Autonomous Robots (2021) 45:435–456
https://doi.org/10.1007/s10514-021-09978-5

A behavior-based framework for safe deployment of humanoid robots

Nicola Scianca[1] · Paolo Ferrari[1] · Daniele De Simone[1] · Leonardo Lanari[1] · Giuseppe Oriolo[1]

Received: 17 February 2020 / Accepted: 11 February 2021 / Published online: 14 March 2021
© The Author(s) 2021

Abstract
We present a complete framework for the safe deployment of humanoid robots in environments containing humans. Proceeding from some general guidelines, we propose several safety behaviors, classified in three categories, i.e., override, temporary override, and proactive. Activation and deactivation of these behaviors is triggered by information coming from the robot sensors and is handled by a state machine. The implementation of our safety framework is discussed with respect to a reference control architecture. In particular, it is shown that an MPC-based gait generator is ideal for realizing all behaviors related to locomotion. Simulation and experimental results on the HRP-4 and NAO humanoids, respectively, are presented to confirm the effectiveness of the proposed method.

Keywords Humanoid robots · Safety · Coexistence · Behaviors · MPC

1 Introduction

The most recent paradigms for adopting robotic technologies in applications emphasize the role of collaboration between robots and humans. Since collaboration implies sharing a common environment, safety concerns immediately become relevant. In industrial contexts, these have been addressed through the introduction of lightweight, compliant manipulators (Bicchi and Tonietti 2004) and the development of new techniques for safe coexistence and interaction with humans (De Santis et al. 2008). Clearly, similar issues arise in service applications; for example, Kruse et al. (2013) give a survey of human-aware robot navigation and Tadele et al. (2014) provide a review of recent research on safety for domestic robots.

One of the most essential safety layers in a robot is related to the avoidance of obstacles, static or dynamic. Starting with the pioneering work by Khatib (1985) on artificial potential fields, a huge literature on the topic has flourished (Minguez et al. 2016). Recently, researchers have started looking at this issue in the context of safe human-robot coexistence and interaction (De Luca and Flacco 2012; Lacevic et al. 2013; Navarro et al. 2018). These methods, however, are almost invariably devoted to fixed-base manipulators or wheeled robots.

On the other hand, there exists a growing interest in the use of humanoids for assembly operations where access by fixed-base manipulators or wheeled robots is impossible. For example, a recent EU H2020 research project targeted the adoption of humanoids in aeronautic manufacturing (Kheddar et al. 2019). Indeed, there is a widespread view that humanoids represent a rather natural choice of platform in environments that are specially designed to accommodate humans. Whether one agrees or not, there is no doubt that the challenging problem of safe deployment of humanoid robots needs be addressed.

The design of safety layers for humanoids must account for their unique characteristics: in particular, the fact that they can displace their base only through steps and that balance must be maintained at all times during motion (Kajita et al. 2014). One of the first works showing a humanoid (ASIMO) safely navigating an environment populated by dynamic obstacles via vision-based replanning was due to

✉ Giuseppe Oriolo
oriolo@diag.uniroma1.it

Nicola Scianca
scianca@diag.uniroma1.it

Paolo Ferrari
ferrari@diag.uniroma1.it

Daniele De Simone
desimone@diag.uniroma1.it

Leonardo Lanari
lanari@diag.uniroma1.it

[1] Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, via Ariosto 25, 00185 Roma, Italy

Michel et al. (2005). More recent results range from reactive Model Predictive Control techniques (Bohorquez et al. 2016; Naveau et al. 2017) to full-fledged whole-body motion planners (Baudouin et al. 2011; Ferrari et al. 2019).

Another important body of research related to the reliability of humanoids originated from the 2015 DARPA Robotics Challenge, in which research teams competed to effectively control humanoids in environments designed to emulate a real-world disaster scenario (Krotkov et al. 2017; Atkeson et al. 2018). Many planning and control techniques developed in this context by the participating teams proved to be effective (Lim et al. 2017), providing strong inspiration for achieving robust operation of humanoids.

All the above works, to which many more could be added (Wieber et al. 2016), focus however on a single issue which is considered relevant for reliable operation, such as robust locomotion or collision avoidance. Furthermore, humans are generally absent in the considered scenario, and tasks are often executed in supervised autonomy with the aid of specifically designed user interfaces (Marion et al. 2017), as it happens in the DARPA Challenge. In the literature, there is no general study that looks at the safety problem from a global viewpoint in order to design a holistic framework for achieving safe operation of humanoids.

The objective of this paper is to propose a complete framework for the safe deployment of humanoid robots in environments that may contain humans. Proceeding from some general guidelines, we propose several safety behaviors, classified in three categories:

- override behaviors, which stop the execution of the current task to account for the presence of an immediate danger, leading to a state from which normal operation can only be resumed after human intervention;
- temporary override behaviors, that take control of the robot for the limited amount of time necessary to address safety concerns, after which task execution can be automatically resumed;
- proactive behaviors, which are aimed at increasing the overall level of safety by an adaptation or enhancement of the robot activity, without interrupting the current task.

Activation and deactivation of these behaviors is triggered by information coming from the robot sensors and is handled by a state machine. To allow this, the state of the robot is identified by the current context (essentially, the task the robot is executing) and all active behaviors.

We also discuss the implementation of our safety framework in a reference control architecture, showing in particular that all behaviors related to locomotion can be efficiently realized in an MPC setting. Two humanoid platforms are used to show the performance of the proposed method, i.e., HRP-4 in simulation and NAO in experiments.

The paper is organized as follows. Section 2 briefly reviews the existing safety standards for robots, while Sect. 3 formulates some general guidelines for safe deployment of humanoids. In Sect. 4 we enunciate the robot sensing capabilities assumed by our framework. An overview of the proposed safety behaviors is given in Sect. 5, while Sect. 6 provides a detailed description of each behavior. Section 7 presents the state machine that orchestrates activation and deactivation of the behaviors. A reference control architecture is outlined in Sect. 8, and the implementation of the proposed behaviors within such architecture is discussed in Sect. 9. Simulation and experimental results are presented in Sects. 10 and 11 , respectively. Section 12 presents additional results and discusses limitations and possible extensions of the proposed method. Section 13 offers some concluding remarks.

2 Safety standards

Standards codify a set of practices that inform the design and operation of technologies. A product does not necessarily have to follow international standards as, unlike laws and regulations, these are not mandatory. However, they often provide a guarantee of compliance with regulations which otherwise can be quite hard to accommodate. It is therefore advantageous whenever possible to follow an existing standard, as this simplifies the design process.

The main international standards are published by either IEC or ISO committees. The first body focuses on standards related to electronics, while the second covers the remaining areas. Standards are divided in three categories.

- *Type A* standards provide basic rules and guidelines for machine safety (e.g., ISO 12100 “Basic Concepts, Design Principles” and ISO 14121 “Principles of Risk Assessment”).
- *Type B* standards are further divided into two subtypes: B1 covers aspects such as safety distances or ergonomic principles (e.g., ISO 13857 “Safety Distances”). B2 describes rules concerning protective equipment for different applications (e.g., IEC 13850 “Emergency Stop”).
- *Type C* standards refer to specific kinds of machines or areas of application, such as robots, and describe practical requirements and precautionary measures relating to all significant risks (e.g., see ISO 10218 “Robots and Robotic Devices—Safety Requirements for Industrial Robots”). Type C standards refer to Type A and B standards for generalities but may deviate from them when needed by the application.

The fundamental standard for industrial robot safety is ISO 10218, which highlights three particular aspects.

First, the standard suggests that means must be provided for the control and/or the release of hazardous energy stored in the robot. Examples of energy storage sources are batteries, springs and gravity.

The second aspect concerns safety stops. All robotic systems should have both a protective and an emergency stop function. Protective stops are used for risk reduction and can be activacted and deactivated automatically. Emergency stops are used in dangerous situations and require manual intervention.

Finally, ISO 10218 defines conditions for safe human-robot collaborative operation, identifying in particular four modes:

- *Safety-rated monitored stop*. The robot must stop whenever a human enters the shared workspace. This method does not allow collaborative work but only coexistence in the workspace. The robot may resume automatic operation when the human leaves.
- *Hand guiding*. The human can move the robot by physically interacting with it, e.g., to allow simplified path/point teaching. When provided, hand guiding equipment (such as a joystick) shall be located close to the end-effector. This mode must provide an emergency stop, an enabling device and safety-rated monitored speed limit.
- *Speed and separation monitoring*. The robot progressively reduces its speed as the human approaches. Failure to maintain the desired relative speed or separation distance will trigger a protective stop.
- *Power and force limiting*. In this mode, humans and robots can safely interact with little or no additional safety components because the robot force/power are bounded by design or control. Safe bounds are determined following ISO 10218-2 and ISO/TS 15066.

Almost all the above mentioned standards are specifically devised for industrial fixed-base manipulators, with only few exceptions addressing the case of wheeled mobile robots. No existing standard explicitly considers humanoid robots, whose peculiar nature must be properly taken into account.

3 Safety guidelines

With an eye to the standards discussed in the previous section, we provide here a qualitative description of the guidelines that inspired the design of our safety behaviors. In particular, we argue that the following recommendations should be followed for safe operation of a humanoid robot.

- *Watch what you're doing*. The robot should watch its main area of operation. When performing a manipulation task, it will therefore look at its hand(s) and/or at the object to be manipulated. When performing a locomotion task, it should direct its gaze towards the area where it is about to step.
- *Be on the lookout*. If the robot is idle, then it should scan the surroundings to identify possible sources of danger. In particular, if a moving object (e.g., a human) is detected, the robot should keep an eye on it.
- *Evade if you can*. When a moving object approaches, the humanoid robot should perform an evasive action, if this can be done safely.
- *Halt if you must*. In a situation of clear and present danger, the robot should terminate any activity and stop as soon as possible.
- *Beware of obstacles*. In the vicinity of unexpected objects, robot velocities and forces should be modified, scaled down or even zeroed in order to reduce potential damage in the case of a collision.
- *Look for support*. When locomotion is expected to be challenging (e.g., on stairs), the robot should try to establish additional contact with the environment (e.g., with a handrail). The possibility of improving balance by adding contacts should also be considered whenever a non-negligible risk of falling is detected.
- *Protect yourself*. In the imminence of a potentially damaging event, such as an unavoidable fall, the robot should act so as to minimize damage to itself and/or the environment.

Some of these guidelines will directly result into one or more safety behaviors (Sects. 5 and 6) that are activated to improve the level of safety when necessary. Other guidelines must also be taken into account at the basic planning/control stage. For example, *watch what you're doing* generates a behavior aimed at increasing the level of safety (*scan*) but also calls for visual-servoed manipulation/locomotion strategies during normal operation; the *look for support* guideline is reflected in a safety behavior (*add_contact*) but has also consequences at the planning stage (e.g., generation of stair climbing motions must include handrail grasping and releasing).

4 Sensing assumptions

We now specify which information must be made available by the robot sensory system (or by external sensing devices) to implement the proposed safety framework. We shall not discuss in any detail the perception processes that provide such information.

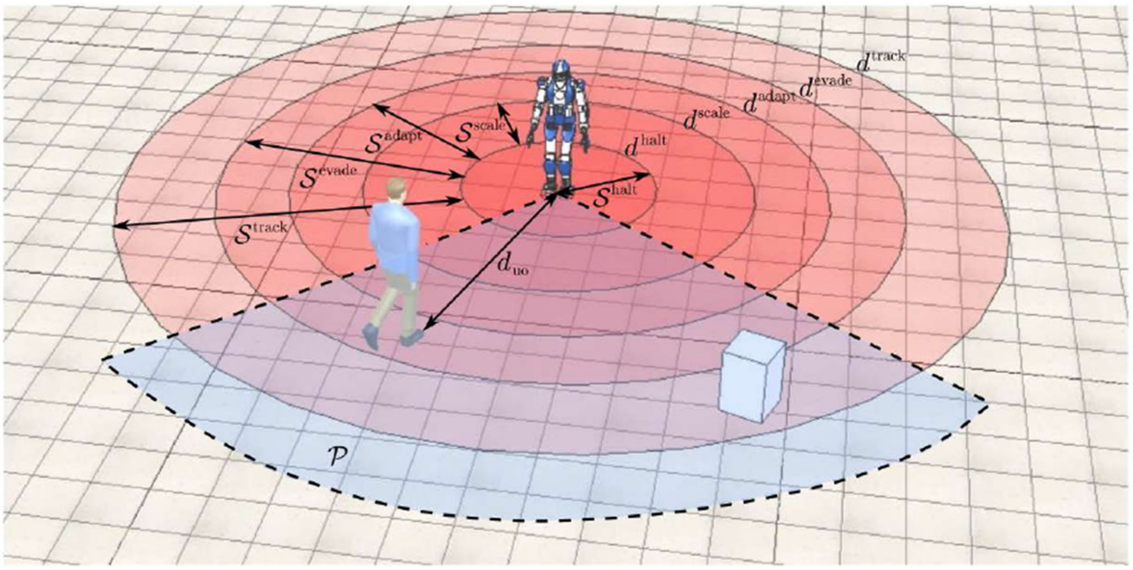

Fig. 1 The humanoid robot can identify unexpected objects within a perception area $\mathcal{P}$ and measure the minimum distance d_{uo} to the closest of them. Also shown are the various safety areas (with the associated thresholds) defined in Sect. 6.2

Throughout the paper, it is assumed that at any time instant the humanoid robot has a certain level of *awareness* about its own state and the surrounding environment, defined as follows:

- The robot knows whether there are *unexpected objects* (i.e., objects that are not present in the available map of the environment) within a perception area $\mathcal{P}$, whose extension depends on the specific sensory system, and in particular can measure the position (e.g., range and bearing) of the nearest point on each such object. As a consequence, it can compute the distance d_{uo} to the closest unexpected object in $\mathcal{P}$ (see Fig. 1). If there is no unexpected object in $\mathcal{P}$, d_{uo} is set to ∞. Techniques for identifying unexpected objects are typically based on a comparison between the predicted and the actual scene; for example, Radke et al. (2005) consider change detection in images.
- The robot is able to establish whether the closest unexpected object in $\mathcal{P}$ is *moving* or *stationary* (e.g., a human walking vs. some misplaced furniture). In practice, this can be done by looking at significant variations of d_{uo} (Stark et al. 2016) once the effect of the robot's own motion has been removed. The f_{mo} flag is used to specify whether the unexpected object is moving or not.
- The robot can detect *unexpected contacts* with the environment, indicated by the f_{uc} flag. Depending on the contact detection method, other information may be available, such as the location of the contact point on the robot body or the interaction force (Flacco et al. 2016).
- The robot knows the current risk of fall, represented by r_{fall}. For example, r_{fall} can be estimated from the position of the Zero Moment Point based on inertial measurements (Ogata et al. 2007).
- The robot knows the location of *contact surfaces* that can be reached without stepping from its current posture. Contact surfaces are surfaces (or points) of the environment with which the robot may safely establish a contact for additional support (Caron et al. 2017). The existence of reachable contact surfaces is encoded in the f_{cs} flag.
- The robot knows l_{battery}, its current battery level.

5 Overview of safety behaviors

This section provides a general description of the behaviors adopted by the humanoid robot to increase the level of safety for itself and the environment, which may include humans. In particular, three categories of safety behaviors are introduced, i.e., *override*, *temporary override* and *proactive*. We explain the idea behind each behavior and the situation in which it will be activated. A formal description, with detailed triggering conditions for each case, will be given in the next section.

5.1 Override behaviors

Override behaviors stop the execution of the current task and lead to a state from which normal operation can only be resumed after human intervention. They are intended as a way to react to unexpected and dangerous situations from which it would not be safe (or even possible) to resume the task automatically. We define two override behaviors:

- *halt*. In many situations, robot operation becomes critical: for example, when the battery level l_{battery} is too low, or when the distance d_{uo} to an unexpected moving obstacle goes below a certain threshold. In these cases, the *stop if you must* guideline indicates that the robot should immediately abort any task. The *halt* behavior is an emergency stop procedure which interrupts any operation as quickly as possible. However, one should keep in mind that a humanoid robot, especially during locomotion, must stop in such a way to maintain balance.
- *self-protect*. While it is obviously desirable to avoid falling altogether, there are several reasons which might lead to a loss of balance, for example a hardware/software fault, or an unexpected collision. To properly handle this event, one can design a *self-protect* behavior to be activated during falls, as suggested by the *protect yourself* guideline. In practice, when the robot detects an unrecoverable loss of balance, it must immediately adopt measures aimed at minimizing damage to itself and the environment.

5.2 Temporary override behaviors

Some events which cause safety concerns require the robot to stop task execution for a limited amount of time. As soon as these concerns have been properly handled, task execution can resume automatically. In particular, this is the case of the following behaviors:

- *stop*. This behavior is activated when external circumstances suggest to stop walking as a precaution, in application of the *beware of obstacles* guideline. This is for example the case of an unexpected object moving in the vicinity of the robot; in this situation, locomotion is interrupted and only resumed if the object leaves the area. The *stop* behavior is also used for transitioning from normal operation to other safety behaviors (e.g., to *track*) and vice versa (e.g., from *evade*). Note that *stop* differs from *halt* because it is more graceful and does not require human intervention to restart the robot.
- *evade*. Following the *evade if you can* guideline, if an unexpected moving object tends to approach the robot, this performs an evasive maneuver to avoid collision. At the end of the maneuver, the robot can resume normal operation as soon as the object does not constitute a threat anymore.
- *add_contact*. This behavior, which descends from the *look for support* guideline, allows the robot to establish new contacts for additional support whenever it is standing and the risk of fall is estimated to be non-negligible. In fact, we consider intrinsically risky trying to establish a new contact while the robot is walking. Besides, if the robot is ascending or descending stairs, additional contact with a handrail should have already been taken into account at the planning stage.
- *track*. If an unexpected moving object has been detected in its vicinity, the robot keeps its gaze directed at it, as suggested by the *be on the lookout* guideline. Note that this behavior can only be activated when the robot is idle or performing an observation task (see Sect. 6.1): in any other case, averting the gaze from the current task can be dangerous (*watch what you're doing* guideline). In particular, if the robot is walking, it will need to stop before starting to track the object.

5.3 Proactive behaviors

Proactive behaviors are actions intended to increase the overall safety level by calling for an adaptation or enhancement of the current robot activity. They include:

- *scan*. During manipulation or locomotion, the robot keeps its gaze directed at the main area of operation, as suggested by the *watch what you're doing* guideline. If the robot is idle, then it scans its surroundings, in application of the *be on the lookout* guideline.
- *adapt_footsteps*. During locomotion, the robot is in general controlled via high-level directives, such as tracking a reference velocity, or reaching a specific location in the workspace. The *adapt_footsteps* behavior, inspired to the *beware of obstacles* guideline, allows the robot to locally modify its footstep plan to avoid collision with stationary unexpected objects in its path.
- *scale_velocity-force*. If a nearby unexpected object is detected during manipulation, the robot must decrease all velocities and forces associated to the current manipulation task to reduce the risk of collision or the associated damages, as indicated by the *beware of obstacles* guideline.

6 Behavior-based safety framework

The activation and deactivation of the various safety behaviors is triggered by the information coming from the sensory system as described in Sect. 4 and depends on the current context (see Fig. 2).

In this section, we define the possible contexts and discuss the various safety areas used for behavior activation. Then, we describe each behavior in detail. Transitions to, from and among safety behaviors are actually controlled by a state

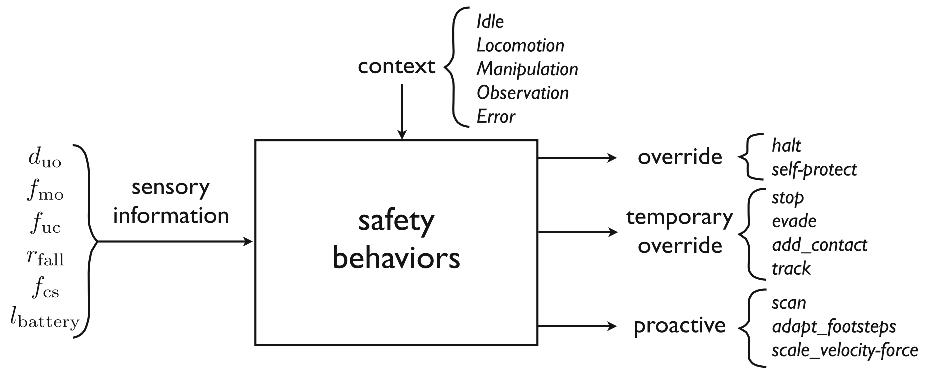

Fig. 2 The activation of safety behaviors depends on the current context and is triggered by sensory information

machine, in which states are defined as a context followed by one or more active behaviors. The structure of the state machine will be discussed in Sect. 7.

6.1 Contexts

The robot *contexts*[1] characterize what the robot is doing at a certain time instant. We identify five robot contexts:

- *Idle*. The humanoid is standing in double support at a fixed position and not performing any particular task.
- *Locomotion*. The humanoid is moving in the environment by taking steps. This includes walking, multi-contact locomotion and ascending/descending stairs.
- *Manipulation*. The humanoid is standing and executing a manipulation task that does not require any stepping.
- *Observation*. The humanoid is standing and executing a high-level observation task (e.g., find an object on a table). No locomotion or manipulation task is simultaneously being executed.
- *Error*. The robot is on hold until restarted by human intervention.

The first four contexts are associated to *normal operation*[2] (i.e., no safety behavior is active, except for *scan*) but also to operation under temporary override or proactive safety behaviors. *Error* is the only *emergency* context, to which the robot is released from override behaviors (*halt*, *self-protect*); from this context, normal operation cannot be resumed automatically.

[1] The definition of contexts could also take into account the various environments: for example, *Locomotion on flat ground* could be different from *Locomotion on stairs*, e.g., in terms of acceptable risk of fall.

[2] Note that an *Idle* context can be included in a mission plan as an intentional pause, e.g., at the transition between different phases or for debugging purposes.

6.2 Safety areas and thresholds

All safety behaviors are triggered by a certain measured quantity becoming larger or smaller than some given threshold. The most relevant of these quantities is the distance d_{uo} between the robot and the closest unexpected object, for which we specify five thresholds, i.e., in decreasing order: d^{track}, d^{evade}, d^{adapt}, d^{scale}, d^{halt}. As shown in Fig. 1, these thresholds on d_{uo} implicitly define five (partially overlapping) annular areas[3] around the robot:

- $\mathcal{S}^{\mathrm{track}}$, defined by $d^{\mathrm{halt}} \leq d_{\mathrm{uo}} \leq d^{\mathrm{track}}$. If the robot is not performing any locomotion or manipulation task (context *Idle* or *Observation*) and an unexpected moving object enters $\mathcal{S}^{\mathrm{track}}$, the robot will start tracking it visually. If the robot is walking (context *Locomotion*), it will have to stop before starting to track the object (walking without watching the stepping area would be unsafe).
- $\mathcal{S}^{\mathrm{evade}}$, defined by $d^{\mathrm{halt}} \leq d_{\mathrm{uo}} \leq d^{\mathrm{evade}}$. If the robot is not performing any task (context *Idle*) and an unexpected moving object enters $\mathcal{S}^{\mathrm{evade}}$, the robot will execute an evasion maneuver.
- $\mathcal{S}^{\mathrm{adapt}}$, defined by $d^{\mathrm{halt}} \leq d_{\mathrm{uo}} \leq d^{\mathrm{adapt}}$. If a walking robot (context *Locomotion*) detects an unexpected stationary object in $\mathcal{S}^{\mathrm{adapt}}$, it will adapt the footstep plan to avoid collisions.
- $\mathcal{S}^{\mathrm{scale}}$, defined by $d^{\mathrm{halt}} \leq d_{\mathrm{uo}} \leq d^{\mathrm{scale}}$. If the robot is performing a manipulation task (context *Manipulation*) and an unexpected moving object is detected in $\mathcal{S}^{\mathrm{scale}}$, the robot will reduce all velocities and forces associated to the current manipulation task.

[3] The definition of areas around the robot is also suggested by *proxemics*, the study of the use of space in social interaction (Hall 1966). Its goal is to describe and characterize the distances between humans in different social contexts, and the way these are established and perceived. In robotics, proxemics can serve as a basis to model robot behavior in interaction with humans (Rios-Martinez et al. 2014), which is especially relevant with humanoid robots as they are designed to allow for natural and comfortable interactions.

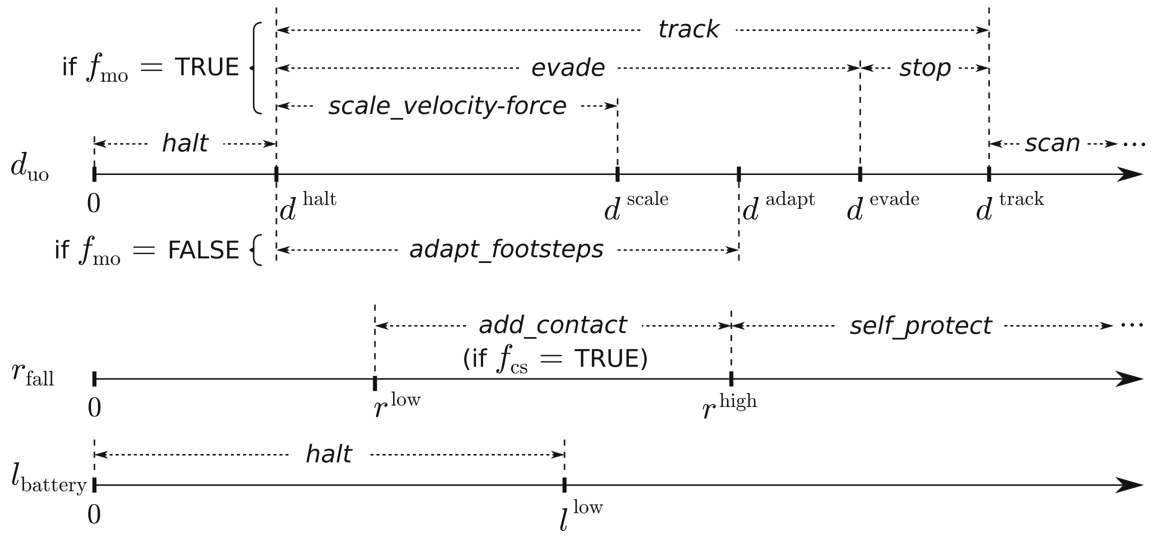

Fig. 3 The different thresholds used for quantities d_{uo} (top), r_{fall} (center) and l_{battery} (bottom), each with its associated behaviors. Consider that the actual activation of a behavior depends on the current context (not shown here). The only situation which does not appear in this representation is the triggering of *halt* due to an unexpected contact (f_{uc} = TRUE)

- $\mathcal{S}^{\text{halt}}$, defined by $d_{\text{uo}} \leq d^{\text{halt}}$. This is the innermost safety area. If any unexpected object is detected in $\mathcal{S}^{\text{halt}}$, the robot will terminate all operations, irrespective of the current context.

As for the estimated risk of fall, as soon as r_{fall} becomes significant ($r^{\text{low}} \leq r_{\text{fall}} \leq r^{\text{high}}$), the robot will establish a new contact for additional support, provided that a suitable surface is available (f_{cs} = TRUE). If a fall is considered inevitable ($r_{\text{fall}} > r^{\text{high}}$), the robot will take measures aimed at minimizing damages.

Finally, operations are terminated also when the battery level l_{battery} is too low ($l_{\text{battery}} \leq l^{\text{low}}$).

Figure 3 summarizes the activation of behaviors as a function of the values of d_{uo}, r_{fall} and l_{battery}.

The reader may have noticed that the safety behaviors related to collision avoidance (such as *stop*, *evade*, *halt*) discussed so far are only driven by the distance between the robot and the unexpected object, and do not take into account their relative velocity. This choice was made for the following reasons:

– Velocity information may not be available, or in any case it may be computationally more costly to obtain. Indeed, in Sect. 4 we have assumed that the sensory system only provides distance measurements.
– If an unexpected obstacle enters a safety area (e.g., $\mathcal{S}^{\text{evade}}$), its relative velocity w.r.t. the robot is certainly directed towards the half-plane containing the robot (and tangent to the area). Therefore, our choice corresponds to a conservative viewpoint in which any such relative velocity is considered dangerous, independently from its specific orientation and magnitude.
– Working out an extension of the method for the case in which relative velocity is measured and used by the collision avoidance behaviors is relatively easy, see Sect. 12.3.

6.3 Definitions of behaviors

The formal definition of each safety behavior requires the specification of:

- one or more *contexts* from which the behavior can be activated;
- a *trigger*, indicating which particular event or piece of information will cause the behavior to activate, along with specific actions that occur upon triggering, such as deactivating other behaviors;
- the *action*, i.e., which activities are associated to the behavior;
- a *release*, which is an event or piece of information that causes the behavior to deactivate, again including specific actions that occur upon deactivation.

In the following, we define override, temporary override and proactive behaviors in this order.

halt

- **Context**: *Idle*, *Locomotion*, *Manipulation*, *Observation*.

- **Trigger**: $d_{\text{uo}} \leq d^{\text{halt}}$ (T1) OR $f_{\text{uc}} = \text{TRUE}$ (T2) OR $l_{\text{battery}} \leq l^{\text{low}}$ (T3).
  Triggering permanently deactivates any active behavior and inhibits all others from activating, except for *self-protect* in case of a fall.
- **Action**: Depends on the context and the trigger:
  - If the context is *Idle* and the trigger is T1 or T3, the robot will augment its support polygon and/or assume a low-impact configuration (e.g., by folding its arms).
  - If the context is *Idle* and the trigger is T2, the robot will decrease joint stiffness on the kinematic chain where the contact has occurred, provided that the latter is on the upper body. Otherwise, the robot will simply maintain its current posture.
  - If the context is *Locomotion*, *Manipulation* or *Observation* the robot will abort the task and stop any motion as soon as possible, regardless of the trigger.
- **Release**: When the action is completed.
  Upon release, context is changed to *Error*.

self-protect

- **Context**: *Idle*, *Locomotion*, *Manipulation*, *Observation*.
- **Trigger**: $r_{\text{fall}} > r^{\text{high}}$.
  Triggering permanently deactivates any active behavior and inhibits all others from activating.
- **Action**: The robot will abort the task and act so as to minimize the potential damage to itself and/or the environment. To this end, several aspects must be considered, including *(i)* how to fall, i.e., which internal posture to assume before impact to preserve robot integrity *(ii)* where to fall, i.e., how to choose the landing surfaces so as to avoid fragile objects.
- **Release**: When the action is completed.
  Upon release, context is changed to *Error*.

stop

- **Context**: *Locomotion*.
- **Trigger**: $d^{\text{evade}} < d_{\text{uo}} \leq d^{\text{track}}$ AND $f_{mo} = \text{TRUE}$.
- **Action**: The robot will stop walking, ending in a double support configuration. This is done before starting to track (and possibly later evade) an unexpected moving object; or at the end of an evasion maneuver.
- **Release**: When the action is completed.
  Upon release, context is changed to *Idle*.

evade

- **Context**: *Idle*
- **Trigger**: $d^{\text{halt}} < d_{\text{uo}} \leq d^{\text{evade}}$ AND $f_{\text{mo}} = \text{TRUE}$.
  Triggering changes context to *Locomotion*.
- **Action**: The robot will execute a reactive evasion maneuver so as to increase the distance to the unexpected moving object.
- **Release**: When $d_{\text{uo}} > d^{\text{evade}}$.
  Upon release, *stop* is activated in order to interrupt the evasion maneuver.

add_contact

- **Context**: *Idle*, *Manipulation*, *Observation*.
- **Trigger**: $r^{\text{low}} \leq r_{\text{fall}} \leq r^{\text{high}}$ AND f_{cs} = TRUE.
  Upon triggering, context is changed to *Idle* if it was *Manipulation* or *Observation*. Moreover, *track* is deactivated if active, and *evade* is inhibited from activation.
- **Action**: The robot will interrupt the task (if the context was *Manipulation* or *Observation*), and select and establish contact with an additional support point on the available surfaces.
- **Release**: When the action is completed.

track

- **Context**: *Idle*, *Observation*.
- **Trigger**: $d^{\text{halt}} < d_{\text{uo}} \leq d^{\text{track}}$ AND $f_{\text{mo}} = \text{TRUE}$.
  Triggering deactivates *scan* and changes the context to *Idle* if it was *Observation*.
- **Action**: The robot will interrupt any observation task (if the context was *Observation*) and direct its gaze at the closest unexpected object moving in $\mathcal{S}^{\text{track}}$.
- **Release**: When $d_{\text{uo}} > d^{\text{track}}$.
  Upon release, *scan* is activated.

scan

- **Context**: *Idle*, *Manipulation*, *Locomotion*.
- **Trigger**: Active by default unless *track* is active.
- **Action**: Depends on the context:
  - If the context is *Idle*, the robot will scan the surrounding environment.
  - If the context is *Locomotion*, the robot will scan the path ahead.
  - If the context is *Manipulation*, the robot will scan the area of operation.
- **Release**: Never.

adapt_footsteps

- **Context**: *Locomotion*.

- **Trigger**: $d^{\text{halt}} < d_{\text{uo}} \leq d^{\text{adapt}}$ AND $f_{\text{mo}} = \text{FALSE}$.
- **Action**: The robot will modify the current footstep plan as needed to avoid the closest unexpected object standing in the scene.
- **Release**: When $d_{\text{uo}} > d^{\text{adapt}}$.

scale_velocity-force

- **Context**: *Manipulation*.
- **Trigger**: $d^{\text{halt}} < d_{\text{uo}} \leq d^{\text{scale}}$ AND $f_{\text{mo}} = \text{TRUE}$.
- **Action**: Robot velocities and/or forces associated to the current task will be reduced.
- **Release**: When $d_{\text{uo}} > d^{\text{scale}}$.

Note the following points.

- Override behaviors force the robot to abort any task and deactivate/inhibit all other kinds of behavior.
- Under temporary override behaviors, any task is interrupted for a limited period of time.
- Among temporary override behaviors, *add_contact* deactivates other behaviors that would cause a conflict of context (*evade*) or steal an essential sensory resource (*track*); for the same latter reason, *track* deactivates *scan*, and *scan* cannot be activated when the context is *Observation*.
- When temporary override behaviors are released, the context will always be *Idle* with the *scan* behavior active. It is important to note that *in this condition control goes back to normal operation*: the robot is ready to resume any task that was interrupted, under prompting by the supervisory module that activates and sequences tasks.
- Finally, proactive behaviors do not interrupt active tasks.

7 State machine

In the proposed framework, activation and deactivation of safety behaviors are handled by a state machine. In particular, the *state* of the robot is uniquely identified by the current context and all active behaviors, and denoted as Context/*behavior_1*/*behavior_2*/…, with behaviors listed in the order of activation.

The complete list of the states is the following:

- Idle/*scan*
- Idle/*track*
- Idle/*scan*/*add_contact*
- Idle/*halt*
- Idle/*self-protect*
- Locomotion/*scan*
- Locomotion/*scan*/*adapt_footsteps*
- Locomotion/*scan*/*stop*
- Locomotion/*track*/*evade*
- Locomotion/*track*/*evade*/*adapt_footsteps*
- Locomotion/*track*/*stop*
- Locomotion/*halt*
- Locomotion/*self-protect*
- Manipulation/*scan*
- Manipulation/*scan*/*scale_velocity-force*
- Manipulation/*halt*
- Manipulation/*self-protect*
- Observation/
- Observation/*halt*
- Observation/*self-protect*
- Error/

Figure 4 gives a complete representation of the state machine. Transition from one state to another is instantaneous and corresponds to activation or deactivation of some behavior. For example, the transition from Locomotion/*scan* to Locomotion/*scan*/*adapt_footsteps* corresponds to the activation of the proactive behavior *adapt_footsteps*: the robot was walking and scanning the stepping area when an unexpected stationary object was detected in $\mathcal{S}^{\text{adapt}}$, so that it became necessary to adapt the footstep plan. Similarly, the transition from Locomotion/*track*/*evade* to Locomotion/*track*/*stop* corresponds to the deactivation of the temporary override behavior *evade*, which automatically triggers *stop*: the robot was performing an evasion maneuver which became unnecessary when the moving unexpected obstacle left $\mathcal{S}^{\text{evade}}$, so that motion was stopped. Note that *track* is still active in the final state after the transition because the object will still be in $\mathcal{S}^{\text{track}}$ after leaving $\mathcal{S}^{\text{evade}}$ (see Fig. 1).

The only *normal operation* states are Idle/*scan*, Locomotion/*scan*, Manipulation/*scan* and Observation/, in which no safety behavior is active apart from the default proactive behavior *scan*. In these states, control is entirely committed to the realization of the desired task.

Finally, examination of Fig. 4 confirms that:

- override behaviors lead to the Error/ state;
- temporary override behaviors ultimately lead to the Idle/*scan* state, that is a normal operation state from which the original task can be resumed;
- proactive behaviors (apart from *scan*) simply disappear in the end, leading back to the state from which they were activated.

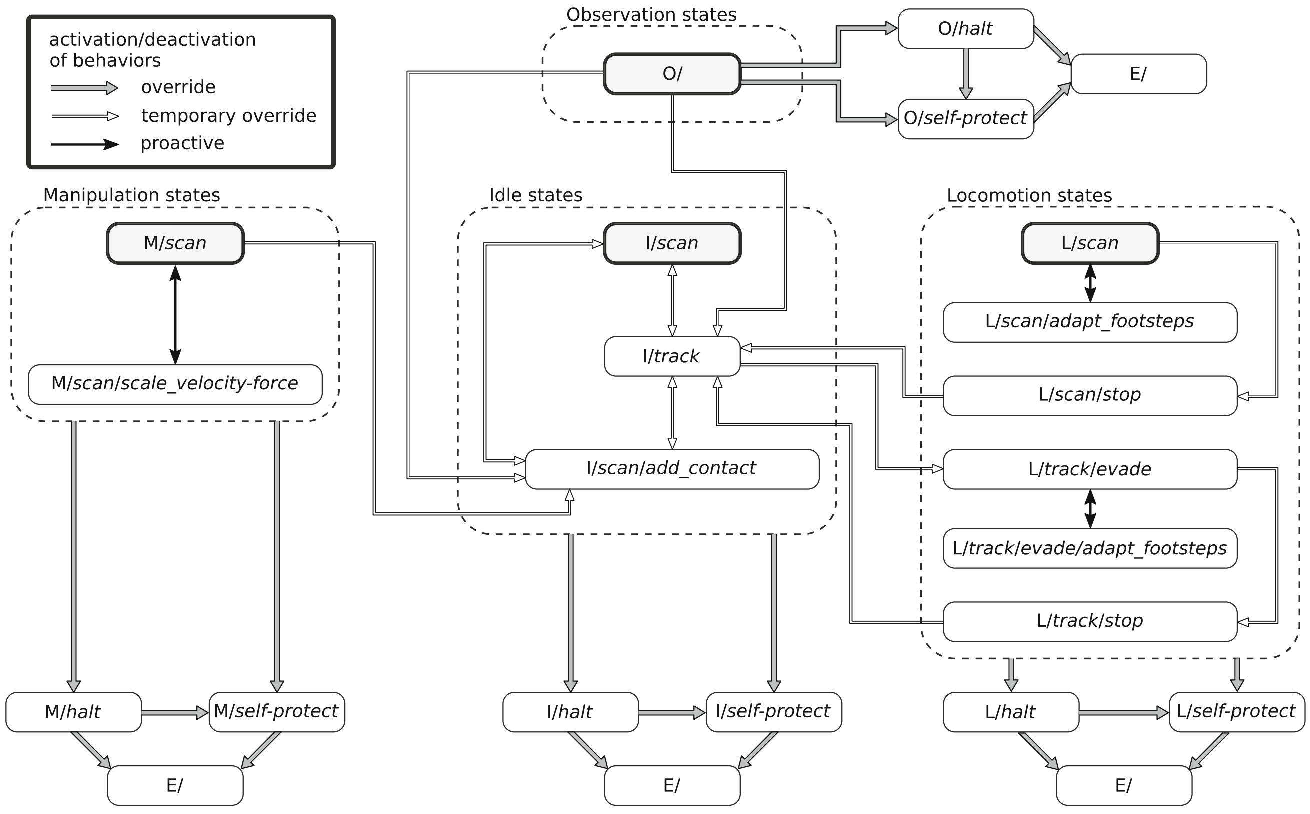

Fig. 4 A representation of the state machine governing the transitions among the proposed safety behaviors. For compactness, contexts within states are denoted only by their initial (I, M, L, O or E). A transition originating from the boundary of a dashed box (e.g., the box of Idle states) indicates that the transition may originate from any state in the box. Normal operation states (i.e., states where no safety behavior is active apart from the default behavior *scan*) are shown in gray

8 Control architecture

The implementation of the safety behaviors, which will be discussed in some detail in the next section, will obviously depend on the specific control architecture of the robot. In view of this, we describe here a possible structure which will be used in the rest of the paper for illustration.

Figure 5 shows a general overview[4] of the control architecture. Note that safety behaviors blocks appear in this scheme as either reference signal generators (*scan*, *track*, *add_contact*, *stop*, *evade*, *halt*, *self-protect*) or signal modifiers (*scale_velocity-force*, *adapt_footsteps*). Note the particular placement of the *self-protect* behavior, which will take control of the robot directly at the joint level. Activation and deactivation of the behaviors is handled in the background by the state machine described in the previous section.

In the rest of this section, we describe the core components of the control architecture without reference to the safety behaviors. The specific way in which each behavior is embedded in the architecture will be discussed in the next section.

[4] The control architecture of humanoids invariably includes a stabilization module, not present in Fig. 5, whose role is to guarantee balancing of the robot in all situations. This low-level module is not discussed since here it bears no relevance to our safety framework.

8.1 Camera motion generator

The camera motion generator is primarily (i.e., during normal operation) in charge of producing a suitable reference motion for the camera when an observation task is being executed (context *Observation*). An image-based visual servoing scheme (Chaumette and Hutchinson 2006) is used to move the camera so as to track as reference signal representing the motion of a desired feature in the image plane, e.g., for finding an object in the scene.

8.2 Hand(s) motion-force generator

During normal operation, the hand(s) motion-force generator is in charge of producing a suitable motion-force reference for the hand(s) when a manipulation task is being executed (context *Manipulation*). For example, when a certain grasp

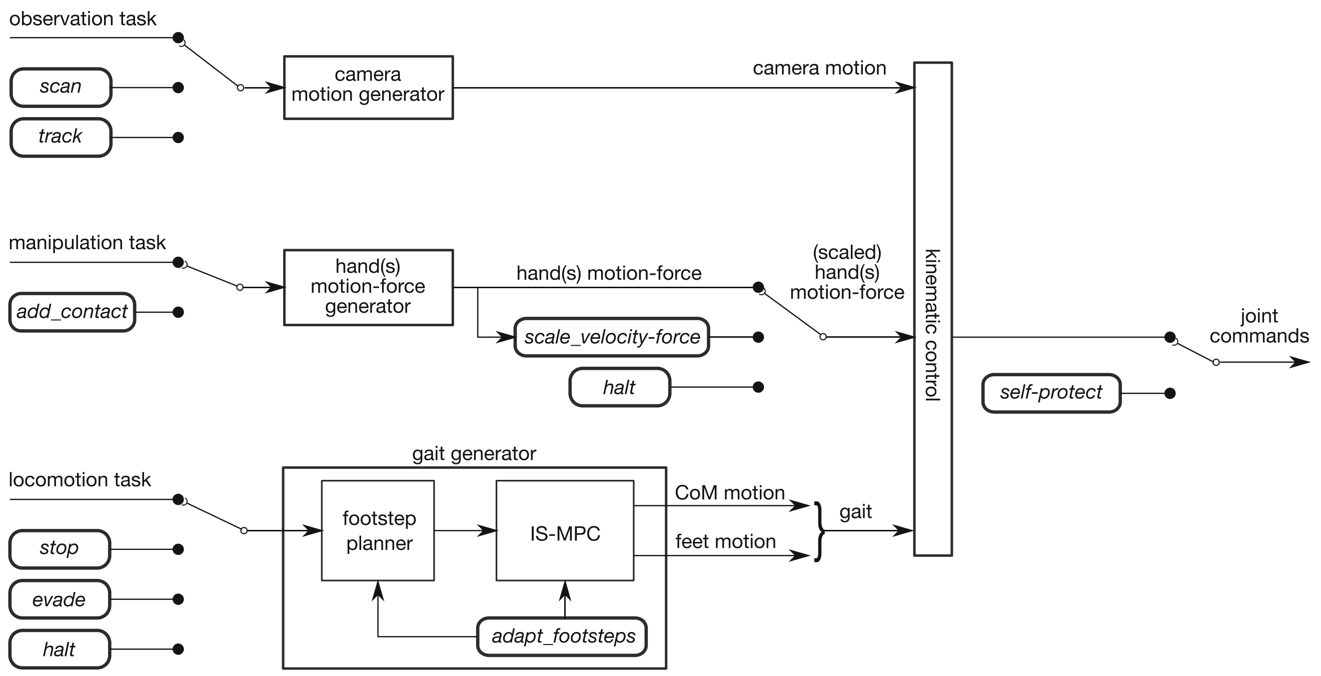

Fig. 5 The considered control architecture. The safety behaviors blocks appear as either signal generators or modifiers

must be executed, the module will plan a continuous motion of the hand(s) so as to reach the associated Cartesian posture.

8.3 Gait generator

The primary role of the gait generator is to produce suitable reference motions for the robot Center of Mass (CoM) in order to execute a locomotion task (context *Locomotion*). Since such a module is specific to humanoid robots, we will describe it in some detail, considering for compactness the case of flat ground. The algorithm at the core of the module can however be easily extended to work on non-flat terrains, in particular to deal with the presence of stairs (Zamparelli et al. 2018).

The gait generator used in this paper is based on Model Predictive Control (MPC). In particular, we adopt a method (Scianca et al. 2020) which makes use of two sequential modules running in real time (Fig. 5). The first module (footstep planner) generates a sequence of timed footsteps to realize high-level omnidirectional velocity commands v_x, v_y, ω. The second module (Intrinsically Stable MPC, or IS-MPC) produces a trajectory of the robot CoM and feet such that (1) the Zero Moment Point (ZMP) is always within the support polygon, thus guaranteeing that balance is maintained, and (2) that the CoM trajectory is bounded with respect to the ZMP trajectory, implying internal stability. Both modules require the solution of QP (Quadratic Programming) problems.

The high-level velocity commands v_x, v_y, ω that drive gait generation are known over a *preview horizon* $T_p = P \cdot \delta$ of P sampling intervals — each of duration δ — in the future. The footstep generation module plans footsteps over the same horizon, whereas IS-MPC plans CoM and ZMP trajectories over a *control horizon* $T_c = C \cdot \delta$. It is assumed that $P \geq C$, so that IS-MPC can take full advantage of the preview information.

8.3.1 Footstep planner

At each sampling instant t_k, the footstep planner receives in input the high-level reference velocities over the preview horizon, i.e., from t_k to $t_k + T_p = t_{k+P}$.

First, the footstep timing over T_p is determined[5] in the form $\mathcal{T}_s^k = \{T_s^1, \ldots, T_s^F\}$, where T_s^j is the step duration between the $(j-1)$-th and the j-th footstep, taken from the start of the single support phase to the next. Since the step duration is variable, the number F of footsteps falling within T_p may change with t_k.

Then, a sequence of F footsteps $(\hat{X}_f^k, \hat{Y}_f^k, \Theta_f^k)$ over the same interval is generated:

$$\hat{X}_f^k = (\hat{x}_f^1 \ \ \ldots \ \ \hat{x}_f^F)^T$$
$$\hat{Y}_f^k = (\hat{y}_f^1 \ \ \ldots \ \ \hat{y}_f^F)^T$$
$$\Theta_f^k = (\theta_f^1 \ \ \ldots \ \ \theta_f^F)^T,$$

[5] Step timing is chosen by a simple heuristic based on the velocity commands (Scianca et al. 2020).

where $(\hat{x}_f^j, \hat{y}_f^j, \theta_f^j)$ is the pose[6] (position + orientation) of the j-th footstep. To this end, we inject the high-level reference velocities into the following omnidirectional motion model

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix} \tag{1}$$

and then distribute the footsteps around the resulting trajectory in accordance with the timing $\mathcal{T}_s^k$. This is done by solving a sequence of two QP problems; the first computes the footstep orientations

$$\begin{cases} \min\limits_{\Theta_f^k} \sum\limits_{j=1}^{F} (\theta_f^j - \theta_f^{j-1} - \int\limits_{t_s^{j-1}}^{t_s^j} \omega(\tau) d\tau)^2 \\ \text{subject to} \quad |\theta_f^j - \theta_f^{j-1}| \leq \theta_{\max} \end{cases}$$

while the second computes the footstep positions

$$\begin{cases} \min\limits_{\hat{X}_f^k, \hat{Y}_f^k} \sum\limits_{j=1}^{F} (\hat{x}_f^j - \hat{x}_f^{j-1} - \Delta x^j)^2 + (\hat{y}_f^j - \hat{y}_f^{j-1} - \Delta y^j)^2 \\ \text{subject to kinematic constraints} \end{cases}$$

In the first QP problem, $\theta_{\max}$ is the maximum allowed rotation between two consecutive footsteps, while t_s^j is the timestamp of the j-th footstep. In the second, $(\hat{x}_f^0, \hat{y}_f^0)$ is the known position of the support foot at t_k and Δx^j, Δy^j are given by

$$\begin{pmatrix} \Delta x^j \\ \Delta y^j \end{pmatrix} = \int_{t_s^{j-1}}^{t_s^j} R_\theta \begin{pmatrix} v_x(\tau) \\ v_y(\tau) \end{pmatrix} d\tau \pm R_j \begin{pmatrix} 0 \\ \ell/2 \end{pmatrix},$$

where R_θ, R_j are the rotation matrices associated respectively to $\theta(\tau)$ and θ_j^f, ℓ is the chosen coronal distance between consecutive footsteps, and the sign of the second term alternates for left/right footsteps. The *kinematic constraints* in the second problem are built to guarantee that the footsteps are kinematically feasible for the specific humanoid being considered.

8.3.2 IS-MPC

Once the timed footstep plan is available, the IS-MPC module is called to generate the CoM motion and simultaneously adjust the footstep positions. The prediction model used by IS-MPC for the sagittal motion is

$$\begin{pmatrix} \dot{x}_c \\ \ddot{x}_c \\ \dot{x}_z \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ \eta^2 & 0 & -\eta^2 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ \dot{x}_c \\ x_z \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \dot{x}_z, \tag{2}$$

i.e., a Linear Inverted Pendulum (LIP) with a dynamic extension. Here, x_c and x_z are the x coordinates of the CoM and the ZMP, respectively, $\eta = \sqrt{g/h_c}$, and h_c is the height of the CoM which is constant in the LIP formulation. The control input of (2) is the ZMP velocity $\dot{x}_z$, which is assumed to be piecewise-constant over the sampling intervals. An identical equation holds for the evolution of the y coordinate (coronal motion).

At each t_k, the ZMP velocities $\dot{X}_z^k = (\dot{x}_z^k, \ldots, \dot{x}_z^{k+C-1})$, $\dot{Y}_z^k = (\dot{y}_z^k, \ldots, \dot{y}_z^{k+C-1})$, and the final footstep positions X_f^k, Y_f^k, over the control horizon are determined by solving the following QP problem:

$$\min \|\dot{X}_z^k\|^2 + \|\dot{Y}_z^k\|^2 + \beta(\|X_f^k - \hat{X}_f^k\|^2 + \|Y_f^k - \hat{Y}_f^k\|^2)$$

subject to:

- *ZMP constraints*. These impose that the ZMP lies at all times within the support polygon of the robot. To avoid nonlinearities, during double support phases the latter is approximated by a *moving constraint* (Aboudonia et al. 2017).
- *Kinematic constraints*. These are the same exact constraints already enforced by the second QP in the footstep planner.
- *Stability constraint*. To ensure that the CoM trajectory does not diverge w.r.t. the ZMP, we impose that

$$\sum_{i=0}^{C-1} e^{-i\eta\delta} \dot{x}_z^{k+i} = -\sum_{i=C}^{\infty} e^{-i\eta\delta} \dot{\tilde{x}}_z^{k+i} + \frac{\eta\,(x_u^k - x_z^k)}{1 - e^{-\eta\delta}}, \tag{3}$$

where

$$x_u^k = x_c^k + \dot{x}_c^k/\eta$$

is the so-called *divergent component of motion* x_u at time t_k (Takenaka et al. 2009). The left-hand side of (3) contains the ZMP velocities within the control horizon (decision variables), whereas the right-hand side includes the ZMP velocities beyond the horizon, collectively referred to as the *tail*, which are unknown (hence the tilde) and must be conjectured to obtain a causal version of the constraint. For example, one may set the tail to zero (*truncated tail*) if the high-level velocity commands

[6] The hat on the position coordinates indicates that these are candidates values which will be later adjusted by IS-MPC. Orientations are final because their inclusion in the MPC formulation would make the problem nonlinear.

indicate that the robot should immediately stop[7]. A more general choice is to use an *anticipative tail*, i.e., surmise the value of such velocities on the basis of the preview information encoded in the footstep plan.

It may be shown (Scianca et al. 2020) that the IS-MPC scheme is recursively feasible and internally stable (in other words, ZMP-to-CoM stable) provided that the preview horizon is sufficiently long, i.e., if P is large enough. Overall, this leads to a gait generation scheme which is anticipative, and therefore suitable not only for regular gaits but also for executing sudden avoidance maneuvers, as well as temporary or emergency stops—all actions which are required in our safety framework.

9 Implementation of behaviors

In this section we discuss the implementation of safety behaviors inside the control architecture of Fig. 5. The main focus will be on those behaviors that are related to locomotion, which are specific to humanoids. For the others we shall simply provide pointers to existing work.

9.1 halt

The *halt* behavior realizes an emergency stop in the presence of immediate threats. As explained in Sect. 6.3, the behavior is declined differently depending on the context and/or trigger. Here, we consider the case in which the context is *Locomotion*.

Activation of *halt* is realized via two mechanisms:

- The high-level velocity commands v_x, v_y, ω are immediately set to zero;
- As a consequence, the truncated tail is used in the stability constraint (3).

9.2 self-protect

As shown by Fig. 5, the *self-protect* behavior takes command of the robot at the joint level and therefore overrides all the preceding control architecture. When an impending fall is detected, this behavior acts so as to minimize the potential damage to itself and/or the environment. In the following, we refer the reader to some relevant works in this direction.

Choosing how to fall means assuming a proper configuration for reducing the effect of the impact with the floor. Fujiwara et al. (2003, 2004) present a controller that limits the impact force, based on the idea of lowering the CoM as soon as possible by crouching or knee-bending; Yasin et al. (2012) use a similar approach to manage forward or backward falls. Braghin et al. (2019) compute whole-body trajectories aimed at minimizing damage due to falling through an optimization-based control strategy.

In addition to lowering the impact force, strategies for absorbing the impact are also useful. A control method that combines robot reconfiguration and post-impact compliance is proposed by Samy and Kheddar (2015): during the falling phase, the robot is kept away from fall singularities, i.e., postures in which impact forces would be poorly absorbed. After the impact, compliance control is activated, with the motors behaving as spring-dampers.

Besides how to fall, it is also important to choose where to fall. Yun et al. (2009) introduce a controller which changes the fall direction in order to avoid specific objects or parts of the environment. This method was extended to multiple objects by Nagarajan and Goswami (2010).

9.3 stop

The *stop* behavior temporarily interrupts locomotion in preparation for tracking a moving object, or at the end of an evasive maneuver.

Activation of *stop* is realized via two mechanisms:

- The high-level velocity commands v_x, v_y, ω go from their current value to zero over a fixed *arrest time*;
- As a consequence, the anticipative tail is used in the stability constraint (3).

Once the motion has been stopped, the state becomes Idle/*track* (see Fig. 4). If the unexpected moving object leaves $\mathcal{S}^{\text{track}}$, the robot goes to the normal operation state Idle/*scan*, where the original locomotion task can be resumed.

The different effect of *stop* vs. *halt* will be illustrated via simulation in the next section.

9.4 evade

The *evade* behavior is realized by sending to the robot high-level velocity commands for avoiding an unexpected moving object that has entered the $\mathcal{S}^{\text{evade}}$ area.

To devise such commands, consider the geometry of the problem as shown in Fig. 6. The angle θ_{obs} under which the robot sees the moving object is directly measured by the robot (see the first assumption in Sect. 4). The chosen direction of evasion is represented by $\boldsymbol{n}_{\text{eva}}$, with the robot moving backwards so as to keep the object in its field of view.

[7] This corresponds to imposing a terminal condition known as *capturability constraint* in the MPC formulation (Scianca et al. 2020).

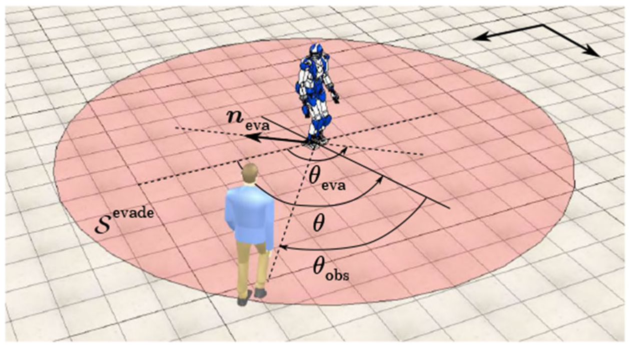

Fig. 6 The geometry of evasion. A moving object enters $\mathcal{S}^{\text{evade}}$. The chosen direction of evasion is $\boldsymbol{n}_{\text{eva}}$

While the humanoid can in principle move in any direction with motion model (1), studies on humans (Mombaur et al. 2010) indicate that time-efficient locomotion requires the orientation of the body to be tangent to the path, similarly to what happens in nonholonomic mobile robots. For this reason, we adopt the unicycle as template model for evasive maneuvers:

$$\begin{aligned}\dot{x} &= v\cos\theta\\ \dot{y} &= v\sin\theta\\ \dot{\theta} &= \omega, \end{aligned} \tag{4}$$

where x, y, θ denote now the position and orientation of the unicycle, and v, ω are its driving and steering velocity inputs. The latter are chosen as

$$v = -\bar{v} \tag{5}$$

$$\omega = k\,(\theta_{\text{eva}} - \theta). \tag{6}$$

While the driving velocity is set to a constant negative value, the steering velocity forces the unicycle (4) to align smoothly with the desired orientation, chosen as[8] $\theta_{\text{eva}} = \theta + \theta_{\text{obs}} - \text{sign}(\theta_{\text{obs}}) \cdot \pi/2$. As a result, we obtain $\omega = k\,(\theta_{\text{obs}} - \text{sign}(\theta_{\text{obs}}) \cdot \pi/2)$, which can be implemented using only on-board measurements. Alternatively, the proportional control law (6) may be replaced with

$$\omega = k\,\text{sign}(\theta_{\text{eva}} - \theta), \tag{7}$$

to make the evader perform the evasion maneuver with a constant curvature radius.

The final step is to send the control inputs (5 and 6) to the gait generator, with the adjustment $v_x = v\cos\theta, v_y = v\sin\theta$ (while ω is unchanged).

[8] This evasion strategy is called *move aside*, as the humanoid moves (backwards) in a direction that is orthogonal to the object line of approach; other strategies are possible (Cognetti et al. 2017).

An observation is in order about obstacle avoidance during evasion maneuvers. Since evasion is a safety behavior, it leads the humanoid to an area which was not contemplated in the original motion plan. Obstacles in this area should therefore be considered as *unexpected*, regardless of their being represented or not in the environment map. With this rationale, *any* obstacle inside $\mathcal{S}^{\text{adapt}}$ will trigger the *adapt_footsteps* behavior during an evasion maneuver (see the first simulation in the next section).

9.5 add_contact

The *add_contact* behavior can be activated in *Idle*, *Observation* and *Manipulation*. The current task is interrupted and context is immediately changed to *Idle* to allow the robot to establish an additional contact. This requires first choosing a posture where one robot body (typically, a hand) is in contact with a reachable contact surface, whose existence is indicated by the f_{cs} flag. The hands motion-force generator module is then invoked to plan a continuous motion-force reference that will achieve the chosen desired posture.

When one or more contact surfaces are reachable, it is necessary to decide which contact to choose. Clearly, it is essential that the added contact improves balance. To this end, one may use concepts such as the generalization of ZMP support areas to the case of multiple non-coplanar contacts (Harada et al. 2006; Caron et al. 2017). These aspects have also been studied in the more general field of multi-contact planning for locomotion and manipulation (Lengagne et al. 2013; Mandery et al. 2015; Werner et al. 2016); an interesting summary is given by Bouyarmane et al. (2019).

We emphasize that additional contacts may also be exploited outside the safety framework, i.e., during normal operation. For example, if the robot is required to climb a staircase it is sensible to take advantage of the handrail if available. In this case, however, the appropriate contacts must be integrated in the locomotion/manipulation task being executed.

9.6 scan and track

Both the *scan* and *track* behaviors are realized by invoking the camera motion generation module.

The *scan* behavior uses an artificial image feature as a reference signal. If the context is *Idle*, the artificial feature moves cyclically throughout the surrounding area so as to achieve complete coverage. In *Locomotion* or *Manipulation*, the feature is fixed at the center of the specific area of interest.

In the *track* behavior, the image feature will be directly the closest point on the unexpected moving object that has triggered the behavior.

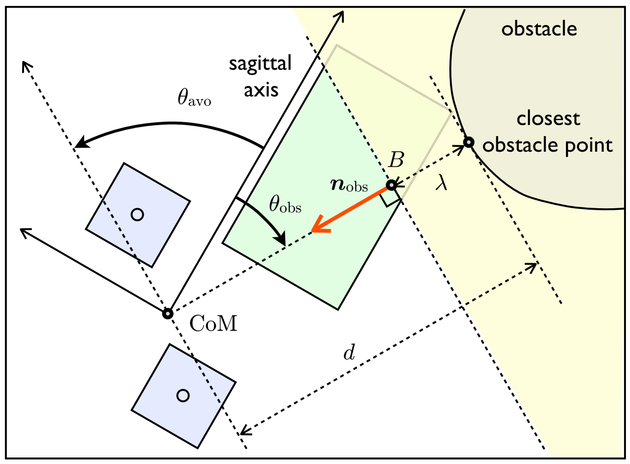

Fig. 7 The geometry of footstep adaptation with the definition of the relevant quantities. The current robot placement is defined by its CoM. Also shown are the current footstep locations (light blue), the kinematically feasible zone (green) and the forbidden zone (yellow) due to the presence of the obstacle for the next footstep (Color figure online)

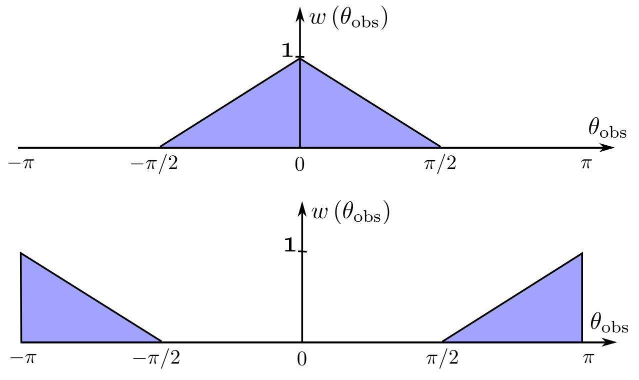

Fig. 8 The weight function $w(\theta_{\text{obs}})$. Top: if the robot is walking forward. Bottom: if the robot is walking backwards

9.7 adapt_footsteps

If the context is *Locomotion* and a stationary unexpected object is detected in $\mathcal{S}^{\text{adapt}}$, the *adapt_footsteps* behavior is invoked. This is realized by minor modifications of the two modules that constitute the gait generator, i.e., the footstep planner and IS-MPC.

Refer to Fig. 7 for the geometry of the problem and the definition of the relevant quantities. In particular, the range d and bearing θ_{obs} are directly measured by the robot (again, see the first assumption in Sect. 4), while θ_{avo} is defined as

$$\theta_{\text{avo}} = \theta_{\text{obs}} - \text{sign}(\theta_{\text{obs}}) \cdot \pi/2.$$

Within the footstep planner, the first modification is in the QP problem used to compute the footstep orientations from ω, whose cost function is replaced by

$$\sum_{j=1}^{F}(\theta_f^j - \theta_f^{j-1} - \int_{t_s^{j-1}}^{t_s^j} \omega(\tau)d\tau)^2 + k_{\text{obs}} \frac{w(\theta_{\text{obs}})}{d^2} \left(\theta_f^j - \theta_{\text{avo}}\right)^2$$

With respect to the original cost function, this contains an additional term that induces an alignment of the foosteps with θ_{avo}, i.e., with the tangent half-line originating at the closest object point (see Fig. 7). This second term is modulated through a scaling factor k_{obs} by the inverse of the squared distance and by the weight function $w(\theta_{\text{obs}})$ defined in Fig. 8. The idea here is that when the robot moves forward only obstacles lying in its front half-plane should be considered; whereas when moving backwards (e.g., during an evasion maneuver) footstep adaptation is only required to avoid obstacles behind the robot.

The second modification in the footstep planner is in the QP problem used to compute the footstep positions, to which a collision avoidance constraint is added. With reference again to Fig. 7, consider the point $B = (x_B, y_B)$ located along the line connecting the CoM with the closest object point, at a safety distance λ from the latter, and draw the normal to the same line through B. The half-plane beyond this line (in yellow in Fig. 7) is a forbidden zone[9] for the footstep locations. This constraint is easily written as

$$\mathbf{n}_{\text{obs}}^{\mathbf{T}} \left\{ \begin{pmatrix} x_{\text{f}}^j \\ y_{\text{f}}^j \end{pmatrix} - \begin{pmatrix} x_B \\ y_B \end{pmatrix} \right\} \geq \mathbf{0} \tag{8}$$

with $\mathbf{n}_{\text{obs}}$ the unit vector defined in Fig. 7.

Constraint (8) must obviously be enforced also in the QP problem of IS-MPC which will determine the final footstep positions.

Wrapping up, we may say that *adapt_footsteps* takes into account the presence of unexpected stationary objects in the robot path at two levels: in the cost function of the footstep orientation QP, and through the introduction of a collision avoidance constraint in the footstep position QP as well as in IS-MPC. Results in a variety of environments prove that this strategy is effective for collision-free locomotion (De Simone et al. 2017). As shown in the next two sections, this will be confirmed by both our simulations and experiments.

9.8 scale_velocity-force

If a moving obstacle enters $\mathcal{S}^{\text{scale}}$ while the robot is in the *Manipulation* context, the *scale_velocity-force* behavior is activated. Both the hand velocity and the interaction forces

[9] Turning the obstacle into a half-plane is necessary to convexify the obstacle avoidance constraint. This might appear to be a conservative choice, but it should be noted that the position of the half-plane is updated at each iteration, so the robot will eventually be able to go around the obstacle.

are reduced for enhancing the level of safety. Studies are available in which velocity/force bounds are derived taking into account the dynamic properties of the robot as well as the possibility of human injury (Haddadin and Croft 2016).

10 Simulations

In this section we provide simulated demonstrations of the proposed safety framework. The used robot is HRP-4, a 1.5 m tall humanoid with 34 degrees of freedom by Kawada Robotics. The robot has been equipped with a depth camera for gathering range and bearing information about the obstacles. All simulations are performed in the V-REP environment, enabling dynamic simulation via the Newton Dynamics engine.

The first simulation is designed so as to bring up several safety concerns in sequence, with the objective of illustrating how the state machine orchestrates transitions between behaviors. Snapshots of the simulation are shown in Fig. 9 (see the accompanying video for a movie clip). Only three of the safety areas defined in Sect. 6.2 are shown around the robot, i.e., $\mathcal{S}^{\text{track}}$, $\mathcal{S}^{\text{evade}}$ and $\mathcal{S}^{\text{adapt}}$, respectively with $d^{\text{track}} = 5$ m, $d^{\text{evade}} = 3$ m, and $d^{\text{adapt}} = 1.5$ m. The remaining areas $\mathcal{S}^{\text{scale}}$ and $\mathcal{S}^{\text{halt}}$ are not shown because not relevant.

At the beginning, the robot is standing, not performing any task, and scanning the environment (state Idle/*scan*). At $t = 9$ s, a human enters $\mathcal{S}^{\text{track}}$. This event triggers the *track* behavior, and the robot starts following the human with its camera (state Idle/*track*). At $t = 25$ s, the human enters $\mathcal{S}^{\text{evade}}$, triggering the *evade* behavior: the robot initiates an evasion maneuver while still tracking the human (state Locomotion/*track*/*evade*). At $t = 29$ s, while the robot is still performing the evasion maneuver, a stationary object (the yellow cylinder) enters $\mathcal{S}^{\text{adapt}}$; the *adapt_footsteps* behavior is activated and the footstep plan is modified to avoid collision (state Locomotion/*track*/*evade*/*adapt_footsteps*). When the human leaves $\mathcal{S}^{\text{evade}}$, at $t = 37$ s, the *stop* behavior is invoked to interrupt the evasion maneuver (state Locomotion/*track*/*stop*); motion is terminated at $t = 40$ s (state Idle/*track*). The robot quits tracking the human when he leaves $\mathcal{S}^{\text{track}}$ at $t = 44$ s (state Idle/*scan*).

At $t = 49$ s the second part of the simulation begins. The robot is commanded to reach a goal in the workspace (bulls-eye mark). Accordingly, the context switches to *Locomotion* and appropriate high-level reference velocities are sent to the gait generator (state Locomotion/*scan*). Since the yellow object is still in $\mathcal{S}^{\text{adapt}}$, the *adapt_footsteps* behavior is immediately activated (state Locomotion/*scan*/*adapt_footsteps*). Once the object goes outside $\mathcal{S}^{\text{adapt}}$, at $t = 62$ s, the robot can walk directly towards the goal (state Locomotion/*scan*). At $t = 83$ s, another stationary object (the green cuboid) enters $\mathcal{S}^{\text{adapt}}$, and again collision is avoided by footstep adaptation (state Locomotion/*scan*/*adapt_footsteps*). As soon as the green object leaves $\mathcal{S}^{\text{adapt}}$ ($t = 104$ s), the robot resumes normal walking (state Locomotion/*scan*) until it reaches the desired goal, where it stops (state Idle/*scan*). Note that the final stop is not the result of a safety behavior; rather, it is produced directly by the gait generator in response to the high-level references velocities vanishing at the goal.

The second simulation, shown in Figs. 10 and 11, is aimed at highlighting the difference between the *halt* and *stop* behaviors (see Sects. 9.1 and 9.3 , respectively). The robot is walking normally (state Locomotion/*scan*), with a reference sagittal velocity $v_x = 0.4$ m/s, when the *halt* and *stop* behaviors are respectively triggered during a double support phase. The arrest time used by *stop* is 2 s. As expected, the results indicate that with *halt* the robot stops immediately, almost bouncing back; whereas a much smoother finish is obtained using *stop*. Again, a clip of the simulation is included in the accompanying video.

11 Experiments

Experiments were simply designed to showcase different safety behaviors on an actual humanoid platform. Indeed, we were more interested in a ‘proof of concept’ rather than a quantitative performance evaluation, also considering the fact that the results will be in any case dependent on the specific platform.

In particular, we implemented the proposed approach on NAO, a 58 cm tall humanoid robot with 23 degrees of freedom by SoftBank Robotics. An Asus Xtion PRO Live camera has been mounted over the robot head for measuring depth. Both the camera motion generator and the gait generator run in real-time on the on-board CPU at a control frequency of 100 Hz.

In the first experiment, shown in Fig. 12 and the accompanying video, we use another NAO controlled through a gamepad as an unexpected moving object that pursues our robot. At the beginning, the robot is standing, not performing any task, and scanning the environment (state Idle/*scan*). At $t = 4$ s, the pursuer enters $\mathcal{S}^{\text{track}}$ (we set $d^{\text{track}} = 1$ m). This triggers the *track* behavior (state Idle/*track*). At $t = 7$ s, the pursuer enters $\mathcal{S}^{\text{evade}}$ ($d^{\text{evade}} = 0.6$ m) and the *evade* behavior is activated (state Locomotion/*track*/*evade*). When the pursuer leaves $\mathcal{S}^{\text{evade}}$, at $t = 16$ s, the *stop* behavior is invoked to interrupt the evasion maneuver (state Locomotion/*track*/*stop*); motion is terminated at $t = 17$ s (state Idle/*track*). The robot quits tracking the pursuer when it leaves $\mathcal{S}^{\text{track}}$ at $t = 20$ s (state Idle/*scan*). The experiment includes a second part, only shown in the video, where the pursuer doubles back and approaches our robot again, triggering another evasion maneuver.

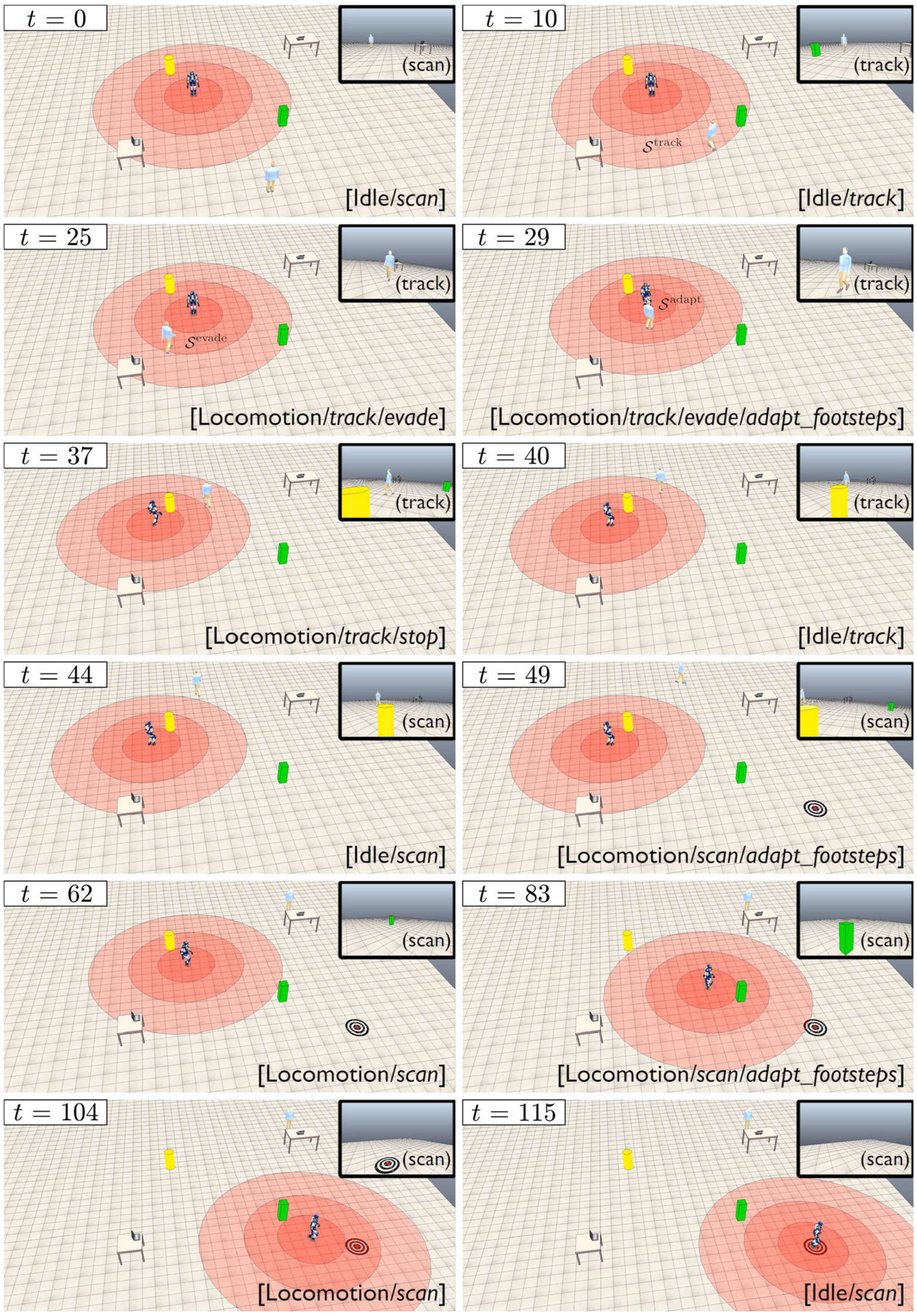

Fig. 9 Simulation 1: A scenario leading to several safety behaviors being activated in sequence. Snapshots correspond to transitions between states. The humanoid camera view is shown in the upper right corner. Clip in the accompanying video

The second experiment, shown in Fig. 13 and the accompanying video, focuses on the *adapt_footsteps* behavior. The robot is following a reference sagittal velocity $v_x = 0.08$ m/s (state Locomotion/*scan*) when, at $t = 6$ s an unexpected stationary object (wooden panel) enters $\mathcal{S}^{\text{adapt}}$ (we set $d^{\text{adapt}} = 0.9$ m). Successful obstacle avoidance is produced by footstep adaptation (state Locomotion/*scan*/*adapt_footsteps*), which is deactivated at $t = 9$ s when the obstacle is no more perceived (state Locomotion/*scan*). Note that the panel on the left flank does not trigger *adapt_footsteps* because it never enters $\mathcal{S}^{\text{adapt}}$. The robot then resumes walking in the desired direction.

Although we just reported results from two experiments, a similar successful performance was consistently achieved in our trials, also thanks to the reliability of the underlying MPC-based controller. Clearly, such performance is possible as long as the necessary sensory information is made avail-

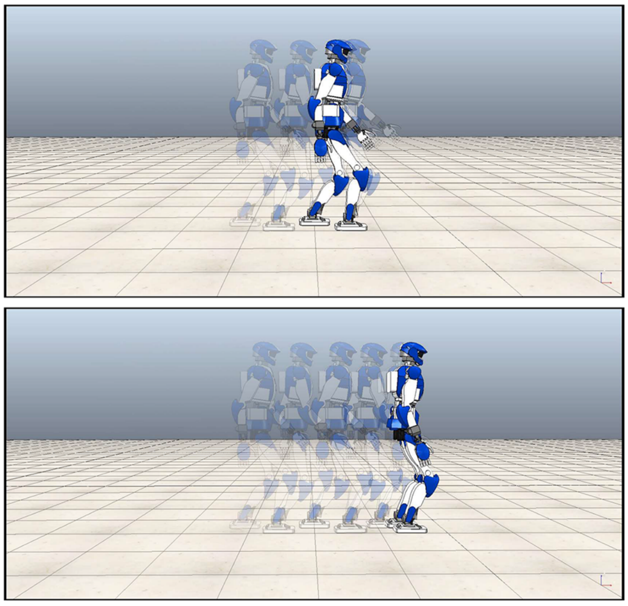

Fig. 10 Simulation 2: Stroboscopic motion of the robot using *halt* (top) vs *stop* (bottom). Clip in the accompanying video

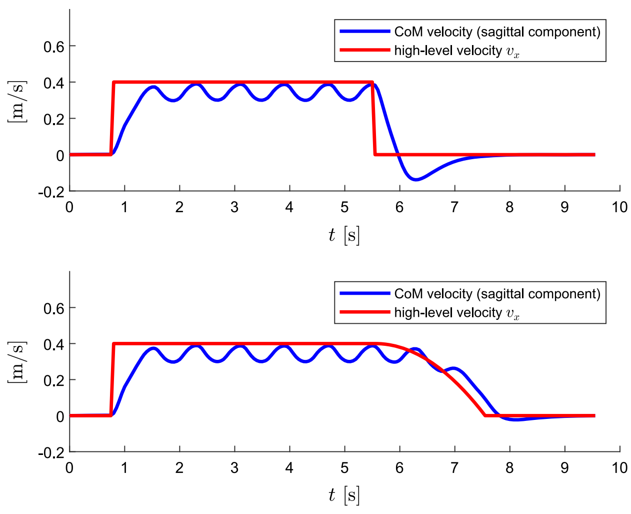

Fig. 11 Simulation 2: Velocity profiles with *halt* (top) and *stop* (bottom)

able to the robot; in this sense, robust perception strategies are an essential prerequisite for safety.

12 Discussion

The objective of this section is to provide some additional analysis and details about the proposed method.

12.1 Effect of safety on performance

The simulation and experimental results of the last two sections show that the proposed framework effectively increases the overall level of safety, ultimately protecting the robot as well as its co-workers. Obviously, this safety improvement will come at a cost, i.e., a deterioration of performance (in terms of, e.g., time needed to complete a task) due to the more cautious attitude of the robot.

To evaluate the above aspect in detail, we have performed a campaign of simulations focusing on a scenario where an HRP-4 humanoid must execute a walk-to locomotion task in a 25×25 m area. A variable number $(1, 3, 5$ or $10)$ of humans walking at 0.2 m/s cross at random the path of the robot. To increase the robot's chances of detecting and avoiding the humans, an omnidirectional camera has been added to its sensory equipment. As a consequence (see Sect. 12.3), the safety behaviors involved in the simulations are *stop*, *evade* and *halt*, for which we have used the following parameters: $d^{\text{stop}} = d^{\text{evade}} = 3$ m, $d^{\text{halt}} = 1$ m, $\bar{v} = -0.3$ m/s and $k = 0.2$. A simulation is stopped and a failure is recorded when the *halt* behavior is triggered, leading the robot to an Error state. Table 1 summarizes the outcome of 10 runs for each scenario, in terms of success rate (how many times the robot was able to complete the task) and completion time (minimum, maximum and average). For comparison, each simulation was also performed without the safety framework, adding however to the control architecture a standard obstacle avoidance module based on artificial potentials (De Luca and Oriolo 1994); in this case, failure means that collision with a human could not be avoided.

The results in Table 1 confirm that our safety framework allows the robot to complete the task in the large majority of cases, even in the presence of many moving humans. As a counterpart, there is a limited increase in the average time needed to complete the task (around 44% going from 1 to 10 humans). Note that the success rate is much lower in the absence of the framework, due to collisions between the robot and the humans. Video clips from a couple of trials with 5 and 10 humans are included in the accompanying video to illustrate the activation of the safety behaviors in this setting.

12.2 Limitations of the method

While the results presented so far are clearly positive, one must acknowledge that there are limitations to the proposed method.

1. Robot contexts (Sect. 6.1) are separated. For example, the possibility that the humanoid performs a manipulation task while walking is not considered here. Our motivation for excluding such cases is twofold: on the one hand,

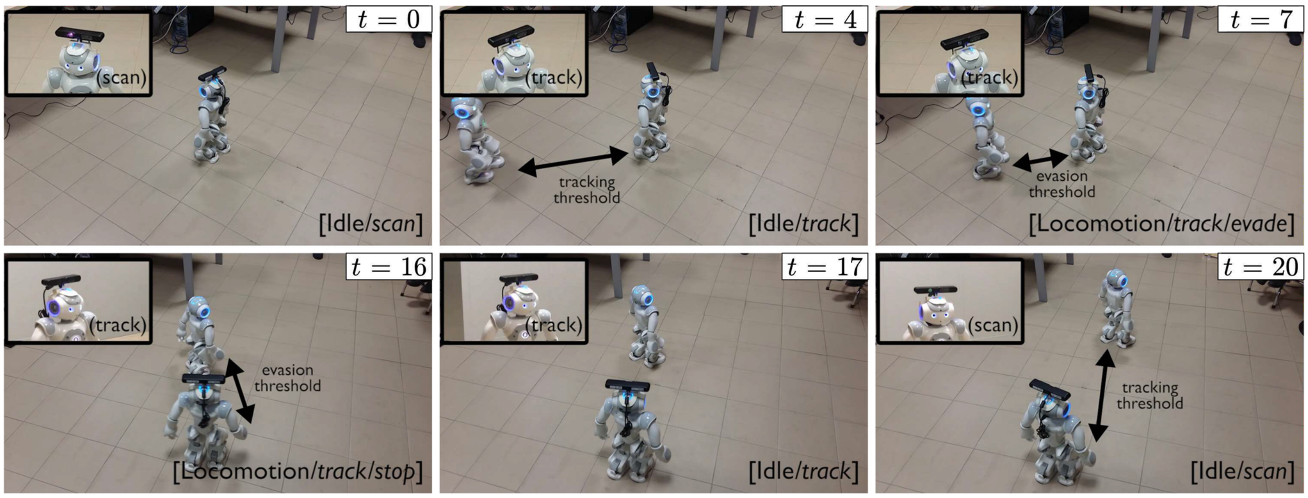

Fig. 12 Experiment 1: A NAO robot executing an evasion maneuver triggered by another NAO used as a moving object. Snapshots correspond to transitions between states. A close-up of the robot head is shown in the upper left corner. Clip in the accompanying video

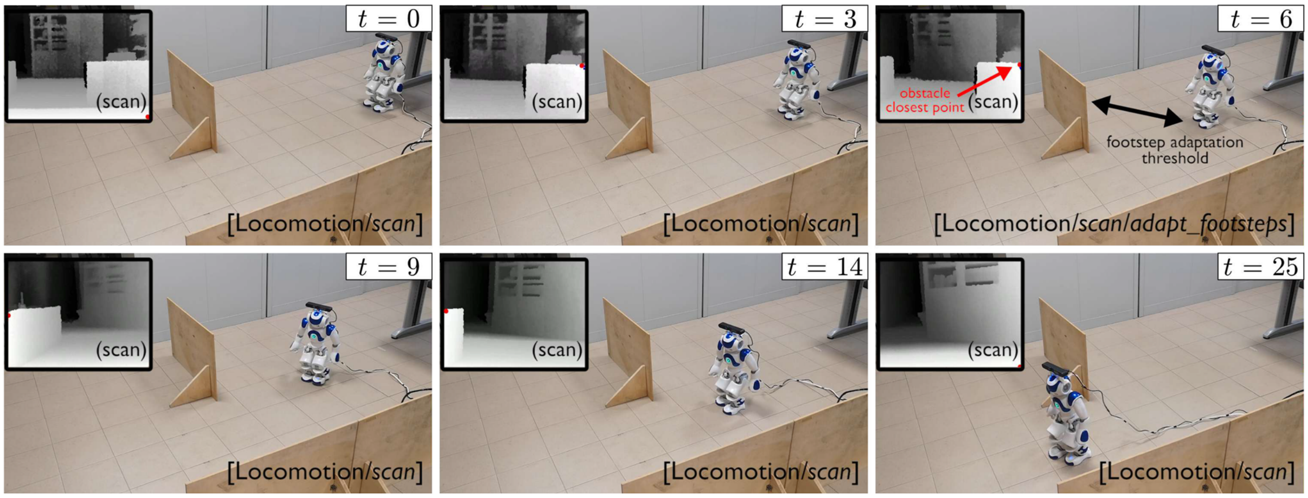

Fig. 13 Experiment 2: A NAO robot avoiding an unexpected obstacle through footstep adaptation. The humanoid camera view is shown in the upper right corner. Clip in the accompanying video

Table 1 HRP-4 executing a walk-to locomotion task in the presence of a variable number of humans: performance data over 10 runs for each scenario. Illustrative clips in the accompanying video

Number of humans	Success rate (%)	Minimum time	Maximum time	Average time	Success rate without safety framework (%)
1	100	84.75	111.22	89.64	90%
3	100	84.75	143.23	113.1	70%
5	100	100.20	148.10	113.59	70%
10	90	103.95	163.75	128.37	60%

we are assuming that sensory resources are limited, so that it may be impossible to adequately monitor both the manipulation and the walking area; on the other hand, it is rather obvious that focusing on one task at a time allows to maximize safety. However, an extension allowing execution of simultaneous tasks should be in principle relatively easy to design: one can simply add the combined contexts (e.g., *Loco-manipulation*) to the list and adapt the definition of behaviors to the new contexts.

2. We are looking at the safety problem from the viewpoint of a single robot. If multiple humanoids, all equipped with the proposed framework, are sharing the same environment, each of them will see the others as unexpected moving obstacles, and perform evasion maneuvers whenever required. While this may not be necessarily optimal, it should be considered that a proper multi-robot safety framework would inevitably require some degree of centralization and inter-robot communication, which may negatively affect the reactiveness of the single robot and the robustness of the safety framework. In any case, such a study is out of the scope of this paper.
3. We are looking at the safety problem in a context of pure coexistence between robots and humans, in the sense that physical collaboration between them is not allowed. This simplifying assumption may be however appropriate for many current applications, especially in industrial contexts where current regulations *de facto* exclude human-humanoid collaboration (Kheddar et al. 2019). However, there is no doubt that in the future such possibility should and will be allowed, making the design of safety frameworks considerably more challenging. Still, we believe that the structure of the proposed approach, based on the definition of override/temporary override/proactive behaviors orchestrated by a state machine, provides a valid template for such extension.

12.3 Adaptations

Although the safety guidelines proposed in Sect. 3 are completely general, our safety framework is in part dependent on the specific equipment of the humanoid, because safety behaviors were designed based on the sensing assumptions of Sect. 4. While this is inevitable, adapting the method to the availability of different measurements is relatively simple.

For example, consider the case in which the humanoid is equipped with an omnidirectional camera, so that the perception area $\mathcal{P}$ becomes a full circle. As a consequence, the robot does not need to direct its gaze, and it becomes possible to observe a moving object while, e.g., scanning the walking area or performing another observation task. This means that *scan* and *track* actually become a single behavior that is always kept active.

Another interesting situation is when the robot can measure relative velocity (direction and magnitude) of moving obstacles with respect to itself. In this case, the trigger of the *evade* behavior may be modified to allow activation only when the moving obstacle is directed 'towards' (in a quantitative sense to be suitably defined) the robot. Allowing an object to cross the robot safety area or not depending on its relative velocity may improve performance (some useless evasion may be avoided) but will obviously increase the level of risk (if the object is a human who brusquely changes direction, there may be no sufficient time left to perform the evasion); a reasonable trade-off between these two aspects must then be found.

Similar adaptations can be derived for other possible variations in the sensory equipment.

Finally, note that the framework description up to Sect. 7 (i.e., including the state machine) is independent from the control architecture of the robot. Clearly, any implementation of the framework must take into account (and conform to) such architecture; hence, to offer a worked out example we have first described a possible control architecture based on MPC (Sect. 8) and then discussed an implementation inside it (Sect. 9). Implementing the framework in a different control architecture, however, does not pose any conceptual difficulty.

12.4 Choice of parameters

A practically relevant issue in our framework is the choice of the various parameters, such as the radiuses of the safety areas. Most of these choices can be made on the basis of simple reasoning.

As an example, we discuss below a possible way to determine the threshold d^{evade}, which defines the $\mathcal{S}^{\text{evade}}$ area, based on the desired minimal distance between the robot and a moving obstacle. The worst case to be considered for this scenario is the one in which an obstacle moving at constant speed is heading towards the humanoid along the robot's sagittal axis. When the obstacle enters $\mathcal{S}^{\text{evade}}$, the robot starts an evasion maneuver under the control (5)–(7), hence moving along an arc of circle of radius $R = \bar{v}/k$. Assuming that the obstacle moves at the same speed $\bar{v}$ of the humanoid, a simple computation shows that the minimum distance between the robot and the obstacles takes the value

$$d_{\min} = R\sqrt{2\left(1 - \cos\frac{d^{\text{evade}}}{R}\right)}.$$

This relationship can be used for selecting a value of d^{evade} that guarantees a desired $d_{\min}$. For illustration, in Fig. 14 we have plotted $d_{\min}$ as a function of d^{evade} for $\bar{v} = 1$ m/s and $k = 0.75$, corresponding to $R = 4/3$ m. To achieve, say,

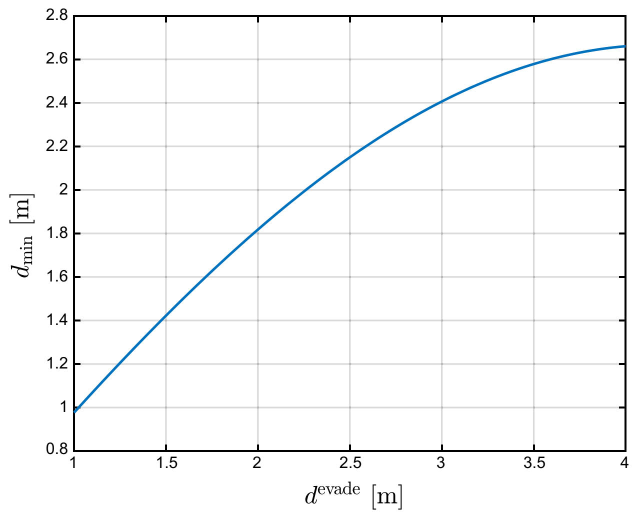

Fig. 14 Minimum robot-obstacle distance $d_{\min}$ as a function of d^{evade}

$d_{\min} = 2.4$ m one should choose $d^{\text{evade}} = 3$ m (as in our simulations).

13 Conclusions

In this paper we have presented a complete framework for the safe deployment of humanoid robots in environments containing humans. This is obtained through the definition of safety behaviors which are differentiated in override, temporary override and proactive. A state machine handles activation/deactivation of these behaviors based on the information given by the robot sensory system. In the description of the implementation, we focused on locomotion since it is the main aspect which distinguishes humanoids. An MPC setting has been used for realizing all locomotion-related behaviors efficiently. Effectiveness of the proposed method has been shown in dynamic simulation on the HRP-4 humanoid and through experiments on a NAO robot.

This work can be improved under several aspects. The main challenge we will consider in the future is going beyond human-robot coexistence to allow physical collaboration between the robot and humans. This obviously raises additional safety problems that can in principle be addressed by properly extending the proposed framework, provided that a communication system has been established between the human and the robot, e.g, using gestures and/or voice commands.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10514-021-09978-5.

Acknowledgements This work was supported by the EU H2020 project COMANOID.

Funding Open Access funding enabled and organized by Projekt DEAL.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Aboudonia, A., Scianca, N., de Simone, D., Lanari, L., & Oriolo, G. (2017). Humanoid gait generation for walk-to locomotion using single-stage MPC. In *17th IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 178–183.

Atkeson, C. G., Benzun, P. B., Banerjee, N., Berenson, D., Bove, C. P., Cui, X., DeDonato, M., Du, R., Feng, S., & Franklin, P., et al. (2018). What happened at the DARPA Robotics Challenge finals. In *The DARPA Robotics Challenge Finals: Humanoid Robots to the Rescue*, Springer, pp. 667–684.

Baudouin, L., Perrin, N., Moulard, T., Lamiraux, F., Stasse, O., & Yoshida, E. (2011). Real-time replanning using 3D environment for humanoid robot. In *11th IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 584–589.

Bicchi, A., & Tonietti, G. (2004). Fast and "soft-arm" tactics. *IEEE Robotics and Automation Magazine*, *11*(2), 22–33.

Bohorquez, N., Sherikov, A., Dimitrov, D., & Wieber, P. B. (2016). Safe navigation strategies for a biped robot walking in a crowd. In *16th IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 379–386.

Bouyarmane, K., Caron, S., Escande, A., & Kheddar, A. (2019). Multi-contact motion planning and control. In A. Goswami & P. Vadakkepat (Eds.), *Humanoid Robotics: A Reference* (pp. 1–42). Netherlands, Dordrecht: Springer.

Braghin, F., Henze, B., & Roa, M. (2019). Optimal trajectory for active safe falls in humanoid robots. In *19th IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 305–312.

Caron, S., Pham, Q. C., & Nakamura, Y. (2017). ZMP support areas for multi-contact mobility under frictional constraints. *IEEE Transactions on Robotics*, *33*(1), 67–80.

Chaumette, F., & Hutchinson, S. (2006). Visual servo control: I basic approaches. *IEEE Robotics & Automation Magazine*, *13*(4), 82–90.

Cognetti, M., De Simone, D., Patota, F., Scianca, N., Lanari, L., & Oriolo, G. (2017). Real-time pursuit-evasion with humanoid robots. In *2017 IEEE International Conference on Robotics and Automation*, pp. 4090–4095.

De Luca, A., & Flacco, F. (2012). Integrated control for pHRI: Collision avoidance, detection, reaction and collaboration. In *2012 IEEE RAS & EMBS international conference on biomedical robotics and biomechatronics*, pp. 288–295.

De Luca, A., & Oriolo, G. (1994). Local incremental planning for nonholonomic mobile robots. In *1994 IEEE International Conference on Robotics and Automation*, vol 1, pp. 104–110.

De Santis, A., Siciliano, B., De Luca, A., & Bicchi, A. (2008). An atlas of physical human-robot interaction. *Mechanism and Machine Theory*, *43*(3), 253–270.

De Simone, D., Scianca, N., Ferrari, P., Lanari, L., & Oriolo, G. (2017). MPC-based humanoid pursuit-evasion in the presence of obstacles. In *2017 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5245–5250.

Ferrari, P., Cognetti, M., & Oriolo, G. (2019). Sensor-based whole-body planning/replanning for humanoid robots. In *19th IEEE-RAS International Conference on Humanoid Robots*, pp. 511–517.

Flacco, F., Paolillo, A., & Kheddar, A. (2016). Residual-based contacts estimation for humanoid robots. In *16th IEEE-RAS International Conference on Humanoid Robots*, pp. 409–415.

Fujiwara, K., Kanehiro, F., Kajita, S., Yokoi, K., Saito, H., Harada, K., Kaneko, K., & Hirukawa, H. (2003). The first human-size humanoid that can fall over safely and stand-up again. In *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol 2, pp. 1920–1926.

Fujiwara, K., Kanehiro, F., Kajita, S., & Hirukawa, H. (2004). Safe knee landing of a human-size humanoid robot while falling forward. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol 1, pp. 503–508.

Haddadin, S., & Croft, E. (2016). Physical human–robot interaction. In *Springer Handbook of Robotics*, Springer, pp. 1835–1874.

Hall, E. T. (1966). *The hidden dimension*. Garden City, N.Y.: Doubleday.

Harada, K., Kajita, S., Kaneko, K., & Hirukawa, H. (2006). Dynamics and balance of a humanoid robot during manipulation tasks. *IEEE Transactions on Robotics*, *22*(3), 568–575.

Kajita, S., Hirukawa, H., Harada, K., & Yokoi, K. (2014). *Introduction to Humanoid Robotics*. Berlin: Springer.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *1985 IEEE International Conference on Robotics and Automation*, vol 2, pp. 500–505.

Kheddar, A., Caron, S., Gergondet, P., Comport, A., Tanguy, A., Ott, C., et al. (2019). Humanoid robots in aircraft manufactoring - the Airbus use-case. *IEEE Robotics and Automation Magazine*, *26*(4), 30–45.

Krotkov, E., Hackett, D., Jackel, L., Perschbacher, M., Pippine, J., Strauss, J., et al. (2017). The DARPA robotics challenge finals: Results and perspectives. *Journal of Field Robotics*, *34*(2), 229–240.

Kruse, T., Pandey, A. K., Alami, R., & Kirsch, A. (2013). Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, *61*(12), 1726–1743.

Lacevic, B., Rocco, P., & Zanchettin, A. (2013). Safety assessment and control of robotic manipulators using danger field. *IEEE Transactions on Robotics*, *29*(5), 1257–1270.

Lengagne, S., Vaillant, J., Yoshida, E., & Kheddar, A. (2013). Generation of whole-body optimal dynamic multi-contact motions. *The International Journal of Robotics Research*, *32*(9–10), 1104–1119.

Lim, J., Lee, I., Shim, I., Jung, H., Joe, H. M., Bae, H., et al. (2017). Robot system of DRC-HUBO+ and control strategy of team KAIST in DARPA Robotics Challenge finals. *Journal of Field Robotics*, *34*(4), 802–829.

Mandery, C., Borràs, J., Jöchner, M., & Asfour, T. (2015). Analyzing whole-body pose transitions in multi-contact motions. In *15th IEEE-RAS International Conference on Humanoid Robots*, pp. 1020–1027.

Marion, P., Fallon, M., Deits, R., Valenzuela, A., Pérez D'Arpino, C., Izatt, G., et al. (2017). Director: A user interface designed for robot operation with shared autonomy. *Journal of Field Robotics*, *34*(2), 262–280.

Michel, P., Chestnutt, J., Kuffner, J., & Kanade, T. (2005). Vision-guided humanoid footstep planning for dynamic environments. In *2005 IEEE-RAS international conference on humanoid robots*, pp. 13–18.

Minguez, J., Lamiraux, F., & Laumond, J. P. (2016). Motion planning and obstacle avoidance. *Springer Handbook of Robotics* (pp. 1177–1201). Springer.

Mombaur, K., Truong, A., & Laumond, J. P. (2010). From human to humanoid locomotion: An inverse optimal control approach. *Autonomous Robots*, *28*, 369–383.

Nagarajan, U., & Goswami, A. (2010). Generalized direction changing fall control of humanoid robots among multiple objects. In *2010 IEEE International Conference on Robotics and Automation*, pp. 3316–3322.

Navarro, B., Fonte, A., Fraisse, P., Poisson, G., & Cherubini, A. (2018). In pursuit of safety: An open-source library for physical human-robot interaction. *IEEE Robotics & Automation Magazine*, *25*(2), 39–50.

Naveau, M., Kudruss, M., Stasse, O., Kirches, C., Mombaur, K., & Souères, P. (2017). A reactive walking pattern generator based on nonlinear model predictive control. *IEEE Robotics and Automation Letters*, *2*(1), 10–17.

Ogata, K., Terada, K., & Kuniyoshi, Y. (2007). Falling motion control for humanoid robots while walking. In *7th IEEE-RAS international conference on humanoid robots*, pp. 306–311.

Radke, R. J., Andra, S., Al-Kofahi, O., & Roysam, B. (2005). Image change detection algorithms: A systematic survey. *IEEE Transactions on Image Processing*, *14*(3), 294–307.

Rios-Martinez, J., Spalanzani, A., & Laugier, C. (2014). From proxemics theory to socially-aware navigation: A survey. *International Journal of Social Robotics*, *7*, 137–153.

Samy, V., & Kheddar, A. (2015). Falls control using posture reshaping and active compliance. In *15th IEEE-RAS international conference on humanoid robots*, pp. 908–913.

Scianca, N., De Simone, D., Lanari, L., & Oriolo, G. (2020). MPC for humanoid gait generation: Stability and feasibility. *IEEE Transactions on Robotics*, *36*(4), 1171–1178.

Stark, M., Schiele, B., & Leonardis, A. (2016). Visual object class recognition. *Springer Handbook of Robotics* (pp. 825–840). Springer.

Tadele, T. S., de Vries, T., & Stramigioli, S. (2014). The safety of domestic robotics: A survey of various safety-related publications. *IEEE Robotics & Automation Magazine*, *21*(3), 134–142.

Takenaka, T., Matsumoto, T., & Yoshiike, T. (2009). Real time motion generation and control for biped robot -1st report: Walking gait pattern generation-. In *2009 IEEE/RSJ*, pp. 1084–1091.

Werner, A., Henze, B., Rodriguez, D. A., Gabaret, J., Porges, O., & Roa, M. A. (2016). Multi-contact planning and control for a torque-controlled humanoid robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5708–5715.

Wieber, P. B., Tedrake, R., & Kuindersma, S. (2016). Modeling and control of legged robots. In *Springer Handbook of Robotics*, Springer, pp. 1203–1234.

Yasin, A., Huang, Q., Yu, Z., Xu, Q., Syed, A. A. (2012). Stepping to recover: A 3D-LIPM based push recovery and fall management scheme for biped robots. In *2012 IEEE international conference on robotics and biomimetics (ROBIO)*, pp. 318–323.

Yun, S., Goswami, A., & Sakagami, Y. (2009). Safe fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping. In *2009 IEEE international conference on robotics and automation*, pp. 781–787.

Zamparelli, A., Scianca, N., Lanari, L., & Oriolo, G. (2018). Humanoid gait generation on uneven ground using intrinsically stable MPC. *IFAC-PapersOnLine*, *51*, 393–398.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.