

```

1  function varargout = gui(varargin)
2  % GUI MATLAB code for gui.fig
3  %     GUI, by itself, creates a new GUI or raises the existing
4  %     singleton*.
5  %
6  %     H = GUI returns the handle to a new GUI or the handle to
7  %     the existing singleton*.
8  %
9  %     GUI('CALLBACK',hObject,eventData,handles,...) calls the
    local
10 %     function named CALLBACK in GUI.M with the given input
    arguments.
11 %
12 %     GUI('Property','Value',...) creates a new GUI or raises
    the
13 %     existing singleton*. Starting from the left, property
    value pairs are
14 %     applied to the GUI before gui_OpeningFcn gets called. An
15 %     unrecognized property name or invalid value makes
    property application
16 %     stop. All inputs are passed to gui_OpeningFcn via
    varargin.
17 %
18 %     *See GUI Options on GUIDE's Tools menu. Choose "GUI
    allows only one
19 %     instance to run (singleton)".
20 %
21 % See also: GUIDE, GUIDATA, GUIHANDLES
22
23 % Edit the above text to modify the response to help gui
24
25 % Last Modified by GUIDE v2.5 12-Sep-2016 13:45:41
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name',       mfilename, ...
30                   'gui_Singleton',   gui_Singleton, ...
31                   'gui_OpeningFcn', @gui_OpeningFcn, ...
32                   'gui_OutputFcn',  @gui_OutputFcn, ...
33                   'gui_LayoutFcn',  [] , ...
34                   'gui_Callback',    []);
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if nargout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});

```

```

43 end
44 % End initialization code - DO NOT EDIT
45 end
46
47
48 % --- Executes just before gui is made visible.
49 function gui_OpeningFcn(hObject, ~, handles, varargin)
50 % This function has no output args, see OutputFcn.
51 % hObject    handle to figure
52 % eventdata  reserved - to be defined in a future version of
MATLAB
53 % handles     structure with handles and user data (see GUIDATA)
54 % varargin    command line arguments to gui (see VARARGIN)
55
56 % Choose default command line output for gui
57 handles.output = hObject;
58
59 % load java files for QR-Code decoding
60 javaaddpath('core-1.7.jar')
61 javaaddpath('javase-1.7.jar')
62
63 % Load parameter functions
64 handles.param = parameter();
65
66 % Set axes palette
67 colormap(handles.camview_infrared, 'jet');
68
69 % load images to axes
70 img = imread('QRCode.png');
71 image(img, 'parent', handles.camview_webcam);
72 handles.camview_webcam =
set_camview_default(handles.camview_webcam);
73 img = imread('laser.png');
74 image(img, 'parent', handles.camview_laser);
75 handles.camview_laser =
set_camview_default(handles.camview_laser);
76 img = imread('multispectral.jpg');
77 image(img, 'parent', handles.camview_multispectral);
78 handles.camview_multispectral =
set_camview_default(handles.camview_multispectral);
79 img = imread('infrared.jpg');
80 image(img, 'parent', handles.camview_infrared);
81 handles.camview_infrared =
set_camview_default(handles.camview_infrared);
82
83 % no laser images captured at start
84 handles.laser_images = false;
85
86 % disable demo mode at start

```

```

87     handles.demo = false;
88
89     % Update handles structure
90     guidata(hObject, handles);
91
92     % UIWAIT makes gui wait for user response (see UIRESUME)
93     % uiwait(handles.LegoDemo);
94 end
95
96 function camview = set_camview_default(camview)
97     camview.XTick = [];
98     camview.YTick = [];
99     camview.CLim = [0, 255];
100    camview.CLimMode = 'manual';
101    camview.DataAspectRatio = [1, 1, 1];
102 end
103
104 function enabled = is_serial_port(handles)
105     enabled = isfield(handles, 'serial') && isa(handles.serial,
        'class_serial_port');
106 end
107
108 function enabled = is_webcam(handles)
109     enabled = isfield(handles, 'webcam') && isa(handles.webcam,
        'class_videoinput');
110 end
111
112 function enabled = is_laser(handles)
113     enabled = isfield(handles, 'laser') && isa(handles.laser,
        'class_videoinput');
114 end
115
116 function enabled = is_infrared(handles)
117     enabled = isfield(handles, 'infrared') &&
        isa(handles.infrared, 'class_videoinput');
118 end
119
120 function enabled = is_multispectral(handles)
121     enabled = isfield(handles, 'multispectral') &&
        isa(handles.multispectral, 'class_gigecam');
122 end
123
124 % --- Executes on button press in camera_init.
125 function camera_init_Callback(hObject, ~, handles) %#ok<DEFNU>
126 % hObject    handle to camera_init (see GCBO)
127 % eventdata  reserved - to be defined in a future version of
MATLAB
128 % handles    structure with handles and user data (see GUIDATA)
129     handles.camera_init.Enable = 'off';

```

```
130
131 % Update handles structure
132 guidata(hObject, handles);
133 drawnow();
134
135 all_success = true;
136
137 if isfield(handles, 'gigelist') == false ||
138     istable(handles.gigelist) == false
139     try
140         handles.gigelist = gigeclist(); % speed up
141         gigeclist initialization
142     catch e
143         all_success = false;
144         warning('Exception in gigeclist(): %s',
145             getReport(e));
146     end
147 end
148
149 if ~is_webcam(handles)
150     try
151         handles.webcam = handles.param.camera_webcam();
152         handles = enable_webcam(handles, 'on');
153     catch e
154         all_success = false;
155         warning('Exception in camera_webcam(): %s',
156             getReport(e));
157     end
158 end
159
160 if ~is_laser(handles)
161     try
162         handles.laser = handles.param.camera_laser();
163         handles = enable_laser(handles, 'on');
164     catch e
165         all_success = false;
166         warning('Exception in camera_laser(): %s',
167             getReport(e));
168     end
169 end
170
171 if ~is_infrared(handles)
172     try
173         handles.infrared = handles.param.camera_infrared();
174         handles = enable_infrared(handles, 'on');
175     catch e
176         all_success = false;
177         warning('Exception in camera_infrared(): %s',
178             getReport(e));
```

```

173         end
174     end
175
176     if ~is_multispectral(handles)
177         try
178             handles.multispectral =
179                 handles.param.camera_multispectral(handles.gigelist);
180             handles = enable_multispectral(handles, 'on');
181         catch e
182             all_success = false;
183             warning('Exception in camera_multispectral(): %s',
184                 getReport(e));
185         end
186     end
187
188     % Set preview data to native camera bit depth (default is 8
189     % bit)
190     imaqmex('feature', '-previewFullBitDepth', true);
191
192     if all_success == false
193         handles.camera_init.Enable = 'on';
194     end
195
196     % Update handles structure
197     guidata(hObject, handles);
198 end
199
200 function handles = enable_webcam(handles, value)
201     if ~is_webcam(handles) || handles.demo
202         return;
203     end
204
205     handles.live_webcam.Enable = value;
206     handles.stop_webcam.Enable = value;
207     handles.snapshot_webcam.Enable = value;
208     handles.qr_button.Enable = value;
209
210     if is_serial_port(handles)
211         handles.demomode.Enable = value;
212     end
213 end
214
215 function handles = enable_laser(handles, value)
216     if ~is_laser(handles) || handles.demo
217         return;
218     end
219
220     handles.live_laser.Enable = value;
221     handles.stop_laser.Enable = value;

```

```

219     handles.capture_start.Enable = value;
220     handles.capture_stop.Enable = value;
221     handles.capture_calc.Enable = value;
222
223     if is_serial_port(handles)
224         handles.demomode.Enable = value;
225     end
226 end
227
228 function handles = enable_multispectral(handles, value)
229     if ~is_multispectral(handles) || handles.demo
230         return;
231     end
232
233     handles.live_multispectral.Enable = value;
234     handles.stop_multispectral.Enable = value;
235     handles.snapshot_multispectral.Enable = value;
236
237     if is_serial_port(handles)
238         handles.demomode.Enable = value;
239     end
240 end
241
242 function handles = enable_infrared(handles, value)
243     if ~is_infrared(handles) || handles.demo
244         return;
245     end
246
247     handles.live_infrared.Enable = value;
248     handles.stop_infrared.Enable = value;
249     handles.snapshot_infrared.Enable = value;
250
251     if is_serial_port(handles)
252         handles.demomode.Enable = value;
253     end
254 end
255
256 % enable control elements
257 function handles = enable_serial(handles, value)
258     if ~handles.demo
259         handles.train_dir_left.Enable = value;
260         handles.train_dir_right.Enable = value;
261         handles.train_speed.Enable = value;
262
263         handles.led0.Enable = value;
264         handles.led1.Enable = value;
265         handles.led2.Enable = value;
266         handles.led3.Enable = value;
267         handles.ledA.Enable = value;

```

```

268
269         handles.halo0.Enable = value;
270         handles.halo1.Enable = value;
271         handles.haloA.Enable = value;
272     end
273
274     handles.demomode.Enable = value;
275 end
276
277 % --- Executes on button press in connect_serial_port.
278 function connect_serial_port_Callback(hObject, ~, handles)
279 %#ok<DEFNU>
280 % hObject     handle to connect_serial_port (see GCBO)
281 % eventdata   reserved - to be defined in a future version of
MATLAB
282 % handles     structure with handles and user data (see GUIDATA)
283     handles.connect_serial_port.Enable = 'off';
284
285     % Update handles structure
286     guidata(hObject, handles);
287     drawnow();
288
289     % Initialize serial port
290     if is_serial_port(handles) == false
291         try
292             handles.serial =
293                 handles.param.serial_port(@serial_callback,
294                     hObject);
295         catch e
296             warning('Exception in camera_webcam(): %s',
297                 getReport(e));
298         end
299     end
300
301     % connect
302     success = false;
303     try
304         success = handles.serial.connect();
305     catch e
306         warning('Exception in serial.connect: %s', getReport(e));
307     end
308
309     if success == true
310         handles = enable_serial(handles, 'on');
311     else
312         % warn that connecting failed
313         waitfor(msgbox('Verbindung konnte nicht hergestellt
314             werden.', 'Fehler', 'warn'));
315         handles.connect_serial_port.Enable = 'on';

```

```

311         end
312
313         % Update handles structure
314         guidata(hObject, handles);
315     end
316
317     % --- Outputs from this function are returned to the command
318     % line.
319     function varargout = gui_OutputFcn(~, ~, handles)
320     % varargout    cell array for returning output args (see
321     % VARARGOUT);
322     % hObject      handle to figure
323     % eventdata    reserved - to be defined in a future version of
324     % MATLAB
325     % handles      structure with handles and user data (see GUIDATA)
326
327     % Get default command line output from handles structure
328     varargout{1} = handles.output;
329 end
330
331 function img = normalize_adjust(img)
332     img = double(img);
333     minv = min(img(:));
334     maxv = max(img(:));
335     diff = maxv - minv;
336     img = (img - minv) ./ diff;
337 end
338
339 function img = normalize_adjust_255(img)
340     img = normalize_adjust(img) .* 255;
341 end
342
343 function img = infrared_adjust(img)
344     img = normalize_adjust_255(img);
345     img = imrotate(img, 90);
346 end
347
348 function preview_normalize_adjust_255(~, event, himage)
349     himage.CData = normalize_adjust_255(event.Data);
350 end
351
352 function preview_gray(~, event, himage)
353     himage.CData = event.Data(:, :, 1);
354 end
355
356 function preview_infrared_adjust(~, event, himage)
357     img = (normalize_adjust(event.Data) .* 64);
358     img = imrotate(img, 90);
359     himage.CData = img;

```



```

357 end
358
359 % --- Executes on button press in live_webcam.
360 function live_webcam_Callback(hObject, ~, handles) %#ok<DEFNU>
361 % hObject      handle to live_webcam (see GCBO)
362 % eventdata    reserved - to be defined in a future version of
MATLAB
363 % handles      structure with handles and user data (see GUIDATA)
364     handles = enable_webcam(handles, 'off');
365     guidata(hObject, handles);
366     drawnow();
367
368     if ~handles.webcam.preview(false, handles.camview_webcam)
369         waitfor(msgbox('Livebild konnte nicht geladen werden.',
            'Fehler', 'warn'));
370     end
371
372     handles = enable_webcam(handles, 'on');
373     guidata(hObject, handles);
374 end
375
376 % --- Executes on button press in live_laser.
377 function live_laser_Callback(hObject, ~, handles) %#ok<DEFNU>
378 % hObject      handle to live_laser (see GCBO)
379 % eventdata    reserved - to be defined in a future version of
MATLAB
380 % handles      structure with handles and user data (see GUIDATA)
381     handles = enable_laser(handles, 'off');
382     guidata(hObject, handles);
383     drawnow();
384
385     colormap(handles.camview_laser, 'gray');
386     if ~handles.laser.preview(@preview_normalize_adjust_255,
handles.camview_laser)
387         waitfor(msgbox('Livebild konnte nicht geladen werden.',
            'Fehler', 'warn'));
388     end
389
390     handles = enable_laser(handles, 'on');
391     guidata(hObject, handles);
392 end
393
394 % --- Executes on button press in live_multispectral.
395 function live_multispectral_Callback(hObject, ~, handles)
%#ok<DEFNU>
396 % hObject      handle to live_multispectral (see GCBO)
397 % eventdata    reserved - to be defined in a future version of
MATLAB
398 % handles      structure with handles and user data (see GUIDATA)

```

```

399     handles = enable_multispectral(handles, 'off');
400     guidata(hObject, handles);
401     drawnow();
402
403     colormap(handles.camview_multispectral, 'hot');
404     if ~handles.multispectral.preview(@preview_gray,
handles.camview_multispectral)
405         waitfor(msgbox('Livebild konnte nicht geladen werden.',
        'Fehler', 'warn'));
406     end
407
408     handles = enable_multispectral(handles, 'on');
409     guidata(hObject, handles);
410 end
411
412 % --- Executes on button press in live_infrared.
413 function live_infrared_Callback(hObject, ~, handles)
414 % hObject    handle to live_infrared (see GCBO)
415 % eventdata  reserved - to be defined in a future version of
MATLAB
416 % handles    structure with handles and user data (see GUIDATA)
417     handles = enable_infrared(handles, 'off');
418     guidata(hObject, handles);
419     drawnow();
420
421     if ~handles.infrared.preview(@preview_infrared_adjust,
handles.camview_infrared, true)
422         waitfor(msgbox('Livebild konnte nicht geladen werden.',
        'Fehler', 'warn'));
423     end
424
425     handles = enable_infrared(handles, 'on');
426     guidata(hObject, handles);
427 end
428
429 % --- Executes on button press in stop_webcam.
430 function stop_webcam_Callback(hObject, ~, handles) %#ok<DEFNU>
431 % hObject    handle to stop_webcam (see GCBO)
432 % eventdata  reserved - to be defined in a future version of
MATLAB
433 % handles    structure with handles and user data (see GUIDATA)
434     handles = enable_webcam(handles, 'off');
435     guidata(hObject, handles);
436     drawnow();
437
438     if ~handles.webcam.stoppreview()
439         waitfor(msgbox('Livebild konnte gestoppt werden.',
        'Fehler', 'warn'));
440     end

```

```

441
442     handles = enable_webcam(handles, 'on');
443     guidata(hObject, handles);
444 end
445
446 % --- Executes on button press in stop_laser.
447 function stop_laser_Callback(hObject, ~, handles) %#ok<DEFNU>
448 % hObject     handle to stop_laser (see GCBO)
449 % eventdata   reserved - to be defined in a future version of
MATLAB
450 % handles     structure with handles and user data (see GUIDATA)
451     handles = enable_laser(handles, 'off');
452     guidata(hObject, handles);
453     drawnow();
454
455     if ~handles.laser.stoppreview()
456         waitfor(msgbox('Livebild konnte gestoppt werden.',
            'Fehler', 'warn'));
457     end
458
459     handles = enable_laser(handles, 'on');
460     guidata(hObject, handles);
461 end
462
463 % --- Executes on button press in stop_multispectral.
464 function stop_multispectral_Callback(hObject, ~, handles)
MATLAB
465 % hObject     handle to stop_multispectral (see GCBO)
466 % eventdata   reserved - to be defined in a future version of
MATLAB
467 % handles     structure with handles and user data (see GUIDATA)
468     handles = enable_multispectral(handles, 'off');
469     guidata(hObject, handles);
470     drawnow();
471
472     if ~handles.multispectral.stoppreview()
473         waitfor(msgbox('Livebild konnte gestoppt werden.',
            'Fehler', 'warn'));
474     end
475
476     handles = enable_multispectral(handles, 'on');
477     guidata(hObject, handles);
478 end
479
480 % --- Executes on button press in stop_infrared.
481 function stop_infrared_Callback(hObject, ~, handles)
482 % hObject     handle to stop_infrared (see GCBO)
483 % eventdata   reserved - to be defined in a future version of
MATLAB

```

```

484 % handles      structure with handles and user data (see GUIDATA)
485     handles = enable_infrared(handles, 'off');
486     guidata(hObject, handles);
487     drawnow();
488
489     if ~handles.infrared.stoppreview()
490         waitfor(msgbox('Livebild konnte gestoppt werden.',
491             'Fehler', 'warn'));
492     end
493
494     handles = enable_infrared(handles, 'on');
495     guidata(hObject, handles);
496 end
497
498 % --- Executes on button press in snapshot_webcam.
499 function snapshot_webcam_Callback(hObject, ~, handles)
500 % hObject      handle to snapshot_webcam (see GCBO)
501 % eventdata    reserved - to be defined in a future version of
502 % MATLAB
503 % handles      structure with handles and user data (see GUIDATA)
504     handles = enable_webcam(handles, 'off');
505     guidata(hObject, handles);
506     drawnow();
507
508     fprintf('Webcam start\n');
509     if ~handles.webcam.snapshot(false, handles.camview_webcam)
510         waitfor(msgbox('Einzelbild konnte nicht geladen
511             werden.', 'Fehler', 'warn'));
512     end
513     fprintf('Webcam end\n');
514
515     handles = enable_webcam(handles, 'on');
516     guidata(hObject, handles);
517 end
518
519 % --- Executes on button press in snapshot_multispectral.
520 function snapshot_multispectral_Callback(hObject, ~, handles)
521 % hObject      handle to snapshot_multispectral (see GCBO)
522 % eventdata    reserved - to be defined in a future version of
523 % MATLAB
524 % handles      structure with handles and user data (see GUIDATA)
525     handles = enable_multispectral(handles, 'off');
526     guidata(hObject, handles);
527     drawnow();
528
529     fprintf('Multispectral start\n');
530     colormap(handles.camview_multispectral, 'hot');
531     if ~handles.multispectral.snapshot(false,
532         handles.camview_multispectral)

```

```

528         waitfor(msgbox('Einzelbild konnte nicht geladen
                    werden.', 'Fehler', 'warn')));
529     end
530     fprintf('Multispectral end\n');
531
532     handles = enable_multispectral(handles, 'on');
533     guidata(hObject, handles);
534 end
535
536 % --- Executes on button press in snapshot_infrared.
537 function snapshot_infrared_Callback(hObject, ~, handles)
538 %#ok<DEFNU>
539 % hObject    handle to snapshot_infrared (see GCBO)
540 % eventdata  reserved - to be defined in a future version of
MATLAB
541 % handles    structure with handles and user data (see GUIDATA)
542     handles = enable_infrared(handles, 'off');
543     guidata(hObject, handles);
544     drawnow();
545
546     fprintf('Infrared start\n');
547     if ~handles.infrared.snapshot(@infrared_adjust,
handles.camview_infrared)
548         waitfor(msgbox('Einzelbild konnte nicht geladen
                    werden.', 'Fehler', 'warn')));
549     end
550     handles.camview_infrared =
set_camview_default(handles.camview_infrared);
551     fprintf('Infrared end\n');
552
553     handles = enable_infrared(handles, 'on');
554     guidata(hObject, handles);
555 end
556
557 % --- Executes during object creation, after setting all
properties.
558 function train_speed_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
559 % hObject    handle to train_speed (see GCBO)
560 % eventdata  reserved - to be defined in a future version of
MATLAB
561 % handles    empty - handles not created until after all
CreateFcns called
562
563 % Hint: slider controls usually have a light gray background.
564 if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
565     set(hObject,'BackgroundColor',[.9 .9 .9]);
566 end
567 end

```

```

567
568 % --- Executes on slider movement.
569 function train_speed_Callback(hObject, ~, handles)
570 % hObject      handle to train_speed (see GCBO)
571 % eventdata    reserved - to be defined in a future version of
MATLAB
572 % handles      structure with handles and user data (see GUIDATA)
573
574 % Hints: get(hObject,'Value') returns position of slider
575 %         get(hObject,'Min') and get(hObject,'Max') to determine
range of slider
576     handles = enable_serial(handles, 'off');
577     guidata(hObject, handles);
578     drawnow();
579
580     try
581         val = round(hObject.Value);
582         hObject.Value = val;
583
584         handles.train_speed_label.String = sprintf('%d', val);
585         handles.serial.setTrainSpeed(val,
handles.train_dir_left.Value == 0);
586     catch e
587         warning('Train exception: %s', getReport(e));
588         waitfor(msgbox('Interner Fehler während der SerialPort
Kommunikation.', 'Fehler', 'warn'));
589     end
590
591     % Update handles structure
592     handles = enable_serial(handles, 'on');
593     guidata(hObject, handles);
594 end
595
596 % --- Executes on button press in train_dir_XXX.
597 function train_dir_Callback(hObject, ~, handles) %#ok<DEFNU>
598 % hObject      handle to train_dir_left (see GCBO)
599 % eventdata    reserved - to be defined in a future version of
MATLAB
600 % handles      structure with handles and user data (see GUIDATA)
601     handles = enable_serial(handles, 'off');
602     guidata(hObject, handles);
603     drawnow();
604
605     try
606         speed = str2double(handles.train_speed_label.String);
607         handles.serial.setTrainSpeed(speed,
handles.train_dir_left.Value == 0);
608     catch e
609         warning('Train exception: %s', getReport(e));

```

```

610         waitfor(msgbox('Interner Fehler während der SerialPort
        Kommunikation.', 'Fehler', 'warn'));
611     end
612
613     % Update handles structure
614     handles = enable_serial(handles, 'on');
615     guidata(hObject, handles);
616 end
617
618 % --- Executes on button press in ledX.
619 function led_Callback(hObject, ~, handles) %#ok<DEFNU>
620 % hObject    handle to led0 (see GCBO)
621 % eventdata  reserved - to be defined in a future version of
        MATLAB
622 % handles    structure with handles and user data (see GUIDATA)
623     handles = enable_serial(handles, 'off');
624     guidata(hObject, handles);
625     drawnow();
626
627     try
628         switch(1)
629             case handles.led0.Value
630                 val = 0;
631             case handles.led1.Value
632                 val = 1;
633             case handles.led2.Value
634                 val = 2;
635             case handles.led3.Value
636                 val = 3;
637             case handles.ledA.Value
638                 val = 4;
639             otherwise
640                 error('Unknown LED Radio Button checked.');
```



```

656 % eventdata reserved - to be defined in a future version of
MATLAB
657 % handles structure with handles and user data (see GUIDATA)
658 handles = enable_serial(handles, 'off');
659 guidata(hObject, handles);
660 drawnow();
661
662 try
663     switch(1)
664         case handles.halo0.Value
665             val = 0;
666         case handles.halo1.Value
667             val = 1;
668         case handles.haloA.Value
669             val = 4;
670         otherwise
671             error('Unknown Halogen Radio Button checked.');
```

672 end

673 handles.serial.setHalogen(val);

674 catch e

675 warning('Halogen exception: %s', getReport(e));

676 waitfor(msgbox('Interner Fehler während der SerialPort
Kommunikation.', 'Fehler', 'warn'));

677 end

678

679 % Update handles structure

680 handles = enable_serial(handles, 'on');

681 guidata(hObject, handles);

682 end

683

684 % --- Executes on button press in qr_button.

685 function qr_button_Callback(hObject, ~, handles)

686 % hObject handle to qr_button (see GCBO)

687 % eventdata reserved - to be defined in a future version of
MATLAB

688 % handles structure with handles and user data (see GUIDATA)

689 handles = enable_webcam(handles, 'off');

690 guidata(hObject, handles);

691 drawnow();

692

693 try

694 img = getimage(handles.camview_webcam);

695

696 text = decode_qr(img);

697 if isempty(text)

698 color = [1, 0, 0];

699 text = 'Nichts erkannt ...';

700 else

701 color = [0, 1, 0];


```

702         end
703         handles.qr_text.String = text;
704         handles.qr_text.ForegroundColor = color;
705     catch e
706         warning('QR-Code exception: %s', getReport(e));
707         waitfor(msgbox('Interner Fehler während der  
QR-Code-Erkennung.', 'Fehler', 'warn'));
708     end
709
710     % Update handles structure
711     handles = enable_webcam(handles, 'on');
712     guidata(hObject, handles);
713 end
714
715 function handle = config_laser_start(handle, demomode)
716     triggerconfig(handle, 'hardware', 'DeviceSpecific',  
    'DeviceSpecific');
717
718     % remove all images in cache
719     if handle.FramesAvailable > 0
720         handle.FramesPerTrigger = handle.FramesAvailable;
721         getdata(handle);
722     end
723
724     if demomode
725         handle.FramesPerTrigger = 500;
726     else
727         handle.FramesPerTrigger = 500;
728     end
729
730     src = getselectedsource(handle);
731     src.TriggerMode = 'On';
732
733     start(handle);
734 end
735
736 % --- Executes on button press in capture_start.
737 function capture_start_Callback(hObject, ~, handles)
738 % hObject    handle to capture_start (see GCBO)
739 % eventdata  reserved - to be defined in a future version of
MATLAB
740 % handles    structure with handles and user data (see GUIDATA)
741     handles = enable_laser(handles, 'off');
742     guidata(hObject, handles);
743     drawnow();
744
745     if handles.laser.stoppreview() == false ||  
handles.laser.config(@config_laser_start, handles.demo}  
== false

```

```

746         waitfor(msgbox('Interner Fehler während der
Laserlinienbild-Aufnahmekonfiguration.', 'Fehler',
'warn'));

747
748         handles = enable_laser(handles, 'on');
749         guidata(hObject, handles);
750         return;
751     end
752
753     if ~handles.demo
754         handles.capture_stop.Enable = 'on';
755     end
756
757     % Update handles structure
758     guidata(hObject, handles);
759 end
760
761 function [handle, images] = get_images(handle)
762     stop(handle);
763     count = handle.FramesAvailable;
764
765     handle.FramesPerTrigger = count;
766     images = getdata(handle);
767
768     triggerconfig(handle, 'manual');
769
770     src = getselectedsource(handle);
771     src.TriggerMode = 'Off';
772     handle.FramesPerTrigger = 1;
773 end
774
775 % --- Executes on button press in capture_stop.
776 function capture_stop_Callback(hObject, ~, handles)
777 % hObject    handle to capture_stop (see GCBO)
778 % eventdata  reserved - to be defined in a future version of
MATLAB
779 % handles    structure with handles and user data (see GUIDATA)
780     handles = enable_laser(handles, 'off');
781     guidata(hObject, handles);
782     drawnow();
783
784     [success, images] = handles.laser.config(@get_images);
785     if success && ~islogical(images)
786         n = size(images, 4);
787         handles.laser_images = images;
788
789         colormap(handles.camview_laser, 'gray');
790         handles.image_slider.Enable = 'on';
791         handles.image_slider.Min = 1;

```

```

792     handles.image_slider.Max = n;
793     handles.image_slider.SliderStep = [1 / (n - 1), 1 / (n
- 1)];
794     for i = 1:n
795         if handles.demo && mod(i, 5) ~= 0
796             continue;
797         end
798         capture_img = handles.laser_images(:, :, 1, i);
799         image(capture_img, 'parent', handles.camview_laser);
800         handles.camview_laser =
            set_camview_default(handles.camview_laser);
801         handles.image_slider.Value = i;
802         handles.img_count.String = sprintf('%d von %d', i,
n);
803         guidata(hObject, handles);
804         drawnow();
805         pause(0.001);
806     end
807
808     handles.cut_begin.Enable = 'on';
809     handles.cut_end.Enable = 'on';
810 else
811     n = 0;
812     handles.laser_images = false;
813     handles.image_slider.Enable = 'off';
814     handles.cut_begin.Enable = 'off';
815     handles.cut_end.Enable = 'off';
816 end
817
818 handles.img_count.String = sprintf('Bilder: %d', n);
819
820 handles = enable_laser(handles, 'on');
821 guidata(hObject, handles);
822 end
823
824 % --- Executes on button press in capture_calc.
825 function capture_calc_Callback(hObject, ~, handles)
826 % hObject    handle to capture_calc (see GCBO)
827 % eventdata  reserved - to be defined in a future version of
MATLAB
828 % handles    structure with handles and user data (see GUIDATA)
829     handles = enable_laser(handles, 'off');
830     guidata(hObject, handles);
831     drawnow();
832
833     if ~islogical(handles.laser_images)
834         n = size(handles.laser_images, 4);
835     else
836         n = 0;

```

```

837     end
838
839     if n < 10
840         if ~handles.demo
841             waitfor(msgbox('Zu wenige Bilder für 3D-Bild
842                             Berechnung.', 'Fehler', 'warn'));
843         end
844
845         handles = enable_laser(handles, 'on');
846         guidata(hObject, handles);
847         return;
848     end
849
850     try
851         Threshold = 100;                % Gray value threshold for
852         background extraction
853         % pre calibration data - use the program
854         laserschnittverfahren3 to calibrate your system
855         ps = 0.054381;                % Pixel relative size
856         (mm/pixel)
857         alpha = 13.844982;            % Triangulation angle in (°)
858         LinCoef = 0;                  % Linear coefficient 105
859         AngCoef = 0;                  % Angular coefficient
860         data3d = get3D(handles.laser_images, Threshold, ps,
861                         alpha, LinCoef, AngCoef);
862
863         handles.img_count.String = sprintf('Bilder: %d', n);
864
865         colormap(handles.camview_laser, 'jet');
866         mesh(data3d, 'parent', handles.camview_laser);
867         handles.camview_laser.YTick = [];
868         handles.camview_laser.XTick = [];
869         handles.camview_laser.ZTick = [];
870         min3d = min(min(data3d));
871         handles.camview_laser.DataAspectRatio = [15, 4.5, 1.5];
872         handles.camview_laser.CLim = [min3d, 0];
873         handles.camview_laser.CLimMode = 'manual';
874         rotate3d(handles.camview_laser, 'on');
875         guidata(hObject, handles);
876         drawnow();
877     catch e
878         warning('3D calc exception: %s', getReport(e));
879         waitfor(msgbox('Interner Fehler während der 3D-Bild
880                             Berechnung.', 'Fehler', 'warn'));
881     end
882
883     handles = enable_laser(handles, 'on');
884     guidata(hObject, handles);
885 end

```

```

880
881 % --- Executes on slider movement.
882 function image_slider_Callback(hObject, ~, handles) %#ok<DEFNU>
883 % hObject    handle to image_slider (see GCBO)
884 % eventdata  reserved - to be defined in a future version of
MATLAB
885 % handles    structure with handles and user data (see GUIDATA)
886
887 % Hints: get(hObject,'Value') returns position of slider
888 %         get(hObject,'Min') and get(hObject,'Max') to determine
range of slider
889     try
890         val = round(hObject.Value);
891         hObject.Value = val;
892
893         if ~islogical(handles.laser_images)
894             handles.img_count.String = sprintf('%d von %d',
val, size(handles.laser_images, 4));
895
896             colormap(handles.camview_laser, 'gray');
897             image(handles.laser_images(:, :, 1, val), 'parent',
handles.camview_laser);
898             handles.camview_laser =
set_camview_default(handles.camview_laser);
899         end
900     catch e
901         warning('Laser silder exception: %s', getReport(e));
902     end
903
904     guidata(hObject, handles);
905 end
906
907 % --- Executes during object creation, after setting all
properties.
908 function image_slider_CreateFcn(hObject, ~, ~) %#ok<DEFNU>
909 % hObject    handle to image_slider (see GCBO)
910 % eventdata  reserved - to be defined in a future version of
MATLAB
911 % handles    empty - handles not created until after all
CreateFcns called
912
913 % Hint: slider controls usually have a light gray background.
914 if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
915     set(hObject,'BackgroundColor',[.9 .9 .9]);
916 end
917 end
918
919

```

```

920 % --- Executes on button press in cut_begin.
921 function cut_begin_Callback(hObject, ~, handles) %#ok<DEFNU>
922 % hObject     handle to cut_begin (see GCBO)
923 % eventdata   reserved - to be defined in a future version of
MATLAB
924 % handles     structure with handles and user data (see GUIDATA)
925     n = size(handles.laser_images, 4);
926     first = handles.image_slider.Value;
927     handles.laser_images = handles.laser_images(:, :, 1,
first:n);
928     n = size(handles.laser_images, 4);
929
930     handles.image_slider.Min = 1;
931     handles.image_slider.Max = n;
932     if n > 1
933         step = 1 / (n - 1);
934     else
935         step = 1;
936     end
937     handles.image_slider.Value = 1;
938     handles.image_slider.SliderStep = [step, step];
939     handles.image_slider.Value = 1;
940     handles.img_count.String = sprintf('%d von %d',
handles.image_slider.Value, n);
941
942     guidata(hObject, handles);
943 end
944
945 % --- Executes on button press in cut_end.
946 function cut_end_Callback(hObject, ~, handles) %#ok<DEFNU>
947 % hObject     handle to cut_end (see GCBO)
948 % eventdata   reserved - to be defined in a future version of
MATLAB
949 % handles     structure with handles and user data (see GUIDATA)
950     last = handles.image_slider.Value;
951     handles.laser_images = handles.laser_images(:, :, 1, 1:last);
952     n = size(handles.laser_images, 4);
953
954     handles.image_slider.Min = 1;
955     handles.image_slider.Max = n;
956     if n > 1
957         step = 1 / (n - 1);
958     else
959         step = 1;
960     end
961     handles.image_slider.Value = 1;
962     handles.image_slider.SliderStep = [step, step];
963     handles.image_slider.Value = n;
964     handles.img_count.String = sprintf('%d von %d',

```

```

        handles.image_slider.Value, n);
965
966     guidata(hObject, handles);
967 end
968
969 % --- Executes on button press in demomode.
970 function demomode_Callback(hObject, ~, handles) %#ok<DEFNU>
971 % hObject     handle to demomode (see GCBO)
972 % eventdata   reserved - to be defined in a future version of
MATLAB
973 % handles     structure with handles and user data (see GUIDATA)
974 % Set train speed
975 if handles.demomode.Value == 0
976     handles.train_speed.Value = 0;
977 else
978     handles.train_speed.Value = 9;
979     handles.train_dir_left.Value = 1;
980 end
981 train_speed_Callback(handles.train_speed, [], handles);
982 handles = guidata(hObject);
983
984 handles = enable_serial(handles, 'off');
985 guidata(hObject, handles);
986 drawnow();
987
988 try
989     if handles.demomode.Value == 0
990         handles.serial.setDemoMode(0);
991
992         stop_infrared_Callback(handles.snapshot_infrared,
[], handles);
993         handles = guidata(hObject);
994
995         handles.demo = false;
996
997         handles = enable_webcam(handles, 'on');
998         handles = enable_laser(handles, 'on');
999         handles = enable_infrared(handles, 'on');
1000        handles = enable_multispectral(handles, 'on');
1001    else
1002        handles.serial.setDemoMode(1);
1003
1004        live_infrared_Callback(handles.snapshot_infrared,
[], handles);
1005        handles = guidata(hObject);
1006
1007        handles = enable_webcam(handles, 'off');
1008        handles = enable_laser(handles, 'off');
1009        handles = enable_infrared(handles, 'off');

```

```

1010         handles = enable_multispectral(handles, 'off');
1011
1012         handles.demo = true;
1013     end
1014     catch e
1015         warning('DemoMode exception: %s', getReport(e));
1016         waitfor(msgbox('Interner Fehler während der SerialPort
1017                     Kommunikation.', 'Fehler', 'warn'));
1018     end
1019     % Update handles structure
1020     handles = enable_serial(handles, 'on');
1021     guidata(hObject, handles);
1022 end
1023
1024 % --- Handles messages from COM-Port.
1025 function serial_callback(type, parameter, hObject)
1026 % hObject    handle to figure
1027 % type       type of the message as string
1028 % parameter  depends on the type
1029 %           if type is 'bat': train battery charging state
1030     handles = guidata(hObject);
1031
1032     switch(type)
1033     case 'prelap1'
1034         if handles.demo
1035             if is_laser(handles)
1036
1037                 capture_start_Callback(handles.capture_start,
1038                                         [], handles);
1039                 handles = guidata(hObject);
1040
1041                 pause(12);
1042
1043                 capture_stop_Callback(handles.capture_stop,
1044                                         [], handles);
1045                 handles = guidata(hObject);
1046
1047                 capture_calc_Callback(handles.capture_calc,
1048                                         [], handles);
1049                 handles = guidata(hObject);
1050             end
1051         end
1052     case 'lap1'
1053     case 'prelap0'
1054     case 'lap0'
1055         if handles.demo
1056             % Camera often crashes Matlab
1057             if is_multispectral(handles) &&

```



```
handles.haloA.Value ~= 1
```

```
1054
        snapshot_multispectral_Callback(handles.snaps
        hot_multispectral, [], handles);
        handles = guidata(hObject);
1055
1056     end
1057
1058     if is_webcam(handles) && handles.ledA.Value ~= 1
1059
        snapshot_webcam_Callback(handles.snapshot_web
        cam, [], handles);
        handles = guidata(hObject);
1060
1061
1062         qr_button_Callback(handles.qr_button, [],
        handles);
        handles = guidata(hObject);
1063
1064     end
1065 end
1066 case 'halo'
1067     if handles.demo
1068         % Camera often crashes Matlab
1069         if is_multispectral(handles)
1070             pause(0.2);
1071
        snapshot_multispectral_Callback(handles.snaps
        hot_multispectral, [], handles);
        handles = guidata(hObject);
1072
1073     end
1074 end
1075 case 'led'
1076     if handles.demo
1077         if is_webcam(handles)
1078
        snapshot_webcam_Callback(handles.snapshot_web
        cam, [], handles);
        handles = guidata(hObject);
1079
1080
1081         qr_button_Callback(handles.qr_button, [],
        handles);
        handles = guidata(hObject);
1082
1083     end
1084 end
1085 case 'bat'
1086     handles.battery_label.String = sprintf('Akku: %s',
        parameter);
1087 otherwise
1088     error('Unknown SerialPort Callback "%s".', type);
1089 end
1090
```

```

1091     % Update handles structure
1092     guidata(hObject, handles);
1093 end
1094
1095 % --- Executes when user attempts to close LegoDemo.
1096 function LegoDemo_CloseRequestFcn(hObject, ~, handles)
1097 %#ok<DEFNU>
1098 % hObject    handle to LegoDemo (see GCBO)
1099 % eventdata  reserved - to be defined in a future version of
MATLAB
1100 % handles    structure with handles and user data (see GUIDATA)
1101 % Hint: delete(hObject) closes the figure
1102     if is_serial_port(handles)
1103         delete(handles.serial);
1104     end
1105
1106     if is_webcam(handles)
1107         delete(handles.webcam);
1108     end
1109
1110     if is_laser(handles)
1111         delete(handles.laser);
1112     end
1113
1114     if is_infrared(handles)
1115         delete(handles.infrared);
1116     end
1117
1118     if is_multispectral(handles)
1119         delete(handles.multispectral);
1120     end
1121
1122     delete(hObject);
1123 end
1124

```