

---

# **guibot Documentation**

*Release 0.11*

**Plamen Dimitrov and Thomas Jarosch**

**Jun 29, 2018**



<b>1</b>	<b>guibot package</b>	<b>1</b>
1.1	Submodules	1
1.1.1	guibot.calibrator module	1
1.1.2	guibot.config module	3
1.1.3	guibot.desktopcontrol module	6
1.1.4	guibot.errors module	11
1.1.5	guibot.finder module	12
1.1.6	guibot.guiobot module	22
1.1.7	guibot.guiobot_proxy module	23
1.1.8	guibot.guiobot_simple module	29
1.1.9	guibot.imagelogger module	30
1.1.10	guibot.inputmap module	32
1.1.11	guibot.location module	34
1.1.12	guibot.match module	34
1.1.13	guibot.path module	36
1.1.14	guibot.region module	37
1.1.15	guibot.target module	45
1.2	Module contents	49
	<b>Python Module Index</b>	<b>51</b>



## 1.1 Submodules

### 1.1.1 guibot.calibrator module

**class** `guibot.calibrator.Calibrator` (*needle=None, haystack=None, config=None*)

Bases: `object`

Provides with a group of methods to facilitate and automate the selection of algorithms and parameters that are most suitable for a given preselected image matching pair.

Use the benchmarking method to choose the best algorithm to find your image. Use the calibration method to find the best parameters if you have already chosen the algorithm. Use the search method to find the best parameters from multiple random starts from a uniform or normal probability distribution.

**\_\_init\_\_** (*needle=None, haystack=None, config=None*)

Build a calibrator object for a given match case.

#### Parameters

- **haystack** (`target.Image` or `None`) – image to look in
- **needle** (`target.Target` or `None`) – target to look for

**benchmark** (*finder, random\_starts=0, uniform=False, calibration=False, max\_attempts=3, \*\*kwargs*)

Perform benchmarking on all available algorithms of a finder for a given needle and haystack.

#### Parameters

- **finder** (`finder.Finder`) – CV backend whose backend algorithms will be benchmarked
- **random\_starts** (*int*) – number of random starts to try with (0 for nonrandom)
- **uniform** (*bool*) – whether to use uniform or normal distribution
- **calibration** (*bool*) – whether to use calibration

- **max\_attempts** (*int*) – maximal number of refinements to reach the parameter delta below the tolerance

**Returns** list of (method, similarity, location, time) tuples sorted according to similarity

**Return type** [(str, float, location.Location, float)]

---

**Note:** Methods that are supported by OpenCV and others but currently don't work are excluded from the dictionary. The dictionary can thus also be used to assess what are the available and working methods besides their success for a given *needle* and *haystack*.

---

**search** (*finder*, *random\_starts=1*, *uniform=False*, *calibration=True*, *max\_attempts=3*, *\*\*kwargs*)

Search for the best match configuration for a given needle and haystack using calibration from random initial conditions.

**Parameters**

- **finder** (*finder.Finder*) – CV backend to use in order to determine deltas, fixed, and free parameters and ultimately tweak to minimize error
- **random\_starts** (*int*) – number of random starts to try with
- **uniform** (*bool*) – whether to use uniform or normal distribution
- **calibration** (*bool*) – whether to use calibration
- **max\_attempts** (*int*) – maximal number of refinements to reach the parameter delta below the tolerance

**Returns** maximized similarity

**Return type** float

If normal distribution is used, the mean will be the current value of the respective CV parameter and the standard variation will be determined from its delta.

**calibrate** (*finder*, *max\_attempts=3*, *\*\*kwargs*)

Calibrate the available match configuration for a given needle and haystack minimizing the matchign error.

**Parameters**

- **finder** (*finder.Finder*) – configuration for the CV backend to calibrate
- **max\_attempts** (*int*) – maximal number of refinements to reach the parameter delta below the tolerance

**Returns** maximized similarity

**Return type** float

This method calibrates only parameters that are not protected from calibration, i.e. that have *fixed* attribute set to false. In order to set all parameters of a background algorithm for calibration use the `finder.Finder.can_calibrate()` method first. Any parameter values will only be changed if they improve the similarity, i.e. minimize the error. The deltas of the final parameters will represent the maximal flat regions in positive and/or negative direction where the same error is still obtained.

---

**Note:** All similarity parameters will be reset to 0.0 after calibration and can be set by client code afterwards.

---

---

**Note:** Special credits for this approach should be given to Prof. Sebastian Thrun, who explained it in his Artificial Intelligence for Robotics class.

---

**run\_default** (*finder*, *\*\*kwargs*)

Run a match case and return error from the match as dissimilarity.

**Parameters** **finder** (*finder.Finder*) – finder with match configuration to use for the run

**Returns** error obtained as unity minus similarity

**Return type** float

**run\_performance** (*finder*, *\*\*kwargs*)

Run a match case and return error from the match as dissimilarity and linear performance penalty.

**Parameters**

- **finder** (*finder.Finder*) – finder with match configuration to use for the run
- **max\_exec\_time** (*float*) – maximum execution time before penalizing the run by increasing the error linearly

**Returns** error obtained as unity minus similarity

**Return type** float

**run\_peak** (*finder*, *\*\*kwargs*)

Run a match case and return error from the match as failure to obtain high similarity of one match and low similarity of all others.

**Parameters**

- **finder** (*finder.Finder*) – finder with match configuration to use for the run
- **peak\_location** (*(int, int)*) – (x, y) of the match whose similarity should be maximized while all the rest minimized

**Returns** error obtained as unity minus similarity

**Return type** float

This run function doesn't just obtain the optimum similarity for the best match in each case of needle and haystack but it minimizes the similarity for spatial competitors where spatial means other matches in the same haystack. Keep in mind that since matching is performed with zero similarity requirement, such matches might not be anything close to the needle. This run function finds use cases where the other matches could resemble the best one and we want to find configuration to better discriminate against those.

## 1.1.2 guibot.config module

**class** `guibot.config.GlobalConfig`

Bases: `object`

Handler for default configuration present in all cases where no specific value is set.

The methods of this class are shared among all of its instances.

**click\_delay** = `None`

time interval between two clicks in a double click

**delay\_after\_drag** = `None`

timeout before drag operation

**delay\_before\_drop = None**  
timeout before drop operation

**delay\_before\_keys = None**  
timeout before key press operation

**delay\_between\_keys = None**  
time interval between two consecutively typed keys

**rescan\_speed\_on\_find = None**  
time interval between two image matching attempts (used to reduce overhead on the CPU)

**smooth\_mouse\_drag = None**  
whether to move the mouse cursor to a location instantly or smoothly

**screen\_autoconnect = None**  
whether to perform complete initialization of the desktop control backend

**preprocess\_special\_chars = None**  
whether to preprocess capital and special characters and handle them internally

**save\_needle\_on\_error = None**  
whether to perform an extra needle dump on matching error

**image\_logging\_level = None**  
logging level similar to the python logging module

**image\_logging\_step\_width = None**  
number of digits when enumerating the image logging steps, e.g. value=3 for 001, 002, etc.

**image\_logging\_destination = None**  
relative path of the image logging steps

**desktop\_control\_backend = None**  
name of the desktop control backend

**find\_backend = None**  
name of the computer vision backend

**contour\_threshold\_backend = None**  
name of the contour threshold backend

**template\_match\_backend = None**  
name of the template matching backend

**feature\_detect\_backend = None**  
name of the feature detection backend

**feature\_extract\_backend = None**  
name of the feature extraction backend

**feature\_match\_backend = None**  
name of the feature matching backend

**text\_detect\_backend = None**  
name of the text detection backend

**text\_ocr\_backend = None**  
name of the optical character recognition backend

**hybrid\_match\_backend = None**  
name of the hybrid matching backend for unconfigured one-step targets



`__metaclass__`  
alias of `GlobalConfig`

**class** `guiobot.config.LocalConfig` (*configure=True, synchronize=True*)

Bases: `object`

Container for the configuration of all desktop control and computer vision backends, responsible for making them behave according to the selected parameters as well as for providing information about them and the current parameters.

`__init__` (*configure=True, synchronize=True*)  
Build a container for the entire backend configuration.

#### Parameters

- **configure** (*bool*) – whether to also generate configuration
- **synchronize** (*bool*) – whether to also apply configuration

Available algorithms can be seen in the `algorithms` attribute whose keys are the algorithm types and values are the members of these types. The algorithm types are shortened as *categories*.

A parameter can be accessed as follows (example):

```
print self.params["control"]["vnc_hostname"]
```

**configure\_backend** (*backend=None, category='type', reset=False*)

Generate configuration dictionary for a given backend.

#### Parameters

- **backend** (*str or None*) – name of a preselected backend, see `algorithms[category]`
- **category** (*str*) – category for the backend, see `algorithms.keys()`
- **reset** (*bool*) – whether to (re)set all parent configurations as well

**Raises** `UnsupportedBackendError` if *backend* is not among the supported backends for the category (and is not *None*) or the category is not found

**configure** (*reset=True*)

Generate configuration dictionary for all backends.

**Parameters** **reset** (*bool*) – whether to (re)set all parent configurations as well

If multiple categories are available and just some of them are configured, the rest will be reset to defaults. To configure specific category without changing others, use `configure()`.

**synchronize\_backend** (*backend=None, category='type', reset=False*)

Synchronize a category backend with the equalizer configuration.

#### Parameters

- **backend** (*str or None*) – name of a preselected backend, see `algorithms[category]`
- **category** (*str*) – category for the backend, see `algorithms.keys()`
- **reset** (*bool*) – whether to (re)sync all parent backends as well

**Raises** `UnsupportedBackendError` if the category is not found

**Raises** `UninitializedBackendError` if there is no backend object that is configured with and with the required name

**synchronize** (*reset=True*)

Synchronize all backends with the current configuration dictionary.

**Parameters** `reset` (*bool*) – whether to (re)sync all parent backends as well

### 1.1.3 guibot.desktopcontrol module

**class** `guibot.desktopcontrol.DesktopControl` (*configure=True, synchronize=True*)

Bases: `guibot.config.LocalConfig`

Desktop control backend, responsible for performing desktop operations like mouse clicking, key pressing, text typing, etc.

**\_\_init\_\_** (*configure=True, synchronize=True*)

Build a desktop control backend.

**width**

Getter for readonly attribute.

**Returns** width of the connected screen

**Return type** `int`

**height**

Getter for readonly attribute.

**Returns** height of the connected screen

**Return type** `int`

**keymap**

Getter for readonly attribute.

**Returns** map of keys to be used for the connected screen

**Return type** `inputmap.Key`

**mousemap**

Getter for readonly attribute.

**Returns** map of mouse buttons to be used for the connected screen

**Return type** `inputmap.MouseButton`

**modmap**

Getter for readonly attribute.

**Returns** map of modifier keys to be used for the connected screen

**Return type** `inputmap.KeyModifier`

**mouse\_location**

Getter for readonly attribute.

**Returns** location of the mouse pointer

**Return type** `location.Location`

**configure\_backend** (*backend=None, category='control', reset=False*)

Custom implementation of the base method.

See base method for details.

**synchronize\_backend** (*backend=None, category='type', reset=False*)

Custom implementation of the base method.

See base method for details.

Select a backend for the instance, synchronizing configuration like screen size, key map, mouse pointer handling, etc. The object that carries this configuration is called screen.

**capture\_screen** (\*args)

Get the current screen as image.

**Parameters** **args** ([int] or `region.Region` or `None`) – region’s (x, y, width, height) or a region object or nothing to obtain an image of the full screen

**Returns** image of the current screen

**Return type** `image.Image`

**Raises** `NotImplementedError` if the base class method is called

**mouse\_move** (location, smooth=True)

Move the mouse to a desired location.

**Parameters**

- **location** (`location.Location`) – location on the screen to move to
- **smooth** (`bool`) – whether to sue smooth transition or just teleport the mouse

**Raises** `NotImplementedError` if the base class method is called

**mouse\_click** (button=None, count=1, modifiers=None)

Click the selected mouse button N times at the current mouse location.

**Parameters**

- **button** (`int` or `None`) – mouse button, e.g. `self.mouse_map.LEFT_BUTTON`
- **count** (`int`) – number of times to click
- **modifiers** (`[str]`) – special keys to hold during clicking (see `inputmap.KeyModifier` for extensive list)

**Raises** `NotImplementedError` if the base class method is called

**mouse\_down** (button)

Hold down a mouse button.

**Parameters** **button** (`int`) – button index depending on backend (see `inputmap.MouseButton` for extensive list)

**Raises** `NotImplementedError` if the base class method is called

**mouse\_up** (button)

Release a mouse button.

**Parameters** **button** (`int`) – button index depending on backend (see `inputmap.MouseButton` for extensive list)

**Raises** `NotImplementedError` if the base class method is called

**keys\_toggle** (keys, up\_down)

Hold down or release together all provided keys.

**Parameters**

- **keys** (`[str]` or `str` (no special keys in the second case)) – characters or special keys depending on the backend (see `inputmap.Key` for extensive list)
- **up\_down** (`bool`) – hold down if true else release

**Raises** `NotImplementedError` if the base class method is called

**keys\_press** (*keys*)

Press (hold down and release) together all provided keys.

**Parameters** **keys** (*[str]* or *str* (no special keys in the second case))  
– characters or special keys depending on the backend (see `inputmap.Key` for extensive list)

**keys\_type** (*text, modifiers*)

Type (press consecutively) all provided keys.

**Parameters**

- **text** (*[str]* or *str* (second case is preferred and first redundant)) – characters only (no special keys allowed)
- **modifiers** (*[str]*) – special keys to hold during typing (see `inputmap.KeyModifier` for extensive list)

**Raises** `NotImplementedError` if the base class method is called

**class** `guiobot.desktopcontrol.AutoPyDesktopControl` (*configure=True, synchronize=True*)

Bases: `guiobot.desktopcontrol.DesktopControl`

Desktop control backend implemented through AutoPy which is a small python library portable to Windows and Linux operating systems.

**\_\_init\_\_** (*configure=True, synchronize=True*)

Build a DC backend using AutoPy.

**configure\_backend** (*backend=None, category='autopy', reset=False*)

Custom implementation of the base method.

See base method for details.

**synchronize\_backend** (*backend=None, category='autopy', reset=False*)

Custom implementation of the base method.

See base method for details.

**capture\_screen** (*\*args*)

Custom implementation of the base method.

See base method for details.

**mouse\_move** (*location, smooth=True*)

Custom implementation of the base method.

See base method for details.

**mouse\_click** (*button=None, count=1, modifiers=None*)

Custom implementation of the base method.

See base method for details.

**mouse\_down** (*button*)

Custom implementation of the base method.

See base method for details.

**mouse\_up** (*button*)

Custom implementation of the base method.

See base method for details.

**keys\_toggle** (*keys, up\_down*)  
 Custom implementation of the base method.  
 See base method for details.

**keys\_type** (*text, modifiers*)  
 Custom implementation of the base method.  
 See base method for details.

**class** `guiobot.desktopcontrol.XDoToolDesktopControl` (*configure=True, synchronize=True*)  
 Bases: `guiobot.desktopcontrol.DesktopControl`

Desktop control backend implemented through the xdotool client and thus portable to Linux operating systems.

**\_\_init\_\_** (*configure=True, synchronize=True*)  
 Build a DC backend using XDoTool.

**configure\_backend** (*backend=None, category='xdotool', reset=False*)  
 Custom implementation of the base method.  
 See base method for details.

**synchronize\_backend** (*backend=None, category='xdotool', reset=False*)  
 Custom implementation of the base method.  
 See base method for details.

**capture\_screen** (*\*args*)  
 Custom implementation of the base method.  
 See base method for details.

**mouse\_move** (*location, smooth=True*)  
 Custom implementation of the base method.  
 See base method for details.

**mouse\_click** (*button=None, count=3, modifiers=None*)  
 Custom implementation of the base method.  
 See base method for details.

**mouse\_down** (*button*)  
 Custom implementation of the base method.  
 See base method for details.

**mouse\_up** (*button*)  
 Custom implementation of the base method.  
 See base method for details.

**keys\_toggle** (*keys, up\_down*)  
 Custom implementation of the base method.  
 See base method for details.

**keys\_type** (*text, modifiers*)  
 Custom implementation of the base method.  
 See base method for details.

**class** `guiobot.desktopcontrol.VNCDotoolDesktopControl` (*configure=True, synchronize=True*)

Bases: `guiobot.desktopcontrol.DesktopControl`

Desktop control backend implemented through the VNCDotool client and thus portable to any guest OS that is accessible through a VNC/RFB protocol.

**\_\_init\_\_** (*configure=True, synchronize=True*)

Build a DC backend using VNCDotool.

**configure\_backend** (*backend=None, category='vncdotool', reset=False*)

Custom implementation of the base method.

See base method for details.

**synchronize\_backend** (*backend=None, category='vncdotool', reset=False*)

Custom implementation of the base method.

See base method for details.

**capture\_screen** (*\*args*)

Custom implementation of the base method.

See base method for details.

**mouse\_move** (*location, smooth=True*)

Custom implementation of the base method.

See base method for details.

**mouse\_click** (*button=None, count=3, modifiers=None*)

Custom implementation of the base method.

See base method for details.

**mouse\_down** (*button*)

Custom implementation of the base method.

See base method for details.

**mouse\_up** (*button*)

Custom implementation of the base method.

See base method for details.

**keys\_toggle** (*keys, up\_down*)

Custom implementation of the base method.

See base method for details.

**keys\_type** (*text, modifiers*)

Custom implementation of the base method.

See base method for details.

**class** `guiobot.desktopcontrol.QemuDesktopControl` (*configure=True, synchronize=True*)

Bases: `guiobot.desktopcontrol.DesktopControl`

Desktop control backend implemented through the Qemu emulator and thus portable to any guest OS that runs on virtual machine.

---

**Note:** This backend can be used in accord with a qemu monitor object (python) provided by a library like virt-test.

---

**\_\_init\_\_** (*configure=True, synchronize=True*)  
Build a DC backend using Qemu.

**configure\_backend** (*backend=None, category='qemu', reset=False*)  
Custom implementation of the base method.  
See base method for details.

**synchronize\_backend** (*backend=None, category='qemu', reset=False*)  
Custom implementation of the base method.  
**Raises** `ValueError` if control backend is 'qemu' and no monitor is selected  
See base method for details.

**capture\_screen** (*\*args*)  
Custom implementation of the base method.  
See base method for details.

**mouse\_move** (*location, smooth=True*)  
Custom implementation of the base method.  
See base method for details.

**mouse\_click** (*button=None, count=3, modifiers=None*)  
Custom implementation of the base method.  
See base method for details.

**mouse\_down** (*button*)  
Custom implementation of the base method.  
See base method for details.

**mouse\_up** (*button*)  
Custom implementation of the base method.  
See base method for details.

**keys\_toggle** (*keys, up\_down*)  
Custom implementation of the base method.  
See base method for details.

**keys\_type** (*text, modifiers*)  
Custom implementation of the base method.  
See base method for details.

### 1.1.4 guiobot.errors module

**exception** `guiobot.errors.GuiBotError`  
Bases: `exceptions.Exception`  
GuiBot exception base class

**exception** `guiobot.errors.FileNotFoundError`  
Bases: `guiobot.errors.GuiBotError`  
Exception raised when a picture file cannot be found on disc

**exception** `guiobot.errors.IncompatibleTargetError`

Bases: `guiobot.errors.GuiBotError`

Exception raised when a matched target is of type that cannot be handled by the finder

**exception** `guiobot.errors.IncompatibleTargetFileError`

Bases: `guiobot.errors.GuiBotError`

Exception raised when a matched target is restored from a file of unsupported type

**exception** `guiobot.errors.FindError` (*failed\_target=None*)

Bases: `guiobot.errors.GuiBotError`

Exception raised when an Image cannot be found on the screen

`__init__` (*failed\_target=None*)

Build the exception possibly providing the failed target.

**Parameters** `failed_target` (`target.Target` or `None`) – the target that wasn't found

**exception** `guiobot.errors.NotFindError` (*failed\_target=None*)

Bases: `guiobot.errors.GuiBotError`

Exception raised when an Image can be found on the screen but should not be

`__init__` (*failed\_target=None*)

Build the exception possibly providing the failed target.

**Parameters** `failed_target` (`target.Target` or `None`) – the target that was found

**exception** `guiobot.errors.UnsupportedBackendError`

Bases: `guiobot.errors.GuiBotError`

Exception raised when a non-existent method is used for finding a target

**exception** `guiobot.errors.MissingHotmapError`

Bases: `guiobot.errors.GuiBotError`

Exception raised when an attempt to access a non-existent hotmap in the image logger is made

**exception** `guiobot.errors.UninitializedBackendError`

Bases: `guiobot.errors.GuiBotError`

Exception raised when a region is created within an empty screen (a disconnected desktop control backend)

### 1.1.5 guiobot.finder module

**class** `guiobot.finder.CVParameter` (*value, min\_val=None, max\_val=None, delta=10.0, tolerance=1.0, fixed=True, enumerated=False*)

Bases: `object`

A class for a single parameter used for CV backend configuration.

`__init__` (*value, min\_val=None, max\_val=None, delta=10.0, tolerance=1.0, fixed=True, enumerated=False*)

Build a computer vision parameter.

**Parameters**

- **value** (*bool or int or float or str or None*) – value of the parameter
- **min\_val** (*int or float or None*) – lower boundary for the parameter range
- **max\_val** (*int or float or None*) – upper boundary for the parameter range



- **delta** (*float*) – delta for the calibration and random value (no calibration if *delta* < *tolerance*)
- **tolerance** (*float*) – tolerance of calibration
- **fixed** (*bool*) – whether the parameter is prevented from calibration
- **enumerated** (*bool*) – whether the parameter value belongs to an enumeration or to a range (distance matters)

As a rule of thumb a good choice for the parameter delta is one fourth of the range since the delta will be used as standard deviation when generating a random value for the parameter from a normal distribution. The delta to tolerance ratio is basically the number of failing trials before the parameter converges and is usually set to ten.

**\_\_repr\_\_** ()

Provide a representation of the parameter for storing and reporting.

**Returns** special syntax representation of the parameter

**Return type** str

**static from\_string** (*raw*)

Parse a CV parameter from string.

**Parameters** **raw** (*str*) – string representation for the parameter

**Returns** parameter parsed from the representation

**Return type** *CVParameter*

**Raises** *ValueError* if unsupported type is encountered

**random\_value** (*mu=None, sigma=None*)

Return a random value of the CV parameter given its range and type.

**Parameters**

- **mu** (*bool or int or float or str or None*) – mean for a normal distribution, uniform distribution if None
- **sigma** (*bool or int or float or str or None*) – standard deviation for a normal distribution, quarter range if None (maximal range is equivalent to maximal data type values)

**Returns** a random value conforming to the CV parameter range and type

**Return type** bool or int or float or str or None

---

**Note:** Only uniform distribution is used for boolean values.

---

**class** `guiobot.finder.Finder` (*configure=True, synchronize=True*)

Bases: `guiobot.config.LocalConfig`

Base for all image matching functionality and backends.

The image finding methods include finding one or all matches above the similarity defined in the configuration of each backend.

There are many parameters that could contribute for a good match. They can all be manually adjusted or automatically calibrated.

**static from\_match\_file** (*filename*)

Read the configuration from a match file with the given filename.

**Parameters** `filename` (*str*) – match filename for the configuration

**Returns** target finder with the parsed (and generated) settings

**Return type** `finder.Finder`

**Raises** `IOError` if the respective match file couldn't be read

The influence of the read configuration is that of an overwrite, i.e. all parameters will be generated (if not already present) and then the ones read from the configuration file will be overwritten.

**static to\_match\_file** (*finder, filename*)

Write the configuration to a match file with the given filename.

**Parameters**

- **finder** (`finder.Finder`) – match configuration to save
- **filename** (*str*) – match filename for the configuration

**\_\_init\_\_** (*configure=True, synchronize=True*)

Build a finder and its CV backend settings.

**configure\_backend** (*backend=None, category='find', reset=False*)

Custom implementation of the base method.

See base method for details.

**synchronize\_backend** (*backend=None, category='find', reset=False*)

Custom implementation of the base method.

See base method for details.

**can\_calibrate** (*category, mark*)

Fix the parameters for a given category backend algorithm, i.e. disallow the calibrator to change them.

**Parameters**

- **mark** (*bool*) – whether to mark for calibration
- **category** (*str*) – backend category whose parameters are marked

**Raises** `UnsupportedBackendError` if *category* is not among the supported backend categories

**copy** ()

**find** (*needle, haystack*)

Find all needle targets in a haystack image.

**Parameters**

- **needle** (`target.Target` or [`target.Target`]) – image, text, pattern, or a list or chain of such to look for
- **haystack** (`target.Image`) – image to look in

**Returns** all found matches (one in most use cases)

**Return type** [`match.Match`]

**Raises** `NotImplementedError` if the base class method is called

**log** (*lvl*)

Log images with an arbitrary logging level.

**Parameters** **lvl** (*int*) – logging level for the message

**class** `guiobot.finder.AutoPyFinder` (*configure=True, synchronize=True*)

Bases: `guiobot.finder.Finder`

Simple matching backend provided by AutoPy.

**\_\_init\_\_** (*configure=True, synchronize=True*)

Build a CV backend using AutoPy.

**configure\_backend** (*backend=None, category='autopy', reset=False*)

Custom implementation of the base method.

See base method for details.

**find** (*needle, haystack*)

Custom implementation of the base method.

**Parameters** **needle** (Image) – target iamge to search for

See base method for details.

**Warning:** AutoPy has a bug when finding multiple matches so it will currently only return a single match.

**class** `guiobot.finder.ContourFinder` (*configure=True, synchronize=True*)

Bases: `guiobot.finder.Finder`

Contour matching backend provided by OpenCV.

Essentially, we will find all countours in a binary image, preprocessed with Gaussian blur and adaptive threshold and return the ones with area (size) similar to the searched image.

**\_\_init\_\_** (*configure=True, synchronize=True*)

Build a CV backend using OpenCV's contour matching.

**configure\_backend** (*backend=None, category='contour', reset=False*)

Custom implementation of the base method.

See base method for details.

**configure** (*threshold\_filter=None, reset=True*)

Custom implementation of the base method.

**Parameters** **threshold\_filter** (*str or None*) – name of a preselected backend

**find** (*needle, haystack*)

Custom implementation of the base method.

**Parameters** **needle** (Image) – target iamge to search for

See base method for details.

First extract all contours from a binary (boolean, threshold) version of the needle and haystack and then match the needle contours with one or more sets of contours in the haystack image. The number of needle matches depends on the set similarity and can be improved by requiring minimal area for the contours to be considered.

**log** (*lvl*)

Custom implementation of the base method.

See base method for details.

**class** `guiobot.finder.TemplateFinder` (*configure=True, synchronize=True*)

Bases: `guiobot.finder.Finder`

Template matching backend provided by OpenCV.

`__init__` (*configure=True, synchronize=True*)

Build a CV backend using OpenCV's template matching.

`configure_backend` (*backend=None, category='template', reset=False*)

Custom implementation of the base method.

See base method for details.

`find` (*needle, haystack*)

Custom implementation of the base method.

**Parameters** `needle` (Image) – target iamge to search for

**Raises** `UnsupportedBackendError` if the choice of template matches is not among the supported ones

See base method for details.

`log` (*lvl*)

Custom implementation of the base method.

See base method for details.

**class** `guiobot.finder.FeatureFinder` (*configure=True, synchronize=True*)

Bases: `guiobot.finder.Finder`

Feature matching backend provided by OpenCV.

---

**Note:** SURF and SIFT are proprietary algorithms and are not available by default in newer OpenCV versions (>3.0).

---

`__init__` (*configure=True, synchronize=True*)

Build a CV backend using OpenCV's feature matching.

`configure_backend` (*backend=None, category='feature', reset=False*)

Custom implementation of the base method.

**Some relevant parameters are:**

- **detect filter - works for certain detectors and** determines how many initial features are detected in an image (e.g. hessian threshold for SURF detector)
- **match filter - determines what part of all matches** returned by feature matcher remain good matches
- **project filter - determines what part of the good** matches are considered inliers
- ratio test - boolean for whether to perform a ratio test
- symmetry test - boolean for whether to perform a symmetry test

See base method for details.

`configure` (*feature\_detect=None, feature\_extract=None, feature\_match=None, reset=True*)

Custom implementation of the base method.

**Parameters**

- **feature\_detect** (*str or None*) – name of a preselected backend
- **feature\_extract** (*str or None*) – name of a preselected backend
- **feature\_match** (*str or None*) – name of a preselected backend

**synchronize\_backend** (*backend=None, category='feature', reset=False*)

Custom implementation of the base method.

See base method for details.

**synchronize** (*feature\_detect=None, feature\_extract=None, feature\_match=None, reset=True*)

Custom implementation of the base method.

**Parameters**

- **feature\_detect** (*str or None*) – name of a preselected backend
- **feature\_extract** (*str or None*) – name of a preselected backend
- **feature\_match** (*str or None*) – name of a preselected backend

**find** (*needle, haystack*)

Custom implementation of the base method.

**Parameters needle** (*Image*) – target iamge to search for

See base method for details.

**Warning:** Finding multiple matches is currently not supported and this will currently only return a single match.

Available methods are: a combination of feature detector, extractor, and matcher.

**log** (*lvl*)

Custom implementation of the base method.

See base method for details.

**class** `guiobot.finder.CascadeFinder` (*classifier\_datapath='.', configure=True, synchronize=True*)

Bases: `guiobot.finder.Finder`

Cascade matching backend provided by OpenCV.

This matcher uses Haar cascade for object detection. It is the most advanced method for object detection excluding convolutional neural networks. However, it requires the generation of a Haar cascade (if such is not already provided) of the needle to be found.

TODO: Currently no similarity requirement can be applied due to the cascade classifier API.

**\_\_init\_\_** (*classifier\_datapath='.', configure=True, synchronize=True*)

Build a CV backend using OpenCV's cascade matching options.

**configure\_backend** (*backend=None, category='cascade', reset=False*)

Custom implementation of the base method.

See base method for details.

**find** (*needle, haystack*)

Custom implementation of the base method.

**Parameters needle** (*Pattern*) – target pattern (cascade) to search for

See base method for details.

**class** `guiobot.finder.TextFinder` (*configure=True, synchronize=True*)

Bases: `guiobot.finder.ContourFinder`

Text matching backend provided by OpenCV.

This matcher will find a text (string) needle in the haystack, eventually relying on Tesseract or simpler kNN-based OCR, using extremal regions or contours before recognition, and returning a match if the string is among the recognized strings using string metric similar to Hamming distance.

Extremal Region Filter algorithm described in: Neumann L., Matas J.: Real-Time Scene Text Localization and Recognition, CVPR 2012

**\_\_init\_\_** (*configure=True, synchronize=True*)

Build a CV backend using OpenCV's text matching options.

**configure\_backend** (*backend=None, category='text', reset=False*)

Custom implementation of the base method.

See base method for details.

**configure** (*text\_detector=None, text\_recognizer=None, threshold\_filter=None, threshold\_filter2=None, threshold\_filter3=None, reset=True*)

Custom implementation of the base method.

#### Parameters

- **text\_detector** (*str or None*) – name of a preselected backend
- **text\_recognizer** (*str or None*) – name of a preselected backend
- **threshold\_filter** (*str or None*) – threshold filter for the text detection stage
- **threshold\_filter2** (*str or None*) – additional threshold filter for the OCR stage
- **threshold\_filter3** (*str or None*) – additional threshold filter for distance transformation

**synchronize\_backend** (*backend=None, category='text', reset=False*)

Custom implementation of the base method.

See base method for details.

**synchronize** (*text\_detector=None, text\_recognizer=None, threshold\_filter=None, threshold\_filter2=None, threshold\_filter3=None, reset=True*)

Custom implementation of the base method.

#### Parameters

- **text\_detector** (*str or None*) – name of a preselected backend
- **text\_recognizer** (*str or None*) – name of a preselected backend
- **threshold\_filter** (*str or None*) – threshold filter for the text detection stage
- **threshold\_filter2** (*str or None*) – additional threshold filter for the OCR stage
- **threshold\_filter3** (*str or None*) – additional threshold filter for distance transformation

**find** (*needle, haystack*)

Custom implementation of the base method.

**Parameters** **needle** (Text) – target text to search for

See base method for details.

**log** (*lvl*)

Custom implementation of the base method.

See base method for details.

**class** `guiobot.finder.TemplateFeatureFinder` (*configure=True, synchronize=True*)

Bases: `guiobot.finder.TemplateFinder`, `guiobot.finder.FeatureFinder`

Hybrid matcher using both OpenCV's template and feature matching.

Feature matching is robust at small regions not too abundant of features where template matching is too picky. Template matching is good at large feature abundant regions and can be used as a heuristic for the feature matching. The current matcher will perform template matching first and then feature matching on the survived template matches to select among them one more time.

A separate (usually lower) front similarity is used for the first stage template matching in order to remove a lot of noise that would otherwise be distracting for the second stage feature matching.

**\_\_init\_\_** (*configure=True, synchronize=True*)

Build a CV backend using OpenCV's template and feature matching.

**configure\_backend** (*backend=None, category='tempfeat', reset=False*)

Custom implementation of the base method.

See base method for details.

**configure** (*template\_match=None, feature\_detect=None, feature\_extract=None, feature\_match=None, reset=True*)

Custom implementation of the base methods.

See base methods for details.

**synchronize** (*feature\_detect=None, feature\_extract=None, feature\_match=None, reset=True*)

Custom implementation of the base method.

See base method for details.

**find** (*needle, haystack*)

Custom implementation of the base method.

See base method for details.

Use template matching to deal with feature dense regions and guide a final feature matching stage.

**log** (*lvl*)

Custom implementation of the base method.

See base method for details.

**class** `guiobot.finder.DeepFinder` (*classifier\_datapath='.', configure=True, synchronize=True*)

Bases: `guiobot.finder.Finder`

Deep learning matching backend provided by PyTorch.

The current implementation contains a basic convolutional neural network which can be trained to produce needle locations from a haystack image.

**\_\_init\_\_** (*classifier\_datapath='.', configure=True, synchronize=True*)

Build a CV backend using OpenCV's text matching options.

**configure\_backend** (*backend=None, category='deep', reset=False*)

Custom implementation of the base method.

See base method for details.

**synchronize\_backend** (*backend=None, category='deep', reset=False*)

Custom implementation of the base method.

See base method for details.

**find** (*needle, haystack*)

Custom implementation of the base method.

**Parameters** **needle** (*Pattern*) – target pattern (cascade) to search for

See base method for details.

**train** (*epochs, train\_samples, train\_targets, data\_filename=None*)

Train the convolutional neural network.

**Parameters**

- **epochs** (*int*) – number of training epochs (train on all samples for each)
- **train\_samples** (*str*) – filename for the samples dataset
- **train\_targets** (*str*) – filename for the targets dataset
- **data\_filename** – file name for storing the trained model (won't store if None)
- **data\_filename** – str or None

**test** (*train\_samples, train\_targets*)

Test the convolutional neural network.

**Parameters**

- **train\_samples** (*str*) – filename for the samples dataset
- **train\_targets** (*str*) – filename for the targets dataset

**log** (*lvl*)

Custom implementation of the base method.

See base method for details.

**class** `guibot.finder.CustomFinder` (*configure=True, synchronize=True*)

Bases: `guibot.finder.Finder`

Custom matching backend with in-house CV algorithms.

<b>Warning:</b> This matcher is currently not supported by our configuration.
---

---

**Todo:** “in-house-raw” performs regular knn matching, but “in-house-region” performs a special filtering and replacement of matches based on positional information (it does not have ratio and symmetry tests and assumes that the needle is transformed preserving the relative positions of each pair of matches, i.e. no rotation is allowed, but scaling for example is supported)

---

**\_\_init\_\_** (*configure=True, synchronize=True*)

Build a CV backend using custom matching.

**configure\_backend** (*backend=None, category='custom', reset=False*)

Custom implementation of the base method.

See base method for details.

**find** (*needle, haystack*)

Custom implementation of the base method.

See base method for details.



---

**Todo:** This custom feature matching backend needs more serious reworking before it even makes sense to get properly documented.

---

**detect\_features** (*needle, haystack*)

In-house feature detection algorithm.

**Parameters**

- **needle** (*target . Image*) – image to look for
- **haystack** (*target . Image*) – image to look in

**Warning:** This method is currently not fully implemented. The current MSER might not be used in the actual implementation.

**regionMatch** (*desc1, desc2, kp1, kp2, refinements=50, recalc\_interval=10, variants\_k=100, variants\_ratio=0.33*)

Use location information to better decide on matched features.

**Parameters**

- **desc1** – descriptors of the first image
- **desc2** – descriptors of the second image
- **kp1** – key points of the first image
- **kp2** – key points of the second image
- **refinements** (*int*) – number of points to relocate
- **recalc\_interval** (*int*) – recalculation on a number of refinements
- **variants\_k** (*int*) – kNN parameter for to limit the alternative variants of a badly positioned feature
- **variants\_ratio** (*float*) – internal ratio test for knnMatch autopstop (see below)

**Returns** obtained matches

The knn distance is now only a heuristic for the search of best matched set as is information on relative location with regard to the other matches.

---

**Todo:** handle a subset of matches (ignoring some matches if not all features are detected)

---



---

**Todo:** disable kernel mapping (multiple needle feature mapped to a single haystack feature)

---

**knnMatch** (*desc1, desc2, k=1, desc4kp=1, autopstop=0.0*)

Performs k-Nearest Neighbor matching.

**Parameters**

- **desc1** – descriptors of the first image
- **desc2** – descriptors of the second image
- **k** (*int*) – categorization up to k-th nearest neighbor

- **desc4kp** (*int*) – legacy parameter for the old SURF() feature detector where desc4kp = len(desc2) / len(kp2) or analogically len(desc1) / len(kp1) i.e. needle row 5 is a descriptor vector for needle keypoint 5
- **autostop** (*float*) – stop automatically if the ratio (dist to k)/(dist to k+1) is close to 0, i.e. the k+1-th neighbor is too far.

**Returns** obtained matches

**class** `guiobot.finder.HybridFinder` (*configure=True, synchronize=True*)

Bases: `guiobot.finder.Finder`

Match a target through a sequence of differently configured attempts.

This matcher can work with any other matcher in the background and with unique or repeating matchers for each step. If a step fails, the matcher tries the next available along the fallback chain or fails if the end of the chain is reached.

**\_\_init\_\_** (*configure=True, synchronize=True*)

Build a hybrid matcher.

**configure\_backend** (*backend=None, category='hybrid', reset=False*)

Custom implementation of the base method.

See base method for details.

**synchronize\_backend** (*backend=None, category='hybrid', reset=False*)

Custom implementation of the base method.

See base method for details.

**find** (*needle, haystack*)

Custom implementation of the base method.

See base method for details.

## 1.1.6 guiobot.guiobot module

**class** `guiobot.guiobot.GuiBot` (*dc=None, cv=None*)

Bases: `guiobot.region.Region`

The main guiobot object is the root (first and screen wide) region with some convenience functions added.

**See also:**

Real API is inherited from `region.Region`.

**\_\_init\_\_** (*dc=None, cv=None*)

Build a guiobot object.

### Parameters

- **dc** (`desktopcontrol.DesktopControl` or `None`) – DC backend used for any desktop control
- **cv** (`finder.Finder` or `None`) – CV backend used for any target finding

We will initialize with default region of full screen and default desktop control and computer vision backends if none are provided.

**add\_path** (*directory*)

Add a path to the list of currently accessible paths if it wasn't already added.

**Parameters** **directory** (*str*) – path to add

**remove\_path** (*directory*)

Remove a path from the list of currently accessible paths.

**Parameters** **directory** (*str*) – path to add

## 1.1.7 guiobot.guiobot\_proxy module

Frontend with serialization compatible API allowing the use of Pyro4 modified `guiobot.GuiBot` object (creating and running the same object remotely and manipulating it locally). All the methods delegate their calls to this object with some additional postprocessing to make the execution remote so for information about the API please refer to it and `region.Region`.

`guiobot.guiobot_proxy.serialize_custom_error` (*class\_obj*)

Serialization method for the `errors.UnsupportedBackendError` which was chosen just as a sample.

**Parameters** **class\_obj** (*classobj*) – class object for the serialized error class

**Returns** serialization dictionary with the class name, arguments, and attributes

**Return type** {str, str or getset\_descriptor or dictproxy}

`guiobot.guiobot_proxy.register_exception_serialization` ()

We put here any exceptions that are too complicated for the default serialization and define their serialization methods.

---

**Note:** This would not be needed if we were using the Pickle serializer but its security problems at the moment made us prefer the serpent serializer paying for it with some extra setup steps and functions below.

---

**class** `guiobot.guiobot_proxy.GuiBotProxy` (*dc=None, cv=None*)

Bases: `guiobot.guiobot.GuiBot`

The proxy guiobot object is just a wrapper around the actual guiobot object that takes care of returning easily serializable Pyro4 proxy objects instead of the real ones or their serialized copies.

It allows you to move the mouse, type text and do any other GuiBot action from code which is executed on another machine somewhere on the network.

**\_\_init\_\_** (*dc=None, cv=None*)

Build a guiobot object.

### Parameters

- **dc** (`desktopcontrol.DesktopControl` or `None`) – DC backend used for any desktop control
- **cv** (`finder.Finder` or `None`) – CV backend used for any target finding

We will initialize with default region of full screen and default desktop control and computer vision backends if none are provided.

**find** (*target, timeout=10*)

Find a target (image, text, etc.) on the screen.

### Parameters

- **target** (str or `target.Target`) – target to look for
- **timeout** (*int*) – timeout before giving up

**Returns** match obtained from finding the target within the region

**Return type** `match.Match`

**Raises** `errors.FindError` if no match is found

This method is the main entrance to all our target finding capabilities and is the milestone for all target expect methods.

**find\_all** (*target*, *timeout=10*, *allow\_zero=False*)

Find multiples of a target on the screen.

**Parameters**

- **target** (str or `target.Target`) – target to look for
- **timeout** (*int*) – timeout before giving up
- **allow\_zero** (*bool*) – whether to allow zero matches or raise error

**Returns** matches obtained from finding the target within the region

**Return type** [`match.Match`]

**Raises** `errors.FindError` if no matches are found and zero matches are not allowed

This method is similar the one above but allows for more than one match.

**sample** (*target*)

Sample the similarity between a target and the screen, i.e. an empirical probability that the target is on the screen.

**Parameters** **target** (str or `target.Target`) – target to look for

**Returns** similarity with best match on the screen

**Return type** float

---

**Note:** Not all matchers support a ‘similarity’ value. The ones that don’t will return zero similarity (similarly to the target logging case).

---

**exists** (*target*, *timeout=0*)

Check if a target exists on the screen using the matching success as a threshold for the existence.

**Parameters**

- **target** (str or `target.Target`) – target to look for
- **timeout** (*int*) – timeout before giving up

**Returns** match obtained from finding the target within the region or nothing if no match is found

**Return type** `match.Match` or `None`

**wait** (*target*, *timeout=30*)

Wait for a target to appear (be matched) with a given timeout as failing tolerance.

**Parameters**

- **target** (str or `target.Target`) – target to look for
- **timeout** (*int*) – timeout before giving up

**Returns** match obtained from finding the target within the region

**Return type** `match.Match`

**Raises** `errors.FindError` if no match is found

**wait\_vanish** (*target*, *timeout=30*)

Wait for a target to disappear (be unmatched, i.e. matched without success) with a given timeout as failing tolerance.

**Parameters**

- **target** (*str* or `target.Target`) – target to look for
- **timeout** (*int*) – timeout before giving up

**Returns** whether the target disappeared from the region

**Return type** `bool`

**Raises** `errors.NotFoundError` if match is still found

**hover** (*target\_or\_location*)

Hover the mouse over a target or location.

**Parameters** **target\_or\_location** (`match.Match` or `location.Location` or *str* or `target.Target`) – target or location to hover to

**Returns** match from finding the target or nothing if hovering over a known location

**Return type** `match.Match` or `None`

**click** (*target\_or\_location*, *modifiers=None*)

Click on a target or location using the left mouse button and optionally holding special keys.

**Parameters**

- **target\_or\_location** (`match.Match` or `location.Location` or *str* or `target.Target`) – target or location to click on
- **modifiers** (*[str]*) – special keys to hold during clicking (see `inputmap.KeyModifier` for extensive list)

**Returns** match from finding the target or nothing if clicking on a known location

**Return type** `match.Match` or `None`

The special keys refer to a list of key modifiers, e.g.:

```
self.click('my_target', [KeyModifier.MOD_CTRL, 'x']).
```

**right\_click** (*target\_or\_location*, *modifiers=None*)

Click on a target or location using the right mouse button and optionally holding special keys.

Arguments and return values are analogical to `Region.click()`.

**double\_click** (*target\_or\_location*, *modifiers=None*)

Double click on a target or location using the left mouse button and optionally holding special keys.

Arguments and return values are analogical to `Region.click()`.

**multi\_click** (*target\_or\_location*, *count=3*, *modifiers=None*)

Click N times on a target or location using the left mouse button and optionally holding special keys.

Arguments and return values are analogical to `Region.click()`.

**click\_expect** (*click\_image\_or\_location*, *expect\_image\_or\_location=None*, *modifiers=None*, *timeout=60*)

Click on an image or location and wait for another one to appear.

**Parameters**

- **click\_image\_or\_location** (*Image or Location*) – image or location to click on
- **expect\_image\_or\_location** (*Image or Location or None*) – image or location to wait for
- **modifiers** (*[Key] or None*) – key modifiers when clicking
- **timeout** (*int*) – time in seconds to wait for

**Returns** match obtained from finding the second target within the region

**Return type** `match.Match`

**click\_vanish** (*click\_image\_or\_location, expect\_image\_or\_location=None, modifiers=None, timeout=60*)

Click on an image or location and wait for another one to disappear.

**Parameters**

- **click\_image\_or\_location** (*Image or Location*) – image or location to click on
- **expect\_image\_or\_location** (*Image or Location or None*) – image or location to wait for
- **modifiers** (*[Key] or None*) – key modifiers when clicking
- **timeout** (*int*) – time in seconds to wait for

**Returns** whether the second target disappeared from the region

**Return type** `bool`

**click\_at\_index** (*anchor, index=0, find\_number=3, timeout=10*)

Find all instances of an anchor image and click on the one with the desired index given that they are horizontally then vertically sorted.

**Parameters**

- **anchor** (*str or target.Target*) – image to find all matches of
- **index** (*int*) – index of the match to click on (assuming  $\geq 1$  matches), sorted according to their (x,y) coordinates
- **find\_number** (*int*) – expected number of matches which is necessary for fast failure in case some elements are not visualized and/or proper matching result
- **timeout** (*int*) – timeout before which the number of matches should be found

**Returns** match from finding the target of the desired index

**Return type** `match.Match`

---

**Note:** This method is a good replacement of a number of coincident limitations regarding the Windows version of autopsy and Pyro and therefore the (Windows) virtual user:

- autopsy has an old BUG regarding capturing the screen at a region with boundaries, different than the entire screen -> subregioning which is the main way to deal with any kind of highly repeating and homogeneous interface, is totally unavailable here.
- Pyro4 cannot serialize generators, so this is an implementation of a “generator step” involving clicking on consecutive matches.

- The serialized virtual user now returns a list of proxified matches when calling `find_all`, but they are all essentially useless as they don't proxify their returned objects and cannot be sent back as arguments. The special proxy interface of the virtual user was implemented only to handle the most basic case - serialize the objects returned by the main shared class by proxifying them (turning them into remote objects as well, which already have a well-defined serialization method) and nothing more.

**mouse\_down** (*target\_or\_location*, *button=None*)

Hold down an arbitrary mouse button on a target or location.

**Parameters**

- **target\_or\_location** (`match.Match` or `location.Location` or `str` or `target.Target`) – target or location to toggle on
- **button** (`int` or `None`) – button index depending on backend (default is left button) (see `inputmap.MouseButton` for extensive list)

**Returns** `match` from finding the target or nothing if toggling on a known location

**Return type** `match.Match` or `None`

**mouse\_up** (*target\_or\_location*, *button=None*)

Release an arbitrary mouse button on a target or location.

**Parameters**

- **target\_or\_location** (`match.Match` or `location.Location` or `str` or `target.Target`) – target or location to toggle on
- **button** (`int` or `None`) – button index depending on backend (default is left button) (see `inputmap.MouseButton` for extensive list)

**Returns** `match` from finding the target or nothing if toggling on a known location

**Return type** `match.Match` or `None`

**drag\_drop** (*src\_target\_or\_location*, *dst\_target\_or\_location*, *modifiers=None*)

Drag from and drop at a target or location optionally holding special keys.

**Parameters**

- **src\_target\_or\_location** (`match.Match` or `location.Location` or `str` or `target.Target`) – target or location to drag from
- **dst\_target\_or\_location** (`match.Match` or `location.Location` or `str` or `target.Target`) – target or location to drop at
- **modifiers** (`[str]`) – special keys to hold during dragging and dropping (see `inputmap.KeyModifier` for extensive list)

**Returns** `match` from finding the target or nothing if dropping at a known location

**Return type** `match.Match` or `None`

**drag\_from** (*target\_or\_location*, *modifiers=None*)

Drag from a target or location optionally holding special keys.

Arguments and return values are analogical to `Region.drag_drop()` but with *target\_or\_location* as *src\_target\_or\_location*.

**drop\_at** (*target\_or\_location*, *modifiers=None*)

Drop at a target or location optionally holding special keys.

Arguments and return values are analogical to `Region.drag_drop()` but with `target_or_location` as `dst_target_or_location`.

**press\_keys** (*keys*)

Press a single key or a list of keys simultaneously.

**Parameters** *keys* (*[str]* or *str* (possibly special keys in both cases)) – characters or special keys depending on the backend (see `inputmap.Key` for extensive list)

**Returns** *self*

**Return type** `Region`

Thus, the line `self.press_keys([Key.ENTER])` is equivalent to the line `self.press_keys(Key.ENTER)`. Other examples are:

```
self.press_keys([Key.CTRL, 'X'])
self.press_keys(['a', 'b', 3])
```

**press\_at** (*target\_or\_location=None, keys=None*)

Press a single key or a list of keys simultaneously at a specified target or location.

This method is similar to `Region.press_keys()` but with an extra argument like `Region.click()`.

**type\_text** (*text, modifiers=None*)

Type a list of consecutive character keys (without special keys).

**Parameters**

- **text** (*[str]* or *str* (no special keys in both cases)) – characters or strings (independent of the backend)
- **modifiers** (*[str]*) – special keys to hold during typing (see `inputmap.KeyModifier` for extensive list)

**Returns** *self*

**Return type** `Region`

Thus, the line `self.type_text(['hello'])` is equivalent to the line `self.type_text('hello')`. Other examples are:

```
self.type_text('ab3') # compare with press_keys()
self.type_text(['Hello', ' ', 'user3614']) # in cases with appending
```

Special keys are only allowed as modifiers here - simply call `Region.press_keys()` multiple times for consecutively typing special keys.

**type\_at** (*target\_or\_location=None, text="", modifiers=None*)

Type a list of consecutive character keys (without special keys) at a specified target or location.

This method is similar to `Region.type_text()` but with an extra argument like `Region.click()`.

**fill\_at** (*anchor, text, dx, dy, del\_flag=True, esc\_flag=True, mark\_clicks=1*)

Fills a new text at a text box with variable content using an anchor image and a displacement from that image.

**Parameters**

- **anchor** (`Match` or `Location` or `Target` or `str`) – target of reference for the input field
- **text** (*str*) – text to fill in



- **dx** (*int*) – displacement from the anchor in the x direction
- **dy** (*int*) – displacement from the anchor in the y direction
- **del\_flag** (*bool*) – whether to delete the highlighted text
- **esc\_flag** (*bool*) – whether to escape any possible fill suggestions
- **mark\_clicks** (*int*) – 0, 1, 2, ... clicks to highlight previous text

**Returns** self

**Return type** Region

**Raises** `exceptions.ValueError` if *mark\_click* is not acceptable value

If the delete flag is set the previous content will be deleted or otherwise the new text will be added in the end of the current text. If the escape flag is set an escape will be pressed after typing in order to avoid any entry suggestions from a dropdown list that could cover important image matching areas.

Since different interfaces behave differently, one might need a single, double or triple click to mark the already present text that has to be replaced.

**select\_at** (*anchor, image\_or\_index, dx, dy, dw=0, dh=0, mark\_clicks=1*)

Select an option at a dropdown list using either an integer index or an option image if the order cannot be easily inferred.

**Parameters**

- **anchor** (Match or Location or Target or str) – target of reference for the input dropdown menu
- **image\_or\_index** (*str or int*) – item image or item index
- **dx** (*int*) – displacement from the anchor in the x direction
- **dy** (*int*) – displacement from the anchor in the y direction
- **dw** (*int*) – width to add to the displacement for an image search area
- **dh** (*int*) – height to add to the displacement for an image search area
- **mark\_clicks** (*int*) – 0, 1, 2, ... clicks to highlight previous text

**Returns** self

**Return type** Region

It uses an anchor image which is rather constant and a displacement to locate the dropdown location. It moves down to the option if index is used where index 0 represents the current selection.

To avoid the limitations of the index method, an image of the option can be provided and will be matched in the area with and under the dropdown list. This also handles cases where the option coincides with the previously selected option. For more details see the really cool note in the end of this method.

## 1.1.8 guiobot.guiobot\_simple module

Frontend with simple procedural API allowing the use of a module instead of the `guiobot.GuiBot` object (creating and running this same object internally). All the methods delegate their calls to this object so for information about the API please refer to it and `region.Region`.

`guiobot.guiobot_simple.initialize()`

`guiobot.guiobot_simple.check_initialized()`

`guiobot.guiobot_simple.add_path(directory)`

```
guiobot.guiobot_simple.remove_path (directory)
guiobot.guiobot_simple.find (target, timeout=10)
guiobot.guiobot_simple.find_all (target, timeout=10, allow_zero=False)
guiobot.guiobot_simple.sample (target)
guiobot.guiobot_simple.exists (target, timeout=0)
guiobot.guiobot_simple.wait (target, timeout=30)
guiobot.guiobot_simple.wait_vanish (target, timeout=30)
guiobot.guiobot_simple.get_mouse_location ()
guiobot.guiobot_simple.hover (target_or_location)
guiobot.guiobot_simple.click (target_or_location, modifiers=None)
guiobot.guiobot_simple.right_click (target_or_location, modifiers=None)
guiobot.guiobot_simple.double_click (target_or_location, modifiers=None)
guiobot.guiobot_simple.multi_click (target_or_location, count=3, modifiers=None)
guiobot.guiobot_simple.click_expect (click_image_or_location, expect_image_or_location=None,
                                     modifiers=None, timeout=60)
guiobot.guiobot_simple.click_vanish (click_image_or_location, expect_image_or_location=None,
                                     modifiers=None, timeout=60)
guiobot.guiobot_simple.click_at_index (anchor, index=0, find_number=3, timeout=10)
guiobot.guiobot_simple.mouse_down (target_or_location, button=None)
guiobot.guiobot_simple.mouse_up (target_or_location, button=None)
guiobot.guiobot_simple.drag_drop (src_target_or_location, dst_target_or_location, modifiers=None)
guiobot.guiobot_simple.drag_from (target_or_location, modifiers=None)
guiobot.guiobot_simple.drop_at (target_or_location, modifiers=None)
guiobot.guiobot_simple.press_keys (keys)
guiobot.guiobot_simple.press_at (target_or_location=None, keys=None)
guiobot.guiobot_simple.type_text (text, modifiers=None)
guiobot.guiobot_simple.type_at (target_or_location=None, text="", modifiers=None)
guiobot.guiobot_simple.fill_at (anchor, text, dx, dy, del_flag=True, esc_flag=True, mark_clicks=1)
guiobot.guiobot_simple.select_at (anchor, image_or_index, dx, dy, dw=0, dh=0, mark_clicks=1)
```

### 1.1.9 guiobot.imagelogger module

**class** `guiobot.imagelogger.ImageLogger`

Bases: `object`

Logger for the image matching process with the help of images.

It always contains the current match case: the needle and haystack images/targets being matched and the hotmap (an image with additional drawn information on it), the matched similarity and the matched coordinates.

Generally, each finder class takes care of its own image logging, performing drawing or similar operations on the spot and deciding which hotmaps (also their names and order) to dump.

**step = 1**

number of the current step

**accumulate\_logging = False**

switch to stop logging and later on log all accumulated dumps at once

**logging\_level = 40**

level for the image logging

**logging\_destination = './imglog'**

destination for the image logging in order to dump images (the executing code decides when to clean this directory)

**step\_width = 3**

number of digits for the counter of logged steps

**\_\_init\_\_()**

Build an imagellogger object.

**printable\_step**

Getter for readonly attribute.

**Returns** step number prepended with zeroes to obtain a fixed length enumeration

**Return type** str

**debug()**

Log images with a DEBUG logging level.

**info()**

Log images with an INFO logging level.

**warning()**

Log images with a WARNING logging level.

**error()**

Log images with an ERROR logging level.

**critical()**

Log images with a CRITICAL logging level.

**dump\_matched\_images()**

Write file with the current needle and haystack.

The current needle and haystack (matched images) are stored as *needle* and *haystack* attributes.

**dump\_hotmap(name, hotmap)**

Write a file the given hotmap.

**Parameters**

- **name** (*str*) – filename to use for the image
- **hotmap** (*PIL.Image* or *numpy.ndarray*) – image (with matching results) to write

**clear()**

Clear all accumulated logging including hotmaps, similarities, and locations.

### 1.1.10 guibot.inputmap module

**class** `guibot.inputmap.Key`

Bases: `object`

Helper to contain all key mappings for a custom desktop control backend.

`__init__()`

Build an instance containing an empty key map.

`to_string(key)`

Provide with a text representation of a desired key according to the custom BC backend.

**Parameters** `key` (*str*) – selected key name according to the custom backend

**Returns** text representation of the selected key

**Return type** `str`

**Raises** `ValueError` if `key` is not found in the current key map

**class** `guibot.inputmap.AutoPyKey`

Bases: `guibot.inputmap.Key`

Helper to contain all key mappings for the AutoPy DC backend.

`__init__()`

Build an instance containing the key map for the AutoPy backend.

**class** `guibot.inputmap.XDoToolKey`

Bases: `guibot.inputmap.Key`

Helper to contain all key mappings for the xdotool DC backend.

`__init__()`

Build an instance containing the key map for the xdotool backend.

**class** `guibot.inputmap.VNCDoToolKey`

Bases: `guibot.inputmap.Key`

Helper to contain all key mappings for the VNCDoTool DC backend.

`__init__()`

Build an instance containing the key map for the VNCDoTool backend.

**class** `guibot.inputmap.QemuKey`

Bases: `guibot.inputmap.Key`

Helper to contain all key mappings for the Qemu DC backend.

`__init__()`

Build an instance containing the key map for the Qemu backend.

**class** `guibot.inputmap.KeyModifier`

Bases: `object`

Helper to contain all modifier key mappings for a custom desktop control backend.

`__init__()`

Build an instance containing an empty modifier key map.

`to_string(key)`

Provide with a text representation of a desired modifier key according to the custom BC backend.

**Parameters** `key` (*str*) – selected modifier name according to the current backend

**Returns** text representation of the selected modifier

**Return type** str

**Raises** ValueError if *key* is not found in the current modifier map

**class** `guiobot.inputmap.AutoPyKeyModifier`

Bases: `guiobot.inputmap.KeyModifier`

Helper to contain all modifier key mappings for the AutoPy DC backend.

`__init__()`

Build an instance containing the modifier key map for the AutoPy backend.

**class** `guiobot.inputmap.XDoToolKeyModifier`

Bases: `guiobot.inputmap.KeyModifier`

Helper to contain all modifier key mappings for the xdotool DC backend.

`__init__()`

Build an instance containing the modifier key map for the xdotool backend.

**class** `guiobot.inputmap.VNCDoToolKeyModifier`

Bases: `guiobot.inputmap.KeyModifier`

Helper to contain all modifier key mappings for the VNCDoTool DC backend.

`__init__()`

Build an instance containing the modifier key map for the VNCDoTool backend.

**class** `guiobot.inputmap.QemuKeyModifier`

Bases: `guiobot.inputmap.KeyModifier`

Helper to contain all modifier key mappings for the Qemu DC backend.

`__init__()`

Build an instance containing the modifier key map for the Qemu backend.

**class** `guiobot.inputmap.MouseButton`

Bases: object

Helper to contain all mouse button mappings for a custom desktop control backend.

`__init__()`

Build an instance containing an empty mouse button map.

`to_string(key)`

Provide with a text representation of a desired mouse button according to the custom BC backend.

**Parameters** *key* (*str*) – selected mouse button according to the current backend

**Returns** text representation of the selected mouse button

**Return type** str

**Raises** ValueError if *key* is not found in the current mouse map

**class** `guiobot.inputmap.AutoPyMouseButton`

Bases: `guiobot.inputmap.MouseButton`

Helper to contain all mouse button mappings for the AutoPy DC backend.

`__init__()`

Build an instance containing the mouse button map for the AutoPy backend.

**class** `guiobot.inputmap.XDoToolMouseButton`

Bases: `guiobot.inputmap.MouseButton`

Helper to contain all mouse button mappings for the xdotool DC backend.

`__init__()`

Build an instance containing the mouse button map for the xdotool backend.

**class** `guiobot.inputmap.VNCDoToolMouseButton`

Bases: `guiobot.inputmap.MouseButton`

Helper to contain all mouse button mappings for the VNCDoTool DC backend.

`__init__()`

Build an instance containing the mouse button map for the VNCDoTool backend.

**class** `guiobot.inputmap.QemuMouseButton`

Bases: `guiobot.inputmap.MouseButton`

Helper to contain all mouse button mappings for the Qemu DC backend.

`__init__()`

Build an instance containing the mouse button map for the Qemu backend.

### 1.1.11 `guiobot.location` module

**class** `guiobot.location.Location` (*xpos=0, ypos=0*)

Bases: `object`

Simple location on a 2D surface, region, or screen.

`__init__` (*xpos=0, ypos=0*)

Build a location object.

#### Parameters

- **xpos** (*int*) – x coordinate of the location
- **ypos** (*int*) – y coordinate of the location

`__str__` ()

Provide a compact form for the location.

**x**

Getter for readonly attribute.

**Returns** x coordinate of the location

**Return type** `int`

**y**

Getter for readonly attribute.

**Returns** y coordinate of the location

**Return type** `int`

### 1.1.12 `guiobot.match` module

**class** `guiobot.match.Match` (*xpos, ypos, width, height, dx=0, dy=0, similarity=0.0, dc=None, cv=None*)

Bases: `guiobot.region.Region`

Wrapper around image which adds data necessary for manipulation of matches on a screen.

`__init__` (*xpos*, *ypos*, *width*, *height*, *dx=0*, *dy=0*, *similarity=0.0*, *dc=None*, *cv=None*)  
 Build a match object.

**Parameters**

- **xpos** (*int*) – x coordinate of the upleft vertex of the match region
- **ypos** (*int*) – y coordinate of the upleft vertex of the match region
- **width** (*int*) – x distance from upleft to downright vertex of the match region
- **height** (*int*) – y distance from upleft to downright vertex of the match region
- **dx** (*int*) – x offset from the center of the match region
- **dy** (*int*) – y offset from the center of the match region
- **similarity** (*float*) – attained similarity of the match region

`__str__` ()  
 Provide the target location of the match distinguishing it from any location.

**x**  
 Getter for readonly attribute.  
**Returns** x coordinate of the upleft vertex of the region  
**Return type** int

**y**  
 Getter for readonly attribute.  
**Returns** y coordinate of the upleft vertex of the region  
**Return type** int

**dx**  
 Getter for readonly attribute.  
**Returns** x offset from the center of the match region  
**Return type** int

**dy**  
 Getter for readonly attribute.  
**Returns** y offset from the center of the match region  
**Return type** int

**similarity**  
 Getter for readonly attribute.  
**Returns** similarity the match was obtained with  
**Return type** float

**target**  
 Getter for readonly attribute.  
**Returns** target location to click on if clicking on the match  
**Return type** `location.Location`

**calc\_click\_point** (*xpos*, *ypos*, *width*, *height*, *offset*)  
 Calculate target location to click on if clicking on the match.

#### Parameters

- **xpos** (*int*) – x coordinate of upleft vertex of the match region
- **ypos** (*int*) – y coordinate of upleft vertex of the match region
- **width** (*int*) – width of the match region
- **height** (*int*) – height of the match region
- **offset** (`location.Location`) – offset from the match region center for the final target

**Returns** target location to click on if clicking on the match

**Return type** `location.Location`

### 1.1.13 guibot.path module

**class** `guibot.path.Path`

Bases: `object`

Handler for currently used target paths or sources of targets with a desired name.

The methods of this class are shared among all of its instances.

**add\_path** (*directory*)

Add a path to the list of currently accessible paths if it wasn't already added.

**Parameters** **directory** (*str*) – path to add

**remove\_path** (*directory*)

Remove a path from the list of currently accessible paths.

**Parameters** **directory** (*str*) – path to add

**Returns** whether the removal succeeded

**Return type** `bool`

**clear** ()

Clear all currently accessible paths.

**search** (*filename*, *restriction=""*, *silent=False*)

Search for a filename in the currently accessible paths.

#### Parameters

- **filename** (*str*) – filename of the target to search for
- **restriction** (*str*) – simple string to restrict the number of paths
- **silent** (*bool*) – whether to return `None` instead of error out

**Returns** the full name of the found target file or `None` if silent and no file was found

**Return type** `str` or `None`

**Raises** `FileNotFoundError` if no such file was found and not silent



### 1.1.14 guiobot.region module

**class** `guiobot.region.Region` (*xpos=0, ypos=0, width=0, height=0, dc=None, cv=None*)

Bases: `object`

Region of the screen supporting vertex and nearby region selection, validation of expected images, and mouse and keyboard control.

**\_\_init\_\_** (*xpos=0, ypos=0, width=0, height=0, dc=None, cv=None*)

Build a region object from upleft to downright vertex coordinates.

#### Parameters

- **xpos** (*int*) – x coordinate of the upleft vertex of the region
- **ypos** (*int*) – y coordinate of the upleft vertex of the region
- **width** (*int*) – width of the region (xpos+width for downright vertex x)
- **height** (*int*) – height of the region (ypos+height for downright vertex y)
- **dc** (`desktopcontrol.DesktopControl` or `None`) – DC backend used for any desktop control
- **cv** (`finder.Finder` or `None`) – CV backend used for any target finding

**Raises** `UninitializedBackendError` if the region is empty

If any of the backends is not defined a new one will be initiated using the parameters defined in `config.GlobalConfig`. If *width* or *height* remains zero, it will be set to the maximum available within the screen space.

**x**

Getter for readonly attribute.

**Returns** x coordinate of the upleft vertex of the region

**Return type** `int`

**y**

Getter for readonly attribute.

**Returns** y coordinate of the upleft vertex of the region

**Return type** `int`

**width**

Getter for readonly attribute.

**Returns** width of the region (xpos+width for downright vertex x)

**Return type** `int`

**height**

Getter for readonly attribute.

**Returns** height of the region (ypos+height for downright vertex y)

**Return type** `int`

**center**

Getter for readonly attribute.

**Returns** center of the region

**Return type** `location.Location`

**top\_left**

Getter for readonly attribute.

**Returns** upleft vertex of the region

**Return type** `location.Location`

**top\_right**

Getter for readonly attribute.

**Returns** upright vertex of the region

**Return type** `location.Location`

**bottom\_left**

Getter for readonly attribute.

**Returns** downleft vertex of the region

**Return type** `location.Location`

**bottom\_right**

Getter for readonly attribute.

**Returns** downright vertex of the region

**Return type** `location.Location`

**is\_empty**

Getter for readonly attribute.

**Returns** whether the region is empty, i.e. has zero size

**Return type** `bool`

**last\_match**

Getter for readonly attribute.

**Returns** last match obtained from finding a target within the region

**Return type** `match.Match`

**mouse\_location**

Main region methods

**nearby** (*rrange=50*)

Obtain a region containing the previous one but enlarged by a number of pixels on each side.

**Parameters** *rrange* (*int*) – number of pixels to add

**Returns** new region enlarged by *rrange* on all sides

**Return type** *Region*

**above** (*rrange=0*)

Obtain a region containing the previous one but enlarged by a number of pixels on the upper side.

**Parameters** *rrange* (*int*) – number of pixels to add

**Returns** new region enlarged by *rrange* on upper side

**Return type** *Region*

**below** (*rrange=0*)

Obtain a region containing the previous one but enlarged by a number of pixels on the lower side.

**Parameters** *rrange* (*int*) – number of pixels to add

**Returns** new region enlarged by *rrange* on lower side

**Return type** *Region*

**left** (*rrange=0*)

Obtain a region containing the previous one but enlarged by a number of pixels on the left side.

**Parameters** *rrange* (*int*) – number of pixels to add

**Returns** new region enlarged by *rrange* on left side

**Return type** *Region*

**right** (*rrange=0*)

Obtain a region containing the previous one but enlarged by a number of pixels on the right side.

**Parameters** *rrange* (*int*) – number of pixels to add

**Returns** new region enlarged by *rrange* on right side

**Return type** *Region*

**find** (*target, timeout=10*)

Find a target (image, text, etc.) on the screen.

**Parameters**

- **target** (str or `target.Target`) – target to look for
- **timeout** (*int*) – timeout before giving up

**Returns** match obtained from finding the target within the region

**Return type** `match.Match`

**Raises** `errors.FindError` if no match is found

This method is the main entrance to all our target finding capabilities and is the milestone for all target expect methods.

**find\_all** (*target, timeout=10, allow\_zero=False*)

Find multiples of a target on the screen.

**Parameters**

- **target** (str or `target.Target`) – target to look for
- **timeout** (*int*) – timeout before giving up
- **allow\_zero** (*bool*) – whether to allow zero matches or raise error

**Returns** matches obtained from finding the target within the region

**Return type** [`match.Match`]

**Raises** `errors.FindError` if no matches are found and zero matches are not allowed

This method is similar the one above but allows for more than one match.

**sample** (*target*)

Sample the similarity between a target and the screen, i.e. an empirical probability that the target is on the screen.

**Parameters** *target* (str or `target.Target`) – target to look for

**Returns** similarity with best match on the screen

**Return type** float

---

**Note:** Not all matchers support a ‘similarity’ value. The ones that don’t will return zero similarity (similarly to the target logging case).

---

**exists** (*target*, *timeout=0*)

Check if a target exists on the screen using the matching success as a threshold for the existence.

**Parameters**

- **target** (str or `target.Target`) – target to look for
- **timeout** (*int*) – timeout before giving up

**Returns** match obtained from finding the target within the region or nothing if no match is found

**Return type** `match.Match` or `None`

**wait** (*target*, *timeout=30*)

Wait for a target to appear (be matched) with a given timeout as failing tolerance.

**Parameters**

- **target** (str or `target.Target`) – target to look for
- **timeout** (*int*) – timeout before giving up

**Returns** match obtained from finding the target within the region

**Return type** `match.Match`

**Raises** `errors.FindError` if no match is found

**wait\_vanish** (*target*, *timeout=30*)

Wait for a target to disappear (be unmatched, i.e. matched without success) with a given timeout as failing tolerance.

**Parameters**

- **target** (str or `target.Target`) – target to look for
- **timeout** (*int*) – timeout before giving up

**Returns** whether the target disappeared from the region

**Return type** `bool`

**Raises** `errors.NotFindError` if match is still found

**idle** (*timeout*)

Wait for a number of seconds and continue the nested call chain.

**Parameters** **timeout** (*int*) – timeout to wait for

**Returns** `self`

**Return type** `Region`

This method can be used as both a way to compactly wait for some time while not breaking the call chain. e.g.:

```
aregion.hover('abox').idle(1).click('aboxwithinthebox')
```

and as a way to conveniently perform timeout in between actions.

**hover** (*target\_or\_location*)

Hover the mouse over a target or location.

**Parameters** `target_or_location` (`match.Match` or `location.Location` or `str` or `target.Target`) – target or location to hover to

**Returns** match from finding the target or nothing if hovering over a known location

**Return type** `match.Match` or `None`

**click** (`target_or_location`, `modifiers=None`)

Click on a target or location using the left mouse button and optionally holding special keys.

**Parameters**

- **target\_or\_location** (`match.Match` or `location.Location` or `str` or `target.Target`) – target or location to click on
- **modifiers** (`[str]`) – special keys to hold during clicking (see `inputmap.KeyModifier` for extensive list)

**Returns** match from finding the target or nothing if clicking on a known location

**Return type** `match.Match` or `None`

The special keys refer to a list of key modifiers, e.g.:

```
self.click('my_target', [KeyModifier.MOD_CTRL, 'x']).
```

**right\_click** (`target_or_location`, `modifiers=None`)

Click on a target or location using the right mouse button and optionally holding special keys.

Arguments and return values are analogical to `Region.click()`.

**double\_click** (`target_or_location`, `modifiers=None`)

Double click on a target or location using the left mouse button and optionally holding special keys.

Arguments and return values are analogical to `Region.click()`.

**multi\_click** (`target_or_location`, `count=3`, `modifiers=None`)

Click N times on a target or location using the left mouse button and optionally holding special keys.

Arguments and return values are analogical to `Region.click()`.

**click\_expect** (`click_image_or_location`, `expect_image_or_location=None`, `modifiers=None`, `timeout=60`)

Click on an image or location and wait for another one to appear.

**Parameters**

- **click\_image\_or\_location** (`Image` or `Location`) – image or location to click on
- **expect\_image\_or\_location** (`Image` or `Location` or `None`) – image or location to wait for
- **modifiers** (`[Key]` or `None`) – key modifiers when clicking
- **timeout** (`int`) – time in seconds to wait for

**Returns** match obtained from finding the second target within the region

**Return type** `match.Match`

**click\_vanish** (`click_image_or_location`, `expect_image_or_location=None`, `modifiers=None`, `timeout=60`)

Click on an image or location and wait for another one to disappear.

**Parameters**

- **click\_image\_or\_location** (*Image or Location*) – image or location to click on
- **expect\_image\_or\_location** (*Image or Location or None*) – image or location to wait for
- **modifiers** (*[Key] or None*) – key modifiers when clicking
- **timeout** (*int*) – time in seconds to wait for

**Returns** whether the second target disappeared from the region

**Return type** `bool`

**click\_at\_index** (*anchor, index=0, find\_number=3, timeout=10*)

Find all instances of an anchor image and click on the one with the desired index given that they are horizontally then vertically sorted.

**Parameters**

- **anchor** (*str or target.Target*) – image to find all matches of
- **index** (*int*) – index of the match to click on (assuming  $\geq 1$  matches), sorted according to their (x,y) coordinates
- **find\_number** (*int*) – expected number of matches which is necessary for fast failure in case some elements are not visualized and/or proper matching result
- **timeout** (*int*) – timeout before which the number of matches should be found

**Returns** match from finding the target of the desired index

**Return type** `match.Match`

---

**Note:** This method is a good replacement of a number of coincident limitations regarding the Windows version of autopsy and Pyro and therefore the (Windows) virtual user:

- autopsy has an old BUG regarding capturing the screen at a region with boundaries, different than the entire screen -> subregioning which is the main way to deal with any kind of highly repeating and homogeneous interface, is totally unavailable here.
- Pyro4 cannot serialize generators, so this is an implementation of a “generator step” involving clicking on consecutive matches.
- The serialized virtual user now returns a list of proxified matches when calling `find_all`, but they are all essentially useless as they don’t proxify their returned objects and cannot be sent back as arguments. The special proxy interface of the virtual user was implemented only to handle the most basic case - serialize the objects returned by the main shared class by proxifying them (turning them into remote objects as well, which already have a well-defined serialization method) and nothing more.

---

**mouse\_down** (*target\_or\_location, button=None*)

Hold down an arbitrary mouse button on a target or location.

**Parameters**

- **target\_or\_location** (*match.Match or location.Location or str or target.Target*) – target or location to toggle on
- **button** (*int or None*) – button index depending on backend (default is left button) (see `inputmap.MouseButton` for extensive list)

**Returns** match from finding the target or nothing if toggling on a known location

**Return type** `match.Match` or `None`

**mouse\_up** (*target\_or\_location*, *button=None*)

Release an arbitrary mouse button on a target or location.

**Parameters**

- **target\_or\_location** (`match.Match` or `location.Location` or `str` or `target.Target`) – target or location to toggle on
- **button** (*int* or *None*) – button index depending on backend (default is left button) (see `inputmap.MouseButton` for extensive list)

**Returns** `match` from finding the target or nothing if toggling on a known location

**Return type** `match.Match` or `None`

**drag\_drop** (*src\_target\_or\_location*, *dst\_target\_or\_location*, *modifiers=None*)

Drag from and drop at a target or location optionally holding special keys.

**Parameters**

- **src\_target\_or\_location** (`match.Match` or `location.Location` or `str` or `target.Target`) – target or location to drag from
- **dst\_target\_or\_location** (`match.Match` or `location.Location` or `str` or `target.Target`) – target or location to drop at
- **modifiers** (*[str]*) – special keys to hold during dragging and dropping (see `inputmap.KeyModifier` for extensive list)

**Returns** `match` from finding the target or nothing if dropping at a known location

**Return type** `match.Match` or `None`

**drag\_from** (*target\_or\_location*, *modifiers=None*)

Drag from a target or location optionally holding special keys.

Arguments and return values are analogical to `Region.drag_drop()` but with *target\_or\_location* as *src\_target\_or\_location*.

**drop\_at** (*target\_or\_location*, *modifiers=None*)

Drop at a target or location optionally holding special keys.

Arguments and return values are analogical to `Region.drag_drop()` but with *target\_or\_location* as *dst\_target\_or\_location*.

**press\_keys** (*keys*)

Press a single key or a list of keys simultaneously.

**Parameters** **keys** (*[str]* or *str* (possibly special keys in both cases)) – characters or special keys depending on the backend (see `inputmap.Key` for extensive list)

**Returns** `self`

**Return type** `Region`

Thus, the line `self.press_keys([Key.ENTER])` is equivalent to the line `self.press_keys(Key.ENTER)`. Other examples are:

```
self.press_keys([Key.CTRL, 'X'])
self.press_keys(['a', 'b', 3])
```

**press\_at** (*keys, target\_or\_location*)

Press a single key or a list of keys simultaneously at a specified target or location.

This method is similar to `Region.press_keys()` but with an extra argument like `Region.click()`.

**type\_text** (*text, modifiers=None*)

Type a list of consecutive character keys (without special keys).

#### Parameters

- **text** (*[str] or str (no special keys in both cases)*) – characters or strings (independent of the backend)
- **modifiers** (*[str]*) – special keys to hold during typing (see `inputmap.KeyModifier` for extensive list)

**Returns** `self`

**Return type** `Region`

Thus, the line `self.type_text(['hello'])` is equivalent to the line `self.type_text('hello')`. Other examples are:

```
self.type_text('ab3') # compare with press_keys()
self.type_text(['Hello', ' ', 'user3614']) # in cases with appending
```

Special keys are only allowed as modifiers here - simply call `Region.press_keys()` multiple times for consecutively typing special keys.

**type\_at** (*text, target\_or\_location, modifiers=None*)

Type a list of consecutive character keys (without special keys) at a specified target or location.

This method is similar to `Region.type_text()` but with an extra argument like `Region.click()`.

**fill\_at** (*anchor, text, dx, dy, del\_flag=True, esc\_flag=True, mark\_clicks=1*)

Fills a new text at a text box with variable content using an anchor image and a displacement from that image.

#### Parameters

- **anchor** (*Match or Location or Target or str*) – target of reference for the input field
- **text** (*str*) – text to fill in
- **dx** (*int*) – displacement from the anchor in the x direction
- **dy** (*int*) – displacement from the anchor in the y direction
- **del\_flag** (*bool*) – whether to delete the highlighted text
- **esc\_flag** (*bool*) – whether to escape any possible fill suggestions
- **mark\_clicks** (*int*) – 0, 1, 2, ... clicks to highlight previous text

**Returns** `self`

**Return type** `Region`

**Raises** `exceptions.ValueError` if `mark_click` is not acceptable value

If the delete flag is set the previous content will be deleted or otherwise the new text will be added in the end of the current text. If the escape flag is set an escape will be pressed after typing in order to avoid any entry suggestions from a dropdown list that could cover important image matching areas.



Since different interfaces behave differently, one might need a single, double or triple click to mark the already present text that has to be replaced.

**select\_at** (*anchor, image\_or\_index, dx, dy, dw=0, dh=0, mark\_clicks=1*)

Select an option at a dropdown list using either an integer index or an option image if the order cannot be easily inferred.

**Parameters**

- **anchor** (*Match or Location or Target or str*) – target of reference for the input dropdown menu
- **image\_or\_index** (*str or int*) – item image or item index
- **dx** (*int*) – displacement from the anchor in the x direction
- **dy** (*int*) – displacement from the anchor in the y direction
- **dw** (*int*) – width to add to the displacement for an image search area
- **dh** (*int*) – height to add to the displacement for an image search area
- **mark\_clicks** (*int*) – 0, 1, 2, ... clicks to highlight previous text

**Returns** self

**Return type** *Region*

It uses an anchor image which is rather constant and a displacement to locate the dropdown location. It moves down to the option if index is used where index 0 represents the current selection.

To avoid the limitations of the index method, an image of the option can be provided and will be matched in the area with and under the dropdown list. This also handles cases where the option coincides with the previously selected option. For more details see the really cool note in the end of this method.

### 1.1.15 guiobot.target module

**class** `guiobot.target.Target` (*match\_settings=None*)

Bases: `object`

Target used to obtain screen location for clicking, typing, validation of expected visual output, etc.

**static from\_data\_file** (*filename*)

Read the target type from the extension of the target filename.

**Parameters** **filename** (*str*) – data filename for the target

**Returns** target of type determined from its data filename extension

**Return type** `target.Target`

**Raises** `errors.IncompatibleTargetFileError` if the data file if of unknown type

**static from\_match\_file** (*filename*)

Read the target type and configuration from a match file with the given filename.

**Parameters** **filename** (*str*) – match filename for the configuration

**Returns** target of type determined from its parsed (and generated) settings

**Return type** `target.Target`

**\_\_init\_\_** (*match\_settings=None*)

Build a target object.

**Parameters** `match_settings` (`finder.Finder` or `None`) – predefined configuration for the CV backend if any

`__str__()`

Provide a constant name 'target'.

**similarity**

Getter for readonly attribute.

**Returns** similarity required for the image to be matched

**Return type** `float`

**center\_offset**

Getter for readonly attribute.

**Returns** offset with respect to the target center (used for clicking)

**Return type** `location.Location`

This clicking location is set in the target in order to be customizable, it is then taken when matching to produce a clicking target for a match.

**load** (*filename*)

Load target from a file.

**Parameters** `filename` (*str*) – name for the target file

If no local file is found, we will perform search in the previously added paths.

**save** (*filename*)

Save target to a file.

**Parameters** `filename` (*str*) – name for the target file

**copy** ()

Perform a copy of the target data and match settings.

**Returns** copy of the current target (with settings)

**Return type** `target.Target`

**with\_center\_offset** (*xpos*, *ypos*)

Perform a copy of the target data with new match settings and with a newly defined center offset.

**Parameters**

- `xpos` (*int*) – new offset in the x direction
- `ypos` (*int*) – new offset in the y direction

**Returns** copy of the current target with new center offset

**Return type** `target.Target`

**with\_similarity** (*new\_similarity*)

Perform a copy of the target data with new match settings and with a newly defined required similarity.

**Parameters** `new_similarity` (*float*) – new required similarity

**Returns** copy of the current target with new similarity

**Return type** `target.Target`

**class** `guiobot.target.Image` (*image\_filename=None*, *pil\_image=None*, *match\_settings=None*, *use\_cache=True*)

Bases: `guiobot.target.Target`

Container for image data supporting caching, clicking target, file operations, and preprocessing.

**\_\_init\_\_** (*image\_filename=None, pil\_image=None, match\_settings=None, use\_cache=True*)  
Build an image object.

**Parameters**

- **image\_filename** (*str or None*) – name of the image file if any
- **pil\_image** (*PIL.Image or None*) – image data - use cache or recreate if none
- **match\_settings** (*finder.Finder or None*) – predefined configuration for the CV backend if any
- **use\_cache** (*bool*) – whether to cache image data for better performance

**\_\_str\_\_** ()  
Provide the image filename.

**filename**  
Getter for readonly attribute.  
**Returns** filename of the image  
**Return type** str

**width**  
Getter for readonly attribute.  
**Returns** width of the image  
**Return type** int

**height**  
Getter for readonly attribute.  
**Returns** height of the image  
**Return type** int

**pil\_image**  
Getter for readonly attribute.  
**Returns** image data of the image  
**Return type** PIL.Image

**load** (*filename, use\_cache=True*)  
Load image from a file.  
**Parameters**

- **filename** (*str*) – name for the target file
- **use\_cache** (*bool*) – whether to cache image data for better performance

**save** (*filename*)  
Save image to a file.  
**Parameters** **filename** (*str*) – name for the target file  
**Returns** copy of the current image with the new filename  
**Return type** target.Image

The image is compressed upon saving with a PNG compression setting specified by `config.GlobalConfig.image_quality()`.

**class** `guiobot.target.Text` (*value*, *match\_settings=None*)

Bases: `guiobot.target.Target`

Container for text data which is visually identified using OCR or general text detection methods.

**\_\_init\_\_** (*value*, *match\_settings=None*)

Build a text object.

**Parameters**

- **value** (*str*) – text value to search for
- **match\_settings** (`finder.Finder` or `None`) – predefined configuration for the CV backend if any

**\_\_str\_\_** ()

Provide a part of the text value.

**load** (*filename*)

Load text from a file.

**Parameters filename** (*str*) – name for the target file

**save** (*filename*)

Save text to a file.

**Parameters filename** (*str*) – name for the target file

**distance\_to** (*str2*)

Approximate Hungarian distance.

**Parameters str2** (*str*) – string to compare to

**Returns** string distance value

**Return type** float

**class** `guiobot.target.Pattern` (*data\_filename*, *match\_settings=None*)

Bases: `guiobot.target.Target`

Container for abstracted data which is obtained from training of a classifier in order to recognize a target.

**\_\_init\_\_** (*data\_filename*, *match\_settings=None*)

Build a pattern object.

**Parameters**

- **data\_filename** (*str*) – name of the text file if any
- **match\_settings** (`finder.Finder` or `None`) – predefined configuration for the CV backend if any

**\_\_str\_\_** ()

Provide the data filename.

**load** (*filename*)

Load pattern from a file.

**Parameters filename** (*str*) – name for the target file

**save** (*filename*)

Save pattern to a file.

**Parameters filename** (*str*) – name for the target file

**class** `guiobot.target.Chain` (*target\_name*, *match\_settings=None*)

Bases: `guiobot.target.Target`

Container for multiple configurations representing the same target.

The simplest version of a chain is a sequence of the same match configuration steps performed on a sequence of images until one of them succeeds. Every next step in this chain is a fallback case if the previous step did not succeed.

**\_\_init\_\_** (*target\_name*, *match\_settings=None*)

Build an chain object.

**Parameters**

- **target\_name** (*str*) – name of the target for all steps
- **match\_settings** (`finder.Finder` or `None`) – predefined configuration for the CV backend if any

**\_\_str\_\_** ()

Provide the target name.

**\_\_iter\_\_** ()

Provide an iterator over the steps.

**load** (*steps\_filename*)

Load steps from a sequence definition file.

**Parameters** **steps\_filename** (*str*) – names for the sequence definition file

**Raises** `errors.UnsupportedBackendError` if a chain step is of unknown type

**Raises** `IOError` if an chain step line cannot be parsed

**save** (*steps\_filename*)

Save steps to a sequence definition file.

**Parameters** **steps\_filename** (*str*) – names for the sequence definition file

## 1.2 Module contents



## g

- guiobot, 49
- guiobot.calibrator, 1
- guiobot.config, 3
- guiobot.desktopcontrol, 6
- guiobot.errors, 11
- guiobot.finder, 12
- guiobot.guiobot, 22
- guiobot.guiobot\_proxy, 23
- guiobot.guiobot\_simple, 29
- guiobot.imagellogger, 30
- guiobot.inputmap, 32
- guiobot.location, 34
- guiobot.match, 34
- guiobot.path, 36
- guiobot.region, 37
- guiobot.target, 45





## Symbols

- `__init__()` (guiobot.calibrator.Calibrator method), 1
  - `__init__()` (guiobot.config.LocalConfig method), 5
  - `__init__()` (guiobot.desktopcontrol.AutoPyDesktopControl method), 8
  - `__init__()` (guiobot.desktopcontrol.DesktopControl method), 6
  - `__init__()` (guiobot.desktopcontrol.QemuDesktopControl method), 10
  - `__init__()` (guiobot.desktopcontrol.VNCDoToolDesktopControl method), 10
  - `__init__()` (guiobot.desktopcontrol.XDoToolDesktopControl method), 9
  - `__init__()` (guiobot.errors.FindError method), 12
  - `__init__()` (guiobot.errors.NotFindError method), 12
  - `__init__()` (guiobot.finder.AutoPyFinder method), 15
  - `__init__()` (guiobot.finder.CVParameter method), 12
  - `__init__()` (guiobot.finder.CascadeFinder method), 17
  - `__init__()` (guiobot.finder.ContourFinder method), 15
  - `__init__()` (guiobot.finder.CustomFinder method), 20
  - `__init__()` (guiobot.finder.DeepFinder method), 19
  - `__init__()` (guiobot.finder.FeatureFinder method), 16
  - `__init__()` (guiobot.finder.Finder method), 14
  - `__init__()` (guiobot.finder.HybridFinder method), 22
  - `__init__()` (guiobot.finder.TemplateFeatureFinder method), 19
  - `__init__()` (guiobot.finder.TemplateFinder method), 16
  - `__init__()` (guiobot.finder.TextFinder method), 18
  - `__init__()` (guiobot.guiobot.GuiBot method), 22
  - `__init__()` (guiobot.guiobot\_proxy.GuiBotProxy method), 23
  - `__init__()` (guiobot.imagelogger.ImageLogger method), 31
  - `__init__()` (guiobot.inputmap.AutoPyKey method), 32
  - `__init__()` (guiobot.inputmap.AutoPyKeyModifier method), 33
  - `__init__()` (guiobot.inputmap.AutoPyMouseButton method), 33
  - `__init__()` (guiobot.inputmap.Key method), 32
  - `__init__()` (guiobot.inputmap.KeyModifier method), 32
  - `__init__()` (guiobot.inputmap.MouseButton method), 33
  - `__init__()` (guiobot.inputmap.QemuKey method), 32
  - `__init__()` (guiobot.inputmap.QemuKeyModifier method), 33
  - `__init__()` (guiobot.inputmap.QemuMouseButton method), 34
  - `__init__()` (guiobot.inputmap.VNCDoToolKey method), 32
  - `__init__()` (guiobot.inputmap.VNCDoToolKeyModifier method), 33
  - `__init__()` (guiobot.inputmap.VNCDoToolMouseButton method), 34
  - `__init__()` (guiobot.inputmap.XDoToolKey method), 32
  - `__init__()` (guiobot.inputmap.XDoToolKeyModifier method), 33
  - `__init__()` (guiobot.inputmap.XDoToolMouseButton method), 34
  - `__init__()` (guiobot.location.Location method), 34
  - `__init__()` (guiobot.match.Match method), 35
  - `__init__()` (guiobot.region.Region method), 37
  - `__init__()` (guiobot.target.Chain method), 49
  - `__init__()` (guiobot.target.Image method), 47
  - `__init__()` (guiobot.target.Pattern method), 48
  - `__init__()` (guiobot.target.Target method), 45
  - `__init__()` (guiobot.target.Text method), 48
  - `__iter__()` (guiobot.target.Chain method), 49
  - `__metaclass__` (guiobot.config.GlobalConfig attribute), 4
  - `__repr__()` (guiobot.finder.CVParameter method), 13
  - `__str__()` (guiobot.location.Location method), 34
  - `__str__()` (guiobot.match.Match method), 35
  - `__str__()` (guiobot.target.Chain method), 49
  - `__str__()` (guiobot.target.Image method), 47
  - `__str__()` (guiobot.target.Pattern method), 48
  - `__str__()` (guiobot.target.Target method), 46
  - `__str__()` (guiobot.target.Text method), 48
- ## A
- `above()` (guiobot.region.Region method), 38
  - `accumulate_logging` (guiobot.imagelogger.ImageLogger attribute), 31

- add\_path() (guiobot.guiobot.GuiBot method), 22
  - add\_path() (guiobot.path.Path method), 36
  - add\_path() (in module guiobot.guiobot\_simple), 29
  - AutoPyDesktopControl (class in guiobot.desktopcontrol), 8
  - AutoPyFinder (class in guiobot.finder), 14
  - AutoPyKey (class in guiobot.inputmap), 32
  - AutoPyKeyModifier (class in guiobot.inputmap), 33
  - AutoPyMouseButton (class in guiobot.inputmap), 33
- ## B
- below() (guiobot.region.Region method), 38
  - benchmark() (guiobot.calibrator.Calibrator method), 1
  - bottom\_left (guiobot.region.Region attribute), 38
  - bottom\_right (guiobot.region.Region attribute), 38
- ## C
- calc\_click\_point() (guiobot.match.Match method), 35
  - calibrate() (guiobot.calibrator.Calibrator method), 2
  - Calibrator (class in guiobot.calibrator), 1
  - can\_calibrate() (guiobot.finder.Finder method), 14
  - capture\_screen() (guiobot.desktopcontrol.AutoPyDesktopControl method), 8
  - capture\_screen() (guiobot.desktopcontrol.DesktopControl method), 7
  - capture\_screen() (guiobot.desktopcontrol.QemuDesktopControl method), 11
  - capture\_screen() (guiobot.desktopcontrol.VNCDoToolDesktopControl method), 10
  - capture\_screen() (guiobot.desktopcontrol.XDoToolDesktopControl method), 9
  - CascadeFinder (class in guiobot.finder), 17
  - center (guiobot.region.Region attribute), 37
  - center\_offset (guiobot.target.Target attribute), 46
  - Chain (class in guiobot.target), 48
  - check\_initialized() (in module guiobot.guiobot\_simple), 29
  - clear() (guiobot.imagelogger.ImageLogger method), 31
  - clear() (guiobot.path.Path method), 36
  - click() (guiobot.guiobot\_proxy.GuiBotProxy method), 25
  - click() (guiobot.region.Region method), 41
  - click() (in module guiobot.guiobot\_simple), 30
  - click\_at\_index() (guiobot.guiobot\_proxy.GuiBotProxy method), 26
  - click\_at\_index() (guiobot.region.Region method), 42
  - click\_at\_index() (in module guiobot.guiobot\_simple), 30
  - click\_delay (guiobot.config.GlobalConfig attribute), 3
  - click\_expect() (guiobot.guiobot\_proxy.GuiBotProxy method), 25
  - click\_expect() (guiobot.region.Region method), 41
  - click\_expect() (in module guiobot.guiobot\_simple), 30
  - click\_vanish() (guiobot.guiobot\_proxy.GuiBotProxy method), 26
  - click\_vanish() (guiobot.region.Region method), 41
  - click\_vanish() (in module guiobot.guiobot\_simple), 30
  - configure() (guiobot.config.LocalConfig method), 5
  - configure() (guiobot.finder.ContourFinder method), 15
  - configure() (guiobot.finder.FeatureFinder method), 16
  - configure() (guiobot.finder.TemplateFeatureFinder method), 19
  - configure() (guiobot.finder.TextFinder method), 18
  - configure\_backend() (guiobot.config.LocalConfig method), 5
  - configure\_backend() (guiobot.desktopcontrol.AutoPyDesktopControl method), 8
  - configure\_backend() (guiobot.desktopcontrol.DesktopControl method), 6
  - configure\_backend() (guiobot.desktopcontrol.QemuDesktopControl method), 11
  - configure\_backend() (guiobot.desktopcontrol.VNCDoToolDesktopControl method), 10
  - configure\_backend() (guiobot.desktopcontrol.XDoToolDesktopControl method), 9
  - configure\_backend() (guiobot.finder.AutoPyFinder method), 15
  - configure\_backend() (guiobot.finder.CascadeFinder method), 17
  - configure\_backend() (guiobot.finder.ContourFinder method), 15
  - configure\_backend() (guiobot.finder.CustomFinder method), 20
  - configure\_backend() (guiobot.finder.DeepFinder method), 19
  - configure\_backend() (guiobot.finder.FeatureFinder method), 16
  - configure\_backend() (guiobot.finder.Finder method), 14
  - configure\_backend() (guiobot.finder.HybridFinder method), 22
  - configure\_backend() (guiobot.finder.TemplateFeatureFinder method), 19
  - configure\_backend() (guiobot.finder.TemplateFinder method), 16
  - configure\_backend() (guiobot.finder.TextFinder method), 18
  - contour\_threshold\_backend (guiobot.config.GlobalConfig attribute), 4
  - ContourFinder (class in guiobot.finder), 15
  - copy() (guiobot.finder.Finder method), 14
  - copy() (guiobot.target.Target method), 46
  - critical() (guiobot.imagelogger.ImageLogger method), 31
  - CustomFinder (class in guiobot.finder), 20
  - CVPParameter (class in guiobot.finder), 12

## D

debug() (guiobot.imagelogger.ImageLogger method), 31  
 DeepFinder (class in guiobot.finder), 19  
 delay\_after\_drag (guiobot.config.GlobalConfig attribute), 3  
 delay\_before\_drop (guiobot.config.GlobalConfig attribute), 3  
 delay\_before\_keys (guiobot.config.GlobalConfig attribute), 4  
 delay\_between\_keys (guiobot.config.GlobalConfig attribute), 4  
 desktop\_control\_backend (guiobot.config.GlobalConfig attribute), 4  
 DesktopControl (class in guiobot.desktopcontrol), 6  
 detect\_features() (guiobot.finder.CustomFinder method), 21  
 distance\_to() (guiobot.target.Text method), 48  
 double\_click() (guiobot.guiobot\_proxy.GuiBotProxy method), 25  
 double\_click() (guiobot.region.Region method), 41  
 double\_click() (in module guiobot.guiobot\_simple), 30  
 drag\_drop() (guiobot.guiobot\_proxy.GuiBotProxy method), 27  
 drag\_drop() (guiobot.region.Region method), 43  
 drag\_drop() (in module guiobot.guiobot\_simple), 30  
 drag\_from() (guiobot.guiobot\_proxy.GuiBotProxy method), 27  
 drag\_from() (guiobot.region.Region method), 43  
 drag\_from() (in module guiobot.guiobot\_simple), 30  
 drop\_at() (guiobot.guiobot\_proxy.GuiBotProxy method), 27  
 drop\_at() (guiobot.region.Region method), 43  
 drop\_at() (in module guiobot.guiobot\_simple), 30  
 dump\_hotmap() (guiobot.imagelogger.ImageLogger method), 31  
 dump\_matched\_images() (guiobot.imagelogger.ImageLogger method), 31  
 dx (guiobot.match.Match attribute), 35  
 dy (guiobot.match.Match attribute), 35

## E

error() (guiobot.imagelogger.ImageLogger method), 31  
 exists() (guiobot.guiobot\_proxy.GuiBotProxy method), 24  
 exists() (guiobot.region.Region method), 40  
 exists() (in module guiobot.guiobot\_simple), 30

## F

feature\_detect\_backend (guiobot.config.GlobalConfig attribute), 4  
 feature\_extract\_backend (guiobot.config.GlobalConfig attribute), 4  
 feature\_match\_backend (guiobot.config.GlobalConfig attribute), 4

FeatureFinder (class in guiobot.finder), 16  
 filename (guiobot.target.Image attribute), 47  
 FileNotFoundError, 11  
 fill\_at() (guiobot.guiobot\_proxy.GuiBotProxy method), 28  
 fill\_at() (guiobot.region.Region method), 44  
 fill\_at() (in module guiobot.guiobot\_simple), 30  
 find() (guiobot.finder.AutoPyFinder method), 15  
 find() (guiobot.finder.CascadeFinder method), 17  
 find() (guiobot.finder.ContourFinder method), 15  
 find() (guiobot.finder.CustomFinder method), 20  
 find() (guiobot.finder.DeepFinder method), 19  
 find() (guiobot.finder.FeatureFinder method), 17  
 find() (guiobot.finder.Finder method), 14  
 find() (guiobot.finder.HybridFinder method), 22  
 find() (guiobot.finder.TemplateFeatureFinder method), 19  
 find() (guiobot.finder.TemplateFinder method), 16  
 find() (guiobot.finder.TextFinder method), 18  
 find() (guiobot.guiobot\_proxy.GuiBotProxy method), 23  
 find() (guiobot.region.Region method), 39  
 find() (in module guiobot.guiobot\_simple), 30  
 find\_all() (guiobot.guiobot\_proxy.GuiBotProxy method), 24  
 find\_all() (guiobot.region.Region method), 39  
 find\_all() (in module guiobot.guiobot\_simple), 30  
 find\_backend (guiobot.config.GlobalConfig attribute), 4  
 Finder (class in guiobot.finder), 13  
 FindError, 12  
 from\_data\_file() (guiobot.target.Target static method), 45  
 from\_match\_file() (guiobot.finder.Finder static method), 13  
 from\_match\_file() (guiobot.target.Target static method), 45  
 from\_string() (guiobot.finder.CVParameter static method), 13

## G

get\_mouse\_location() (in module guiobot.guiobot\_simple), 30  
 GlobalConfig (class in guiobot.config), 3  
 GuiBot (class in guiobot.guiobot), 22  
 guiobot (module), 49  
 guiobot.calibrator (module), 1  
 guiobot.config (module), 3  
 guiobot.desktopcontrol (module), 6  
 guiobot.errors (module), 11  
 guiobot.finder (module), 12  
 guiobot.guiobot (module), 22  
 guiobot.guiobot\_proxy (module), 23  
 guiobot.guiobot\_simple (module), 29  
 guiobot.imagelogger (module), 30  
 guiobot.inputmap (module), 32  
 guiobot.location (module), 34  
 guiobot.match (module), 34  
 guiobot.path (module), 36  
 guiobot.region (module), 37  
 guiobot.target (module), 45

GuiBotError, 11

GuiBotProxy (class in `guiobot.guiobot_proxy`), 23

## H

height (`guiobot.desktopcontrol.DesktopControl` attribute), 6

height (`guiobot.region.Region` attribute), 37

height (`guiobot.target.Image` attribute), 47

hover() (`guiobot.guiobot_proxy.GuiBotProxy` method), 25

hover() (`guiobot.region.Region` method), 40

hover() (in module `guiobot.guiobot_simple`), 30

hybrid\_match\_backend (`guiobot.config.GlobalConfig` attribute), 4

HybridFinder (class in `guiobot.finder`), 22

## I

idle() (`guiobot.region.Region` method), 40

Image (class in `guiobot.target`), 46

image\_logging\_destination (`guiobot.config.GlobalConfig` attribute), 4

image\_logging\_level (`guiobot.config.GlobalConfig` attribute), 4

image\_logging\_step\_width (`guiobot.config.GlobalConfig` attribute), 4

ImageLogger (class in `guiobot.imagelogger`), 30

IncompatibleTargetError, 11

IncompatibleTargetFileError, 12

info() (`guiobot.imagelogger.ImageLogger` method), 31

initialize() (in module `guiobot.guiobot_simple`), 29

is\_empty (`guiobot.region.Region` attribute), 38

## K

Key (class in `guiobot.inputmap`), 32

keymap (`guiobot.desktopcontrol.DesktopControl` attribute), 6

KeyModifier (class in `guiobot.inputmap`), 32

keys\_press() (`guiobot.desktopcontrol.DesktopControl` method), 7

keys\_toggle() (`guiobot.desktopcontrol.AutoPyDesktopControl` method), 8

keys\_toggle() (`guiobot.desktopcontrol.DesktopControl` method), 7

keys\_toggle() (`guiobot.desktopcontrol.QemuDesktopControl` method), 11

keys\_toggle() (`guiobot.desktopcontrol.VNCDoToolDesktopControl` method), 10

keys\_toggle() (`guiobot.desktopcontrol.XDoToolDesktopControl` method), 9

keys\_type() (`guiobot.desktopcontrol.AutoPyDesktopControl` method), 9

keys\_type() (`guiobot.desktopcontrol.DesktopControl` method), 8

keys\_type() (`guiobot.desktopcontrol.QemuDesktopControl` method), 11

keys\_type() (`guiobot.desktopcontrol.VNCDoToolDesktopControl` method), 10

keys\_type() (`guiobot.desktopcontrol.XDoToolDesktopControl` method), 9

knnMatch() (`guiobot.finder.CustomFinder` method), 21

## L

last\_match (`guiobot.region.Region` attribute), 38

left() (`guiobot.region.Region` method), 39

load() (`guiobot.target.Chain` method), 49

load() (`guiobot.target.Image` method), 47

load() (`guiobot.target.Pattern` method), 48

load() (`guiobot.target.Target` method), 46

load() (`guiobot.target.Text` method), 48

LocalConfig (class in `guiobot.config`), 5

Location (class in `guiobot.location`), 34

log() (`guiobot.finder.ContourFinder` method), 15

log() (`guiobot.finder.DeepFinder` method), 20

log() (`guiobot.finder.FeatureFinder` method), 17

log() (`guiobot.finder.Finder` method), 14

log() (`guiobot.finder.TemplateFeatureFinder` method), 19

log() (`guiobot.finder.TemplateFinder` method), 16

log() (`guiobot.finder.TextFinder` method), 18

logging\_destination (`guiobot.imagelogger.ImageLogger` attribute), 31

logging\_level (`guiobot.imagelogger.ImageLogger` attribute), 31

## M

Match (class in `guiobot.match`), 34

MissingHotmapError, 12

modmap (`guiobot.desktopcontrol.DesktopControl` attribute), 6

mouse\_click() (`guiobot.desktopcontrol.AutoPyDesktopControl` method), 8

mouse\_click() (`guiobot.desktopcontrol.DesktopControl` method), 7

mouse\_click() (`guiobot.desktopcontrol.QemuDesktopControl` method), 11

mouse\_click() (`guiobot.desktopcontrol.VNCDoToolDesktopControl` method), 10

mouse\_click() (`guiobot.desktopcontrol.XDoToolDesktopControl` method), 9

mouse\_down() (`guiobot.desktopcontrol.AutoPyDesktopControl` method), 8

mouse\_down() (`guiobot.desktopcontrol.DesktopControl` method), 7

mouse\_down() (`guiobot.desktopcontrol.QemuDesktopControl` method), 11

mouse\_down() (`guiobot.desktopcontrol.VNCDoToolDesktopControl` method), 10

mouse\_down() (`guiobot.desktopcontrol.XDoToolDesktopControl` method), 9

- [mouse\\_down\(\)](#) (guiobot.guiobot\_proxyGuiBotProxy method), 27  
[mouse\\_down\(\)](#) (guiobot.region.Region method), 42  
[mouse\\_down\(\)](#) (in module guiobot.guiobot\_simple), 30  
[mouse\\_location](#) (guiobot.desktopcontrol.DesktopControl attribute), 6  
[mouse\\_location](#) (guiobot.region.Region attribute), 38  
[mouse\\_move\(\)](#) (guiobot.desktopcontrol.AutoPyDesktopControl method), 8  
[mouse\\_move\(\)](#) (guiobot.desktopcontrol.DesktopControl method), 7  
[mouse\\_move\(\)](#) (guiobot.desktopcontrol.QemuDesktopControl method), 11  
[mouse\\_move\(\)](#) (guiobot.desktopcontrol.VNCDoToolDesktopControl method), 10  
[mouse\\_move\(\)](#) (guiobot.desktopcontrol.XDoToolDesktopControl method), 9  
[mouse\\_up\(\)](#) (guiobot.desktopcontrol.AutoPyDesktopControl method), 8  
[mouse\\_up\(\)](#) (guiobot.desktopcontrol.DesktopControl method), 7  
[mouse\\_up\(\)](#) (guiobot.desktopcontrol.QemuDesktopControl method), 11  
[mouse\\_up\(\)](#) (guiobot.desktopcontrol.VNCDoToolDesktopControl method), 10  
[mouse\\_up\(\)](#) (guiobot.desktopcontrol.XDoToolDesktopControl method), 9  
[mouse\\_up\(\)](#) (guiobot.guiobot\_proxyGuiBotProxy method), 27  
[mouse\\_up\(\)](#) (guiobot.region.Region method), 43  
[mouse\\_up\(\)](#) (in module guiobot.guiobot\_simple), 30  
[MouseButton](#) (class in guiobot.inputmap), 33  
[mousemap](#) (guiobot.desktopcontrol.DesktopControl attribute), 6  
[multi\\_click\(\)](#) (guiobot.guiobot\_proxyGuiBotProxy method), 25  
[multi\\_click\(\)](#) (guiobot.region.Region method), 41  
[multi\\_click\(\)](#) (in module guiobot.guiobot\_simple), 30
- N**
- [nearby\(\)](#) (guiobot.region.Region method), 38  
[NotFindError](#), 12
- P**
- [Path](#) (class in guiobot.path), 36  
[Pattern](#) (class in guiobot.target), 48  
[pil\\_image](#) (guiobot.target.Image attribute), 47  
[preprocess\\_special\\_chars](#) (guiobot.config.GlobalConfig attribute), 4  
[press\\_at\(\)](#) (guiobot.guiobot\_proxyGuiBotProxy method), 28  
[press\\_at\(\)](#) (guiobot.region.Region method), 43  
[press\\_at\(\)](#) (in module guiobot.guiobot\_simple), 30
- [press\\_keys\(\)](#) (guiobot.guiobot\_proxyGuiBotProxy method), 28  
[press\\_keys\(\)](#) (guiobot.region.Region method), 43  
[press\\_keys\(\)](#) (in module guiobot.guiobot\_simple), 30  
[printable\\_step](#) (guiobot.imagelogger.ImageLogger attribute), 31
- Q**
- [QemuDesktopControl](#) (class in guiobot.desktopcontrol), 10  
[QemuKey](#) (class in guiobot.inputmap), 32  
[QemuKeyModifier](#) (class in guiobot.inputmap), 33  
[QemuMouseButton](#) (class in guiobot.inputmap), 34
- R**
- [random\\_value\(\)](#) (guiobot.finder.CVParameter method), 13  
[Region](#) (class in guiobot.region), 37  
[regionMatch\(\)](#) (guiobot.finder.CustomFinder method), 21  
[register\\_exception\\_serialization\(\)](#) (in module guiobot.guiobot\_proxy), 23  
[remove\\_path\(\)](#) (guiobot.guiobotGuiBot method), 22  
[remove\\_path\(\)](#) (guiobot.path.Path method), 36  
[remove\\_path\(\)](#) (in module guiobot.guiobot\_simple), 30  
[rescan\\_speed\\_on\\_find](#) (guiobot.config.GlobalConfig attribute), 4  
[right\(\)](#) (guiobot.region.Region method), 39  
[right\\_click\(\)](#) (guiobot.guiobot\_proxyGuiBotProxy method), 25  
[right\\_click\(\)](#) (guiobot.region.Region method), 41  
[right\\_click\(\)](#) (in module guiobot.guiobot\_simple), 30  
[run\\_default\(\)](#) (guiobot.calibrator.Calibrator method), 3  
[run\\_peak\(\)](#) (guiobot.calibrator.Calibrator method), 3  
[run\\_performance\(\)](#) (guiobot.calibrator.Calibrator method), 3
- S**
- [sample\(\)](#) (guiobot.guiobot\_proxyGuiBotProxy method), 24  
[sample\(\)](#) (guiobot.region.Region method), 39  
[sample\(\)](#) (in module guiobot.guiobot\_simple), 30  
[save\(\)](#) (guiobot.target.Chain method), 49  
[save\(\)](#) (guiobot.target.Image method), 47  
[save\(\)](#) (guiobot.target.Pattern method), 48  
[save\(\)](#) (guiobot.target.Target method), 46  
[save\(\)](#) (guiobot.target.Text method), 48  
[save\\_needle\\_on\\_error](#) (guiobot.config.GlobalConfig attribute), 4  
[screen\\_autoconnect](#) (guiobot.config.GlobalConfig attribute), 4  
[search\(\)](#) (guiobot.calibrator.Calibrator method), 2  
[search\(\)](#) (guiobot.path.Path method), 36  
[select\\_at\(\)](#) (guiobot.guiobot\_proxyGuiBotProxy method), 29  
[select\\_at\(\)](#) (guiobot.region.Region method), 45  
[select\\_at\(\)](#) (in module guiobot.guiobot\_simple), 30

- serialize\_custom\_error() (in module guiobot.guiobot\_proxy), 23
  - similarity (guiobot.match.Match attribute), 35
  - similarity (guiobot.target.Target attribute), 46
  - smooth\_mouse\_drag (guiobot.config.GlobalConfig attribute), 4
  - step (guiobot.imagelogger.ImageLogger attribute), 31
  - step\_width (guiobot.imagelogger.ImageLogger attribute), 31
  - synchronize() (guiobot.config.LocalConfig method), 5
  - synchronize() (guiobot.finder.FeatureFinder method), 17
  - synchronize() (guiobot.finder.TemplateFeatureFinder method), 19
  - synchronize() (guiobot.finder.TextFinder method), 18
  - synchronize\_backend() (guiobot.config.LocalConfig method), 5
  - synchronize\_backend() (guiobot.desktopcontrol.AutoPyDesktopControl method), 8
  - synchronize\_backend() (guiobot.desktopcontrol.DesktopControl method), 6
  - synchronize\_backend() (guiobot.desktopcontrol.QemuDesktopControl method), 11
  - synchronize\_backend() (guiobot.desktopcontrol.VNCDoToolDesktopControl method), 10
  - synchronize\_backend() (guiobot.desktopcontrol.XDoToolDesktopControl method), 9
  - synchronize\_backend() (guiobot.finder.DeepFinder method), 19
  - synchronize\_backend() (guiobot.finder.FeatureFinder method), 16
  - synchronize\_backend() (guiobot.finder.Finder method), 14
  - synchronize\_backend() (guiobot.finder.HybridFinder method), 22
  - synchronize\_backend() (guiobot.finder.TextFinder method), 18
- T**
- Target (class in guiobot.target), 45
  - target (guiobot.match.Match attribute), 35
  - template\_match\_backend (guiobot.config.GlobalConfig attribute), 4
  - TemplateFeatureFinder (class in guiobot.finder), 18
  - TemplateFinder (class in guiobot.finder), 15
  - test() (guiobot.finder.DeepFinder method), 20
  - Text (class in guiobot.target), 47
  - text\_detect\_backend (guiobot.config.GlobalConfig attribute), 4
  - text\_ocr\_backend (guiobot.config.GlobalConfig attribute), 4
- TextFinder (class in guiobot.finder), 17
  - to\_match\_file() (guiobot.finder.Finder static method), 14
  - to\_string() (guiobot.inputmap.Key method), 32
  - to\_string() (guiobot.inputmap.KeyModifier method), 32
  - to\_string() (guiobot.inputmap.MouseButton method), 33
  - top\_left (guiobot.region.Region attribute), 37
  - top\_right (guiobot.region.Region attribute), 38
  - train() (guiobot.finder.DeepFinder method), 20
  - type\_at() (guiobot.guiobot\_proxy.GuiBotProxy method), 28
  - type\_at() (guiobot.region.Region method), 44
  - type\_at() (in module guiobot.guiobot\_simple), 30
  - type\_text() (guiobot.guiobot\_proxy.GuiBotProxy method), 28
  - type\_text() (guiobot.region.Region method), 44
  - type\_text() (in module guiobot.guiobot\_simple), 30
- U**
- UninitializedBackendError, 12
  - UnsupportedBackendError, 12
- V**
- VNCDoToolDesktopControl (class in guiobot.desktopcontrol), 9
  - VNCDoToolKey (class in guiobot.inputmap), 32
  - VNCDoToolKeyModifier (class in guiobot.inputmap), 33
  - VNCDoToolMouseButton (class in guiobot.inputmap), 34
- W**
- wait() (guiobot.guiobot\_proxy.GuiBotProxy method), 24
  - wait() (guiobot.region.Region method), 40
  - wait() (in module guiobot.guiobot\_simple), 30
  - wait\_vanish() (guiobot.guiobot\_proxy.GuiBotProxy method), 24
  - wait\_vanish() (guiobot.region.Region method), 40
  - wait\_vanish() (in module guiobot.guiobot\_simple), 30
  - warning() (guiobot.imagelogger.ImageLogger method), 31
  - width (guiobot.desktopcontrol.DesktopControl attribute), 6
  - width (guiobot.region.Region attribute), 37
  - width (guiobot.target.Image attribute), 47
  - with\_center\_offset() (guiobot.target.Target method), 46
  - with\_similarity() (guiobot.target.Target method), 46
- X**
- x (guiobot.location.Location attribute), 34
  - x (guiobot.match.Match attribute), 35
  - x (guiobot.region.Region attribute), 37
  - XDoToolDesktopControl (class in guiobot.desktopcontrol), 9
  - XDoToolKey (class in guiobot.inputmap), 32
  - XDoToolKeyModifier (class in guiobot.inputmap), 33
  - XDoToolMouseButton (class in guiobot.inputmap), 33
- Y**
- y (guiobot.location.Location attribute), 34

y (guiobot.match.Match attribute), 35  
y (guiobot.region.Region attribute), 37