moveright(parrotObj,duration,tilt)` moves the Parrot drone, represented by `parrotObj`, to the right for the specified time at a specified angle.

Examples

Move Parrot Drone to Right

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone to the right for the default duration of 0.5 seconds.

```
moveright(parrotObj);
```

Move Parrot Drone to Right for Specified Duration

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone to the right for 5 seconds.

```
moveright(parrotObj,5);
```

Move Parrot Drone to Right for Specified Duration and Angle

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone to the right for 5 seconds at an angle of 0.0873 radians.

```
moveright(parrotObj,5,deg2rad(5));
```

Input Arguments

`parrotObj` — Parrot drone connection

`parrot` object

Parrot drone connection object, specified as a `parrot` object.

`duration` — Duration in seconds

`0.5` (default) | positive real scalar

The time for which the Parrot drone moves to the right, specified in seconds.

Data Types: `double`

tilt — Absolute value of angle at which drone moves

0.052 (default) | positive real scalar

The angle at which the Parrot drone moves to the right, specified in radians.

Data Types: double

See Also

move | moveleft | parrot | takeoff

Introduced in R2019a

moveup

Move Parrot drone upwards

Syntax

```
moveup(parrotObj)
moveup(parrotObj,duration)
moveup(parrotObj,duration,speed)
```

Description

`moveup(parrotObj)` moves the Parrot drone, represented by `parrotObj`, upwards.

`moveup(parrotObj,duration)` moves the Parrot drone, represented by `parrotObj`, upwards for the specified time.

`moveup(parrotObj,duration,speed)` moves the Parrot drone, represented by `parrotObj`, upwards for the specified time at a specified speed.

Examples

Move Parrot Drone Upwards

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone upwards for the default duration of 0.5 seconds.

```
moveup(parrotObj);
```

Move Parrot Drone Upwards for Specified Duration

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone upwards for 2 seconds.

```
moveup(parrotObj,2);
```

Move Parrot Drone Upwards for Specified Duration and Speed

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, move the drone upwards for 2 seconds at a speed of 1 m/s.

```
moveup(parrotObj,2,1);
```

Input Arguments

`parrotObj` — Parrot drone connection

`parrot` object

Parrot drone connection object, specified as a `parrot` object.

`duration` — Duration in seconds

`0.5` (default) | positive real scalar

The time for which the Parrot drone moves up, specified in seconds.

Data Types: `double`

`speed` — Vertical speed at which drone moves

`0.2` (default) | positive real scalar

The absolute value of speed at which the Parrot drone moves up, specified in m/s.

Data Types: `double`

See Also

`move` | `movedown` | `parrot` | `takeoff`

Introduced in R2019a

preview

Preview live video data from Parrot drone FPV camera

Syntax

```
preview(cameraObj)
```

Description

`preview(cameraObj)` creates a preview window that displays live video data from a Parrot drone FPV camera object,represented as `cameraObj`. The preview window also displays the camera URL, resolution, frame rate, and timestamp.

Examples

Preview FPV Camera Image

Connect to a Parrot Mambo FPV drone over a wireless network.

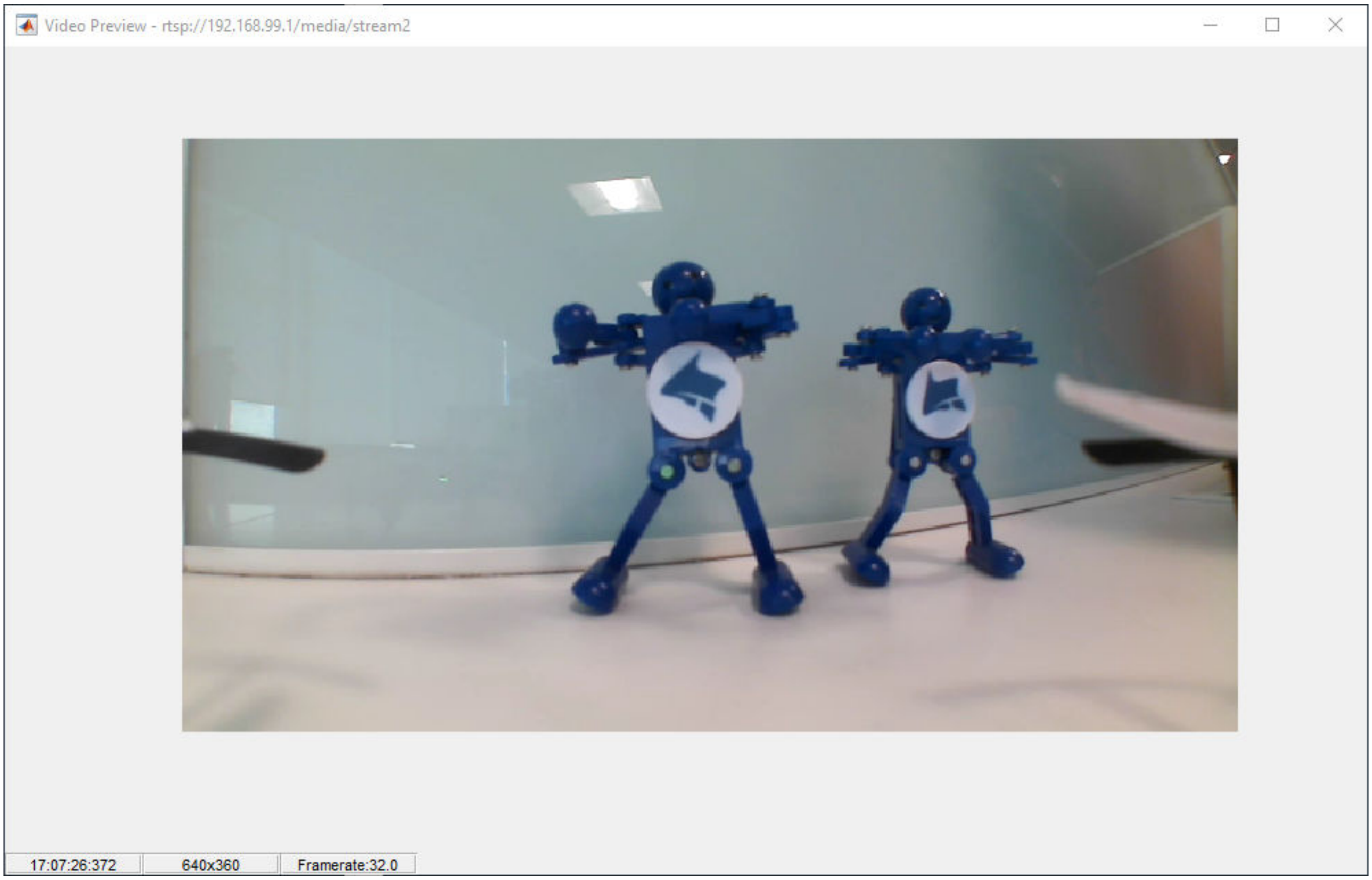
```
parrotObj = parrot('Mambo');
```

Connect to the FPV camera

```
cameraObj = camera(parrotObj,'FPV');
```

Start video streaming from the FPV camera

```
preview(cameraObj)
```



The preview window opens and displays live video stream from your camera. The banner of the preview window shows the camera URL. The lower portion of the window shows the timestamp in seconds, resolution, and frame rate in frames per second.

You can use the `snapshot` function to acquire images from the video stream.

Input Arguments

`cameraObj` — Parrot drone camera connection

`camera` object

Parrot drone camera connection object, specified as a `camera` on page 2-2 object.

See Also

`snapshot` | `camera`

Introduced in R2019a

readHeight

Read current height of Parrot drone

Syntax

```
[height,time] = readHeight(parrotObj)
```

Description

`[height,time] = readHeight(parrotObj)` returns the current height above the takeoff surface in meters along with the time stamp of the Parrot drone, specified as a parrot object. The function returns the height only after the drone has taken off. For a Parrot Mambo drone, before takeoff, the function returns a zero.

Examples

Read Height of Parrot Drone

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, read the height from the takeoff surface.

```
[height,time] = readHeight(parrotObj)
```

```
height = 
        0.8600
time = 
       datetime
       15-Mar-2019 14:07:19
```

Input Arguments

`parrotObj` — Parrot drone connection
`parrot` object

Parrot drone connection object, specified as a `parrot` on page 2-5 object.

Output Arguments

`height` — Current height of drone

positive real scalar

The current height of the Parrot drone above the takeoff surface, specified in meters.

Data Types: `double`

`time` — Timestamp from drone

`datetime`

The time at which the drone sends the measured height, specified as a datetime.

Data Types: `datetime`

See Also

`readOrientation` | `readSpeed`

Introduced in R2019a

readOrientation

Read Euler angles of Parrot drone

Syntax

```
[eulerAngles,time] = readOrientation(parrotObj)
```

Description

`[eulerAngles,time] = readOrientation(parrotObj)`returns the Euler angles that is the azimuth, the pitch, and the roll of Parrot drone, specified as a parrot object, along with the time stamp. For a Parrot Mambo drone, before takeoff, the function returns a zero.

Examples

Read orientation of Parrot Drone

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, read the orientation

```
[eulerAngles,time]= readOrientation(parrotObj)
```

```
eulerAngles =
0.785  0.301  -0.207
time =
     datetime
      15-Mar-2019 14:07:19
```

Input Arguments

`parrotObj` — Parrot drone connection
`parrot` object

Parrot drone connection object, specified as a `parrot` on page 2-5 object.

Output Arguments

`eulerAngles` — Euler rotation angles

1-by-3 vector (default)

Euler rotation angles in radians , returned as an 1-by-3 array of Euler rotation angles. The axis is along ZYX axes. This represents the rotation of the Parrot drone from the NED frame to the estimated body frame.

Data Types: `double`

`time` — Timestamp from drone

`datetime` (default)

The time at which the drone sends the orientation of the Parrot drone, specified as a datetime.

Data Types: `datetime`

See Also

`readHeight` | `readSpeed`

Introduced in R2019a

readSpeed

Read speed of Parrot drone

Syntax

```
[speed,time] = readSpeed(parrotObj)
```

Description

`[speed,time] = readSpeed(parrotObj)` returns the speed of the Parrot drone along with the time stamp. The function returns the speed only after the drone has taken off. For a Parrot Mambo drone, before takeoff, the function returns a zero.

Examples

Read Speed of Parrot Drone

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, read the speed of the drone.

```
[speed,time] = readSpeed(parrotObj)
```

```
speed = 
        0.8600  -0.0240  0.0015
 time = 
        datetime
          15-Mar-2019 14:07:19
```

Input Arguments

`parrotObj` — Parrot drone connection
`parrot` object

Parrot drone connection object, specified as a `parrot` on page 2-5 object.

Output Arguments

speed — Speed of drone

1-by-3 vector (default)

The speed of the drone in m/s along the *x*-, *y*-, and *z*- directions with respect to the inertial NED frame. The NED frame is calculated at drone startup.

Data Types: `double`

`time` — Time stamp from drone

`datetime` (default)

The time at which the drone sends the speed of the Parrot drone, specified as a datetime.

Data Types: `datetime`

See Also

`readOrientation` | `readHeight`

Introduced in R2019a

snapshot

Acquire single image frame from Parrot drone FPV camera

Syntax

```
frame = snapshot(cameraObj)
[frame,ts] = snapshot(cameraObj)
```

Description

`frame = snapshot(cameraObj)` returns a single image from the Parrot drone camera object, specified as `cameraObj`. Calling `snapshot` in a loop returns a new frame for each iteration of the loop.

`[frame,ts] = snapshot(cameraObj)` acquires a single image from the Parrot drone camera object, specified as `cameraObj`, assigns it to the variable `frame`, and returns the timestamp `ts`.

Examples

Capture Image from FPV camera

Connect to a Parrot Mambo FPV drone over a wireless network.

```
parrotObj = parrot('Mambo')
```

```
parrotObj =
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_650559"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Connect to the FPV camera

```
cameraObj = camera(parrotObj,'FPV')
```

```
cameraObj =
          camera with properties:

                    Name: "FPV"
              Resolution: "640x360"
```

Capture an image from the FPV camera

```
frame = snapshot(cameraObj);
```

Acquire One Image Frame and Timestamp from FPV Camera

Connect to a Parrot Mambo FPV drone over a wireless network.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_650559"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Connect to the FPV camera

```
cameraObj = camera(parrotObj,'FPV')
```

```
cameraObj = 
          camera with properties:

                    Name: "FPV"
              Resolution: "640x360"
```

Capture an image from the FPV camera and assign it to `frame`, acquire the timestamp, and assign it to `ts`

```
[frame,ts] = snapshot(cameraObj);
```

Display the acquired image

```
imshow(frame)
```

Display the timestamp of the snapshot

```
ts
```

```
time = 
        datetime
          15-Mar-2019 14:07:19
```

Input Arguments

`cameraObj` — Parrot drone camera connection

`camera` object

Parrot drone camera connection object, specified as a `camera` on page 2-2 object.

Output Arguments

`frame` — Image captured by FPV camera

360-by-640-by-3 matrix

The RGB image captured by the FPV camera of the drone, specified as a real-valued matrix.

See Also

`preview` | `camera`

Introduced in R2019a

takeoff

Initiate Parrot drone takeoff

Syntax

```
takeoff(parrotObj)
```

Description

`takeoff(parrotObj)` initiates the takeoff of a Parrot drone, represented by `parrotObj`.

Examples

Initiate Parrot Drone Takeoff

Connect to a Parrot drone.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

Input Arguments

`parrotObj` — Parrot drone connection
`parrot` object

Parrot drone connection object, specified as a parrot on page 2-5 object.

See Also

`parrot` | `abort` | `land`

Introduced in R2019a

turn

Turn Parrot drone at specified angle

Syntax

```
turn(parrotObj,angle)
```

Description

`turn(parrotObj,angle)` turns the Parrot drone, represented by `parrotObj`, by the angle specified.

Examples

Turn Drone at Specified Angle

Connect to a Parrot drone.

```
parrotObj = parrot()
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Use the `parrot` object to initiate takeoff of the Parrot drone.

```
takeoff(parrotObj)
```

While the Parrot drone is in flight, turn the drone by 0.7854 radians (45 degrees) in the clockwise direction

```
turn(parrotObj,deg2rad(45));
```

Input Arguments

`parrotObj` — Parrot drone connection
`parrot` object

Parrot drone connection object, specified as a `parrot` on page 2-5 object.

`angle` — Relative angle in radians
real number in the interval [-pi, pi]

The angle relative to the line determined by the direction the drone is facing by which the Parrot drone turns, specified in radians. The drone moves in the clockwise direction if `angle` is positive. If the `angle` is negative, the drone moves in the counterclockwise direction.

Data Types: `double`

See Also

`takeoff` | `land` | `abort` | `flip`

Introduced in R2019a

Getting Started with MATLAB® Support Package for Parrot® Drones

This example shows how to use the MATLAB® Support Package for Parrot® drone to perform the basic flight operations on the drone such as take-off, land, and fly along a path.

Introduction

The MATLAB Support Package for Parrot Drone enables you to control a Parrot drone from a computer running MATLAB.

The support package includes functions to pilot a Parrot drone by sending the commands to control its direction, speed, and orientation, and read the flight navigation data such as speed, height, and orientation.

In this example you will learn how to create a parrot object to control and fly the Parrot drone from within MATLAB.

Pre-requisites

If you are new to MATLAB, it is helpful to read the Getting Started section of the MATLAB documentation and running Getting Started with MATLAB example.

Required Hardware

To run this example you need the following:

- A fully charged Parrot FPV drone
- A computer with a WiFi connection

Important pre-flight safety considerations

Before flying the Parrot drone, ensure the following safety procedures:

- Ensure the safety of people, animals, and property in the vicinity of the flight.
- Wear safety glasses at all times.
- Place the drone on a flat surface before starting.
- Fly the drone only indoors, with an open area greater than 10x10 feet, over a non-glossy floor.

Task 1 — Hardware setup

- Power on the Parrot FPV drone, wait for the LEDs on the camera to stabilize.
- Connect your computer to the drone's Wifi network.

Task 2 — Create a parrot object

Create a `parrot` object.

```
p = parrot();
```

Task 3 — Take-off and land the drone

Take off the Parrot FPV drone from a level surface.

Execute the following command at the MATLAB command prompt the takeoff of the drone.

```
takeoff(p);
```

The Parrot drone moves up vertically, and remains there.

Land the drone.

```
land(p);
```

Task 4 — Fly the drone along a square path

Take-off and move the drone forward for 2 seconds and turn the drone by pi/2 radians (90 degrees) at each square vertex.

Repeat this action 4 times (vertices of a square) to make the drone navigate a square path and return it to the starting position.

Use the BatteryLevel property of the drone to ensure there is enough battery charge left for flight.

```
takeoff(p);
movement_step = 1;
 while(movement_step <= 4 && p.BatteryLevel > 10)
  moveforward(p, 2);
  turn(p, deg2rad(90));
  movement_step = movement_step + 1;
 end
```

You can also increase the duration argument in moveforward function to make the drone move forward for more time. Use the optional tilt argument in the moveforward function to vary the speed of drone.

Land the drone.

```
land(p);
```

Task 5 — Fly the drone along a circlular path

Take-off the drone and control the Roll angle and RotationSpeed of the drone to make the drone fly along the perimeter of a circle.

Execute the following command at the MATLAB command prompt to fly the drone in a circle for 5 seconds.

```
takeoff(p);
move(p, 5, 'Roll', deg2rad(4), 'RotationSpeed', deg2rad(120));
```

Vary the value of RotationSpeed NV pair to adjust the speed of drone revolution.

Land the drone.

```
land(p);
```

Task 6 — Fly the drone along a diagonal path

Take-off and move the drone along a diagonal path in the horizontal plane by adjusting the Pitch and Roll angles.

Execute the following command at the MATLAB command prompt to fly the drone along a diagonal path for 5 seconds.

```
takeoff(p);
move(p, 5, 'Pitch', deg2rad(-4), 'Roll', deg2rad(4));
```

Vary the `Pitch` and `Roll` NV pairs to adjust the speed of the drone.

Land the drone.

```
land(p);
```

Task 7 — Clean up

When finished clear the parrot object

```
clear p;
```

Read and Plot Navigation data Using MATLAB; Support Package for Parrot; Drones

This example shows how to use the MATLAB® Support Package for Parrot® drone to acquire and plot real-time navigation data of the Parrot drone

Introduction

The MATLAB Support Package for Parrot Drones enables you to control and read the in flight navigation data of the drone.

In this example, you will learn to read navigation data of the Parrot drone such as the speed, orientation, and height using MATLAB commands.

Prerequisites

Complete "Getting Started with MATLAB® Support Package for Parrot® Drones" on page 1-46.

Required Hardware

To run this example you need the following:

- A fully charged Parrot FPV drone
- A computer with a WiFi connection

Task 1 — Hardware setup

- Power on the Parrot FPV drone.
- Connect your computer to the drone's Wifi network.

Task 2 — Create a parrot object

Create a `parrot` object.

```
p = parrot();
```

Task 3 — Take-off the drone

Start the Parrot FPV drone flight from a level surface.

Execute the following command at the MATLAB command prompt the takeoff of the drone.

```
takeoff(p);
```

Task 3 — Initialize MATLAB `animatedline` and `figure` window properties

This task shows you how to initialize MATLAB to plot the navigation data.

Use MATLAB `animatedline` to plot the variation in speed along the X, Y, and Z axes, separately.

Initialize the figure handle and create animated line instances hx, hy, and hz corresponding to speeds along the X, Y, and Z axes, respectively.

```
f = figure;
hx = animatedline('Color', 'r', 'LineWidth', 2);
hy = animatedline('Color', 'g', 'LineWidth', 2);
```

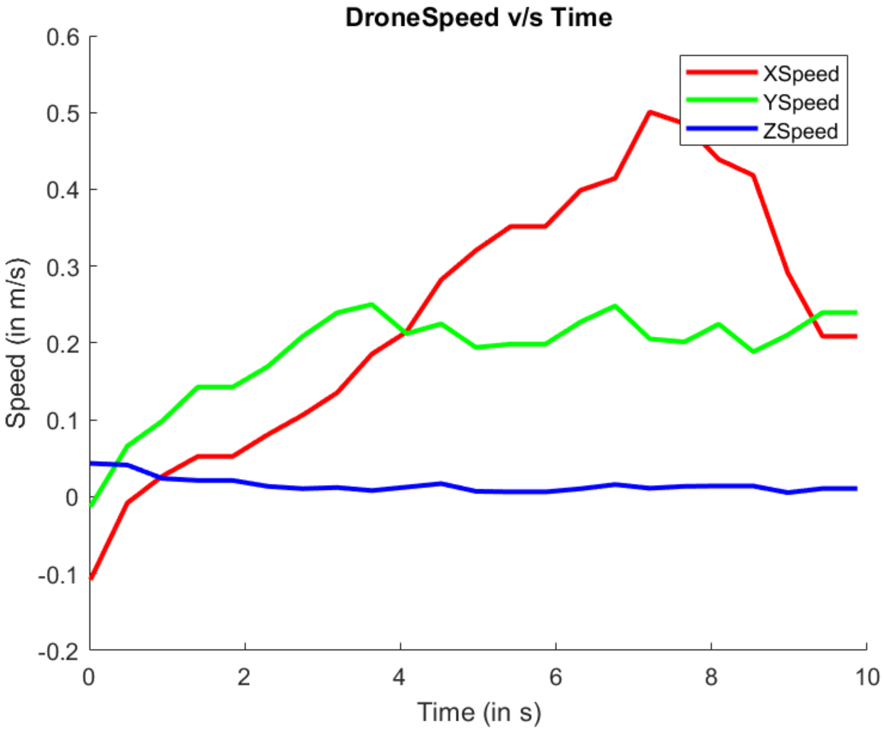
```
   hz = animatedline('Color', 'b', 'LineWidth', 2);
   title('DroneSpeed v/s Time');
   xlabel('Time (in s)');
   ylabel('Speed (in m/s)');
   legend('XSpeed', 'YSpeed', 'ZSpeed');
```

Task 4 — Plot Navigation data during drone flight

Keep flying the drone along the desired path (forward diagonal path in this example) for 10 seconds and plot navigation data (speed) during this flight.

The default value of `duration` in `move` function is 0.5 seconds.

```
  flightTime = 10;
  tObj = tic;
  while(p.BatteryLevel > 10 && toc(tObj) < flightTime)
      move(p, 'Pitch', deg2rad(-4), 'Roll', deg2rad(4));
      speed = readSpeed(p);
      tStamp = toc(tObj);
      addpoints(hx, tStamp, speed(1));
      addpoints(hy, tStamp, speed(2));
      addpoints(hz, tStamp, speed(3));
      drawnow;
      pause(0.1);
 end
```



Task 5 — Land the drone

Land the drone.

```
land(p);
```

Task 6 — Clean up

When finished, clear the connection to the Parrot drone.

```
clear p;
```

Image Classification Using Parrot FPV Drones

This example shows you how to use the MATLAB® Support Package for Parrot® Drones to classify images captured by the drone's FPV camera.

Introduction

The MATLAB® Support Package for Parrot® Drones enables you to control the Parrot drone and capture images from the first person view (FPV) camera. The images captured by the drone's FPV camera can be classified using GoogLeNet, a pretrained deep convolutional neural network. GoogLeNet is trained on more than a million images from ImageNet database. It takes the image as input and provides a label for the object in the image.

Required MathWorks Products

- MATLAB®
- MATLAB® Support Package for Parrot® Drones
- Deep Learning Toolbox™
- Image Processing Toolbox™

Prerequisites

Complete “Getting Started with MATLAB® Support Package for Parrot® Drones” on page 1-46.

Required Hardware

To run this example you need:

- A fully charged Parrot FPV drone
- A computer with a WiFi connection

Task 1 — Create a Connection to the Parrot Drone

Create a `parrot` object.

```
parrotObj = parrot;
```

Task 2 — Create the GoogLeNet Neural Network Object

Create a GoogLeNet neural network object.

```
nnet = googlenet;
```

Task 3 — Activate FPV Camera

Start the drone flight and activate the FPV camera.

```
takeoff(parrotObj);
```

Create a connection to the drone's FPV camera.

```
camObj = camera(parrotObj, 'FPV');
```

Task 4 — Capture and Classify the Object in the Image

Move the drone forward for 2 seconds along the edges of a square path. Capture the image of an object, and classify it while the drone moves forward.

1 Move the drone forward for the default duration of 0.5 seconds for each forward step, ensuring a nonblocking behavior. This enables the drone to capture the image and classify it while in motion.

2 Capture a single frame from the drone's FPV camera.

3 Resize the image and classify the object in image using the neural network.

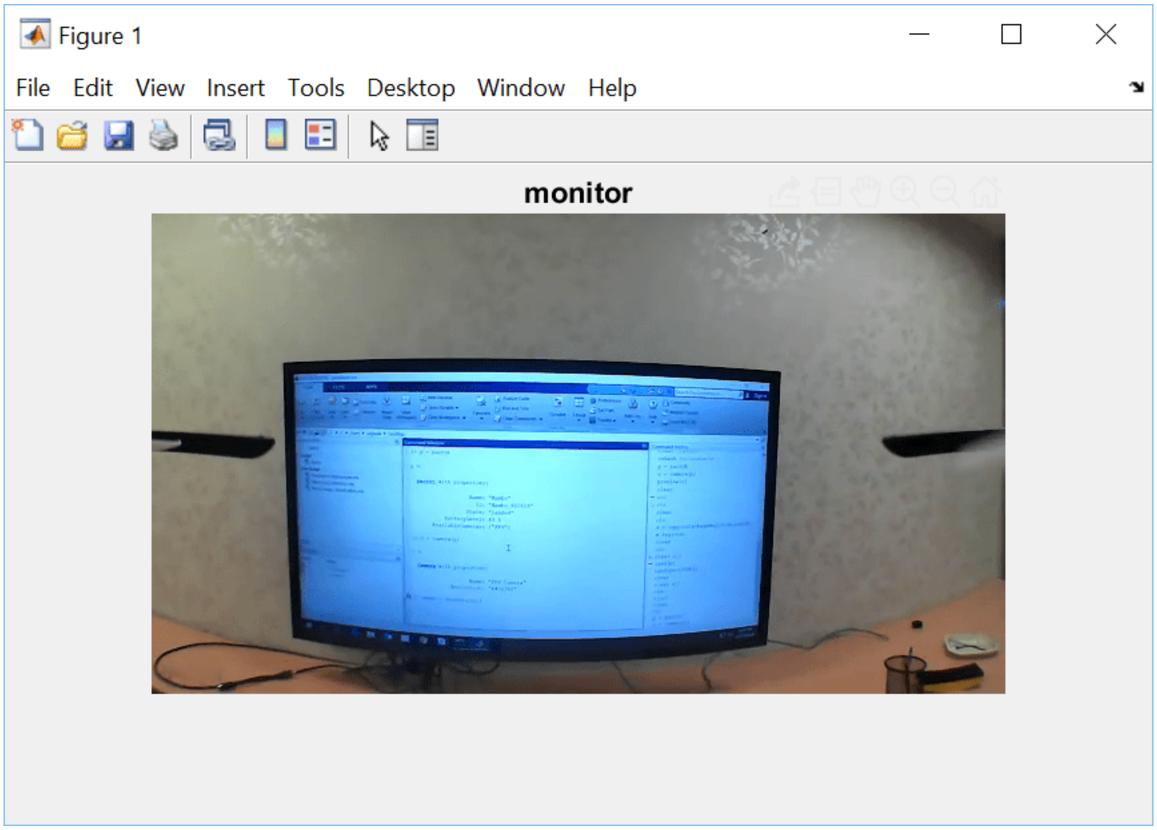
4 Display the image with title as the label returned by the classify function.

5 Turn the drone by π/2 radians at each square vertex.

```
tOuter= tic;
while(toc(tOuter)<=30 && parrotObj.BatteryLevel>20)
    tInner = tic;
    % Keep moving the drone for 2 seconds along each square path edge
    while(toc(tInner)<=2)
        moveforward(parrotObj);                        % Move the drone forward for default time
        picture = snapshot(camObj);                    % Capture image from drone's FPV camera
        resizedPicture = imresize(picture,[224,224]);  % Resize the picture
        label = classify(nnet,resizedPicture);         % Classify the picture
        imshow(picture);                               % Show the picture
        title(char(label));                            % Show the label
        drawnow;
   end
    turn(parrotObj,deg2rad(90));                       % Turn the drone by pi/2 radians
end
```

6 Execute steps 1–5 for 30 seconds.

For example, the drone classifies a monitor screen as captured by the FPV camera.



Task 5 — Land the Drone

Land the drone.

```
land(parrotObj);
```

Task 6 — Clean Up

When finished clear the connection to the Parrot drone, the FPV camera, and GoogLeNet

```
clear parrotObj;
clear camObj;
clear nnet;
```

Face Detection Using Parrot FPV Drones

This example shows how to use a Parrot® drone to automatically detect human faces captured by the drone's FPV camera.

Introduction

Use the MATLAB® Support Package for Parrot® Drones to control the drone and capture images from the FPV camera. A cascade object detector uses the Viola-Jones detection algorithm and a trained classification model for face detection. By default, the detector is configured to detect faces, but it can be used to detect other types of objects.

Required MathWorks Products

- MATLAB®
- MATLAB® Support Package for Parrot® Drones
- Computer Vision System Toolbox™

Prerequisites

Complete “Getting Started with MATLAB® Support Package for Parrot® Drones” on page 1-46.

Required Hardware

To run this example you need:

- A fully charged Parrot FPV drone
- A computer with a WiFi connection

Task 1 — Create a Connection to the Parrot Drone

Create a `parrot` object.

```
parrotObj = parrot;
```

Task 2 — Create a Cascade Object Detector Instance

Create an instance of the cascade object detector to detect faces using the Viola-Jones algorithm.

```
detector = vision.CascadeObjectDetector;
```

Task 3 — Activate FPV Camera

Start the drone flight to activate the FPV camera. Move the drone up to sufficient height to capture faces.

```
takeoff(parrotObj);
moveup(parrotObj,1);
```

Task 4 — Create a Connection to the Drone's FPV Camera

Use the parrot object from Task 1 to create the connection to the drone's FPV camera.

```
camObj = camera(parrotObj,'FPV');
```

Task 5 — Detect Faces While Traversing a Square Path

Detect faces while the drone moves forward for 2 seconds along the edge of a square path.

1 Move the drone forward for the default duration of 0.5 seconds for each forward step, ensuring a nonblocking behaviour. This enables the drone to capture the image and detect faces while in motion.

2 Capture a single frame from the drone's FPV camera.

3 Input the image to the detector, which returns bounding boxes containing the detected objects. The detector performs multiscale object detection on the input image.

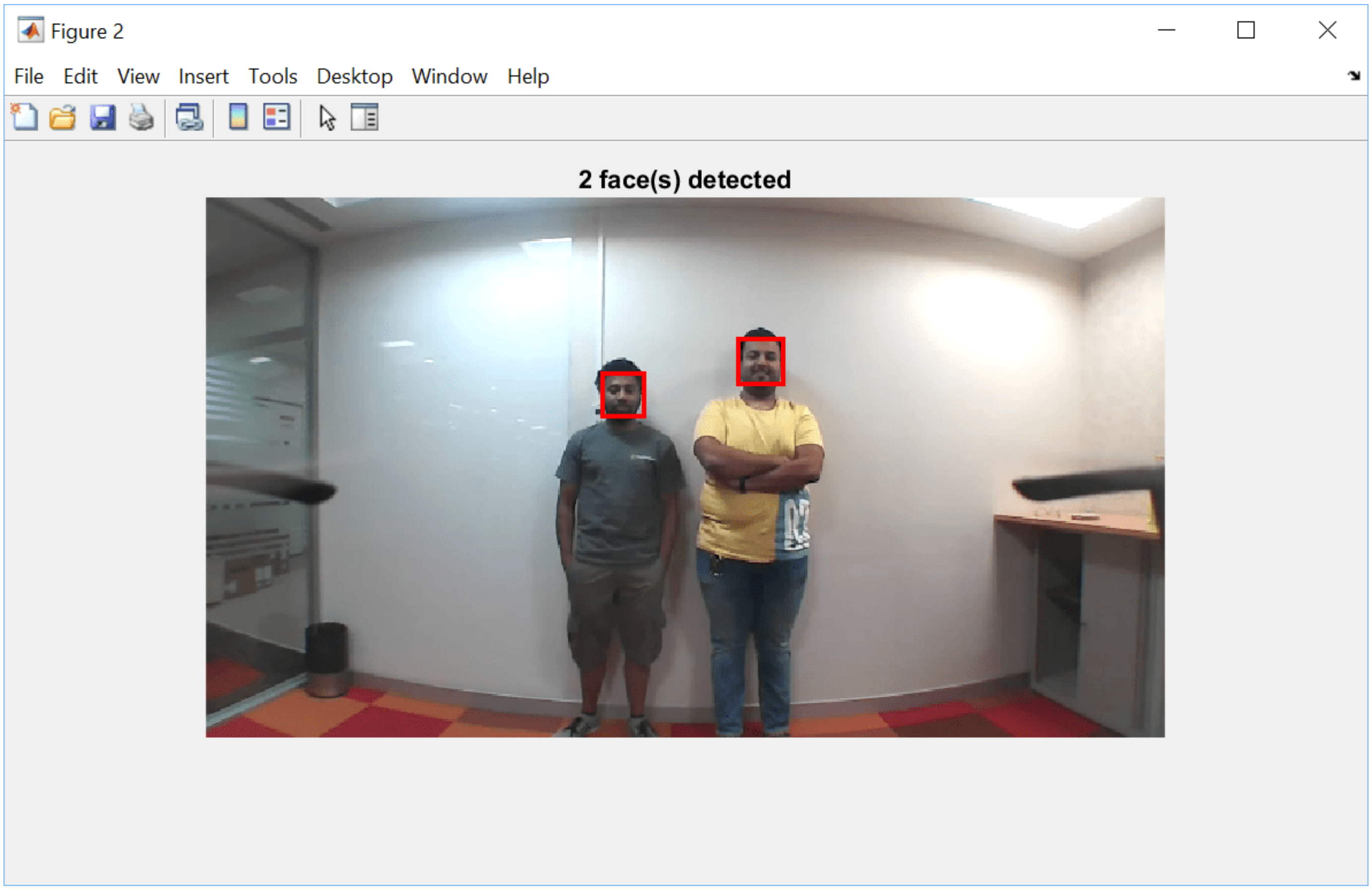
4 Display the image with bounding boxes around faces and the title displaying the number of faces detected.

5 Turn the drone by π/2 radians at each square vertex.

```
tOuter= tic;
while(toc(tOuter)<=30 && parrotObj.BatteryLevel>20)
   tInner = tic;
   % Keep moving the drone for 2 seconds along each square path edge
   while(toc(tInner)<=2)
       moveforward(parrotObj);                                                       % Move the d
       picture = snapshot(camObj);                                                   % Capture in
       bbox = detector(picture);                                                     % Detect fac
       videoOut = insertShape(picture,'Rectangle',bbox,'Color','r','LineWidth',3);   % Insert bou
       imshow(picture);                                                              % Show the p
       title(sprintf(' %d face(s) detected ',size(bbox,1)));
       drawnow;
   end
   turn(parrotObj,deg2rad(90));                                                      % Turn the d
end
```

6 Execute steps 1–5 for 30 seconds.

This example show two faces detected by the drone's FPV camera.



2 face(s) detected

Task 6 — Land the Drone

Land the drone.

```
land(parrotObj);
```

Task 7 — Clean Up

When finished clear the connection to the Parrot drone and the FPV camera.

```
clear parrotObj;
clear camObj;
```

2

Objects

camera

Connection to Parrot drone FPV camera

Description

This object represents a connection to the Parrot drone's first-person view (FPV) camera. To acquire images from the Parrot drone camera, use this object with the functions listed in “Object Functions” on page 2-3.

Note The Parrot Mambo FPV drone's FPV camera enters a low power mode, to save battery and prevent over heating if the drone is idle in the landed state for more than 3 minutes. Creating a new camera connection fails when the FPV camera is in low power mode. The `snapshot` and the `preview` functions time out with an already existing camera connection. To ensure the camera is activated, take off the drone before creating a new camera connection (See `takeoff` on page 1-43).

Creation

Syntax

```
cameraObj = camera(parrotObj)
cameraObj = camera(parrotObj,droneCamera)
```

Description

`cameraObj = camera(parrotObj)` creates a camera object that connects to the camera of the Parrot drone, represented by `parrotObj`.

`cameraObj = camera(parrotObj,droneCamera)` creates a camera object that connects to the specified camera of the Parrot drone, represented by `parrotObj`.

Input Arguments

`parrotObj` — Parrot drone connection
`parrot` object

Parrot drone connection object, specified as a `parrot` object.

`droneCamera` — Drone camera to connect
`"FPV"` (default)

Name of the Parrot drone camera, specified as one of array values in the `AvailableCameras` property of the `parrot` object.

Properties

`Name` — Name of camera
string

This property is read-only.

Name of the connected Parrot drone camera, specified as a string.

Example: "FPV"

Data Types: string

Resolution — Video resolution
string

This property is read-only.

The video resolution of the incoming video stream of the current cameraObj, returned as a string.

Example: "640x360"

Data Types: string

Object Functions

The object functions are used to interact with your Parrot drone camera

preview	Preview live video data from Parrot drone FPV camera
snapshot	Acquire single image frame from Parrot drone FPV camera
closePreview	Close Parrot drone FPV camera preview window

Examples

Connect to Parrot Drone Camera

Connect to a Parrot drone by creating a parrot object..

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Connect to the drone camera using the camera object

```
cameraObj = camera(parrotObj)
```

```
cameraObj = 
          Camera with properties:

                    Name: "FPV"
              Resolution: "640x360"
```

Connect to Specified Parrot Drone Camera

Connect to a Parrot drone by creating a `parrot` object..

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Connect to the FPV camera using the `camera` object

```
cameraObj = camera(parrotObj,'FPV')
```

```
cameraObj = 
          Camera with properties:

                    Name: "FPV"
              Resolution: "640x360"
```

See Also

`parrot` | `preview` | `snapshot`

Introduced in R2019a

parrot

Connection to Parrot drone

Description

This object represents a connection from MATLAB to the Parrot drone. To interact with the Parrot drone, use this object with the functions listed in “Object Functions” on page 2-6.

Note MATLAB Support Package for Parrot Drones supports only the Parrot shipped factory firmware. To restore the original firmware, see “Restore Original Firmware”.

Creation

Syntax

```
parrotObj = parrot
parrotObj = parrot(droneName)
parrotObj = parrot(droneID)
```

Description

`parrotObj = parrot` connects to the first available Parrot drone over the wireless network.

`parrotObj = parrot(droneName)` connects to the Parrot drone with the specified name over the wireless network.

`parrotObj = parrot(droneID)` connects to a Parrot drone with a specific ID over the wireless network.

Input Arguments

`droneName` — Name of the Parrot drone
string

Name of the Parrot drone.

Parrot Drones	String used to create the `parrot` object
Parrot Mambo FPV	`'Mambo'`
Parrot Bebop 2	`'Bebop2'`

`droneID` — Unique ID of the Parrot drone
string

ID of the Parrot drone.

Properties

`Name` — Name of drone

string

Name of the connected Parrot drone, specified as a string.

Example: `"Mambo"`

Data Types: `string`

`ID` — Unique ID of drone

string

The ID of the specific Parrot drone returned as a string.

Example: `"Mambo_564853"`

Data Types: `string`

`State` — Piloting state of drone

`landed` | `landing` | `takingoff` | `hovering` | `emergency` | `flying` | `initializing`

This property is read-only.

The piloting state of the Parrot drone, specified as a string.

Example: `"landed"`

Data Types: `string`

`BatteryLevel` — Current battery level of drone

non negative integer

This property is read-only.

Current battery level of the drone, specified as a percentage.

Example: `50%`

Data Types: `double`

`AvailableCameras` — Available cameras of drone

`"FPV"`

Available cameras of drone, specified as a cell array

Example: `["FPV"]`

Data Types: `struct`

Object Functions

The object functions are used to interact with your Parrot drone

abort	End flight of Parrot drone
flip	Flip Parrot drone in specified direction
land	Land Parrot drone
move	Move Parrot drone in all six directions

moveback	Move Parrot drone backwards
movedown	Move Parrot drone down
moveforward	Move Parrot drone forward
moveleft	Move Parrot drone left
moveright	Move Parrot drone right
moveup	Move Parrot drone upwards
readHeight	Read current height of Parrot drone
readOrientation	Read Euler angles of Parrot drone
readSpeed	Read speed of Parrot drone
takeoff	Initiate Parrot drone takeoff
turn	Turn Parrot drone at specified angle

Examples

Connect to Parrot Drone

Connect to a Parrot drone.

```
parrotObj = parrot()
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Connect to Specific Parrot Drone

Connect to a Parrot Mambo FPV drone over a wireless network.

```
parrotObj = parrot('Mambo')
```

```
parrotObj = 
          parrot with properties:

                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

Connect to Parrot Drone with Specific ID

Connect to a specific Parrot Mambo FPV drone with the ID over a wireless network.

```
parrotObj = parrot('Mambo_564853')
```

```
parrotObj = 
          parrot with properties:
```

```
                    Name: "Mambo"
                      ID: "Mambo_564853"
                   State: "landed"
            BatteryLevel: 50%
        AvailableCameras: ["FPV"]
```

See Also

`abort` | `flip` | `land` | `takeoff`

Introduced in R2019a