Navigation and Manipulation Planning using a Visuo-haptic Sensor on a Mobile Platform

Nicolas Alt and Eckehard Steinbach
Institute for Media Technology, Technische Universität München, Germany

Abstract—Mobile systems interacting with objects in unstructured environments require both haptic and visual sensors to acquire sufficient scene knowledge for tasks such as navigation and manipulation. Typically, separate sensors and processing systems are used for the two modalities. We propose to acquire haptic and visual measurements simultaneously, providing naturally coherent data. For this, compression of a passive, deformable foam rod mounted on the actuator is measured visually by a low-cost camera, yielding a 1D stress function sampled along the contour of the rod. The same camera observes the nearby scene to detect objects and their reactions to manipulation. The system is passively compliant and the complexity of the sensor subsystems is reduced. Furthermore, we present an integrated approach for navigation and manipulation on mobile platforms which integrates haptic data from the sensor. A high-level planning graph represents both the structure of a visually acquired map, as well as manipulable obstacles. Paths within this graph represent high-level navigation and manipulation tasks, e.g. pushing of obstacles. A cost-optimal task plan is generated using standard pathfinding techniques. The approach is implemented and validated on a mobile robotic platform. Obtained forces are compared to a reference, showing high accuracy within the medium sensor range. A real-world experiment is presented which uses the sensor for haptic exploration of obstacles in an office environment. Substantially faster task plans can be found in cluttered scenes compared to purely visual navigation.

I. INTRODUCTION

Autonomous robotic systems typically use dedicated haptic or tactile sensors for interaction with objects. These sensors are required to determine exact contact points and forces when grasping or pushing an object. However, before an object is manipulated by a robot it is typically searched and tracked using a visual sensor, such as a camera or a laser scanner. Robots therefore often rely on two separate sets of sensors, based on the proximity of an object. This system design causes a number of problems: A handover point between the subsystems must be determined, ensuring coherency between haptic and visual perception. Incoherent measurements may lead to failures, such as incorrect grasps. Furthermore, system complexity and costs are higher due to the additional sensors – for example, haptic sensors require a lot of cabling if they cover larger surface areas of the robot. Finally, both the visual and haptic modalities have their specific shortcomings: Visual methods, for instance, often fail for transparent or specular objects, and cannot provide any information about weight or deformability of an object. Haptic sensors only provide sparse information about an object and require time-intensive exploration steps.

It is feasible to combine these two sets of sensors into a single visuo-haptic system. Cameras allow for remote sensing and are therefore essential for environment mapping, even on simple platforms. We propose also using visual sensors to measure tactile data and haptic events in the proximity of the robot, by attaching a deformable material to the robotic actuator. The material deformation is determined visually with high precision, and the acting force can be derived since the material characteristics are known. For low-end robots this approach provides visual and haptic information from one integrated system, which reduces costs by removing additional haptic sensors. More complex systems, on the other hand, profit from more accurate models of the environment obtained by fusion of visual and haptic data.

Our first contribution in this work is a visuo-haptic sensor made of plastic foam, which allows the measurement of contact forces and object shape along a line or curve. The mechanical part is completely passive, since it only consists of a foam rod, and it is naturally compliant. Forces applied to the foam result in a deformation, which is measured by a camera mounted above the rod. The sensor may take advantage of an existing camera and requires mounting of an inexpensive piece of deformable material to the robot or actuator. By detection of contours of the foam rod with visual snakes (see below), its deformation is measured in a dense fashion.
along its entire length. Several haptic properties are acquired from the obstacles by pushing them. These properties, such as friction force, deformation and the shape of the footprint, are referred to as haptic tags. Furthermore, the same camera tracks parts of the scene which are close to the manipulator using visual tracking. Specifically, approaching obstacles are detected, allowing the platform to slow down before contact. During contact the motion against the floor, as well as the reaction of the object, are determined using tracking. The setup is demonstrated on a small robotic platform used to explore a room, see Fig. 1 and also Fig. 9a.

Our second contribution is a method for navigation and manipulation planning which combines visually acquired maps from a Visual SLAM system (see below) with haptic knowledge about obstacles. For the common representation, a high-level navigation graph is first built from the occupancy grid map. Nodes in this graph correspond to high-level navigation decisions. Subsequently the graph is extended by nodes representing movable objects, which constitute a blockade in the map. Thus, these nodes stand for possible manipulation tasks – i.e. pushing or moving objects aside – which are planned using haptic tags. For instance, a paper bin in front of a doorway is detected as a blockade, which can be moved if access to the doorway is required. Using standard pathfinding methods, an integrated navigation and manipulation plan is generated. Compared to purely visual path planning, shorter paths involving manipulation operations are possible, and task plans can be generated to access parts of a room which are visually blocked.

The rest of the paper is structured as follows: In Sec. II the visuo-haptic sensor is introduced. Sec. III introduces the navigation and manipulation planner. Experiments and results from a demonstration system are presented in Sec. IV. This work is an extension of [1], where the sensor concept was originally introduced – i.e. foam tracking in Sec. II up to II-B, as well as experiment IV-B. The sensor concept is extended with scene tracking in Sec. II-C and haptic tags in II-D. Another extension is the planner in Sec. III, together with new experiments in IV-A and IV-C.

A. Related Work

1) Haptic sensors: Sensors which measure contact forces or material hardness have been proposed based on different principles – such as optical readout, pressure-sensitive resistors or piezo-electric transducers [2], [3], [4]. The contact shape determined with a 2D tactile sensor is used for object recognition in [5]. In [6], a single-point sensor mounted on a robotic arm is used to determine the geometry of an unknown scene.

Optical readout of tactile sensors with deformable materials is a commonly used principle: A sensor in the shape of a fingertip is proposed in [7]. The surface of its dome is made of an elastic material with dots drawn on the inside. These dots are used as visual features, tracked from the inside by a specialized high-speed camera. Contact forces are calculated from the displacement of the dots. In [8], an optical haptic sensor is presented that measures the traction field – i.e. magnitude and direction of forces – applied to a block of transparent silicon rubber. A camera looks to the inside of this transparent block and tracks markers which are embedded in the deformable material. The cameras and illumination sources used in these works are an integral part of the sensors and do not observe the environment. Therefore, these sensors, as a system component, are still purely haptic sensors; they do not provide the advantages of a visuo-haptic system.

2) Plastic foams: Mechanical properties of plastic foams as used in this work have been studied intensively, see e.g. [9]. The strain-stress relation for these materials is non-linear, typically showing three regions: A region of linear elasticity for very low strains, a plateau region showing high sensitivity to stress, and a region of densification when very high stress is applied, see also Fig. 2. Elastomeric materials – such as the widely-used polyurethane (PUR) foams – exhibit a monotonically increasing strain-stress relation [9, ch. 5]. This allows the stress (or force) to be uniquely determined from the observed strain (normalized deformation). Manufacturers typically guarantee certain production tolerances and measure the deformation of their materials at several points, according to the standard ISO 2439.

3) Visual tracking: In order to measure the deformation of the foam rod we determine the location of its contour in the image. Approaches based on image edge features have been used for a long time for objects which are well-described by their contour. Snakes [10] are one popular method of this type, which uses a model of several connected points moving in the image to minimize an energy term. This term consists of an image part attracting points to edge features, an internal part ensuring smoothness, and a part for external constraints, which limit the motion of the points and allows for initialization. The energy term is minimized by local search, allowing the tracking of edges as they move. If some edge points are very weak, the smoothness term ensures that the corresponding points are “dragged” by their neighbors. Initialization is done by a human or by another system which has some a priori knowledge of the estimated object position.

4) Mapping and Navigation: Visual SLAM systems based on laser scanners [11] are used on robots for self-localization within the scene and for simultaneously building/updating a 2D scene map. Most approaches use a probabilistic occupancy model for the grid cells (occupied/free or unknown), which is continuously updated when new measurements are available. In this way an entire floor of a building can be mapped by driving a robot platform around it, even though the coverage of a single sensor measurement is much smaller. Here we use the Karto SLAM system [12] with extensions for automatic loop closing [13]. In larger scenes with loops, maps would become inconsistent due to the accumulation of small pose errors. Loops are detected based on local scan matching and subsequent refinement of an internal pose graph. To reduce sensor costs, the laser scanner can be replaced with a single scan line of the Kinect depth sensor [14].

Maps can be directly used to perform navigation tasks, i.e. to find a collision-free path between two points. The state of each cell is set to occupied/free/unknown according to the highest probability. Costs for free cells are often assigned as discussed in [15], based on a distance map $m_d$ which
shows the proximity of the closest obstacle. A platform with a circular footprint of radius \( r_{rob} \) may not come closer to an occupied cell than \( r_{rob} \) – any cell which is closer to an obstacle is thus assigned infinite costs. (For non-circular footprints, see [16].) In proximity to an obstacle – e.g. for \( m_d \in [r_{rob}, 3r_{rob}] \) – the platform must drive slower and may require correction operations. Therefore, in these regions costs are (e.g. linearly) decreasing with increasing \( m_d \). Any point \( m_d > 3r_{rob} \) has constant costs, based on the maximum platform speed. For navigation each grid cell is represented as a node in a graph and connected to its neighbors with the above-calculated weights. Standard path planning algorithms such as \( A^* \) [17] may then be applied to this graph. Alternatively, fast-marching approaches [15], [18] may be used – they create a potential field with a single minimum at a pre-defined goal position. While the generation of this field is time-intensive in larger maps, this representation allows us to quickly find the cheapest path from any point in the map to the given goal. We use this approach to create region clusters in the map.

Direct navigation on maps is used for local planning, but unsuitable for large scenes. Global planning requires more abstract representations. For instance, [19] and [20] use a skeleton for navigation. Skeletons represent the thinned structure of objects and can be created from a binary map \( m := (m_d < r_{rob}) \) using the morphological “thinning” operation [21]. The skeleton exhibits a single path in the middle of each corridor or through each doorway, as well as intersections, which are placed into central areas. Furthermore, paths span into corners (stubs) and circle free-standing obstacles, showing that there are alternative ways to drive around them. The conversion of a skeleton to a topological graph is straightforward, by placing a node at each pixel and connecting it with its direct neighbors. Edge weights of this graph are taken from the cost map discussed above.

Navigation is much faster on such a simplified graph, yet it is not possible to reach the exact goal position. Modern path planning systems are thus often split into two parts [16], using a global planner which works on an abstracted representation of the scene and a local planner which utilizes a detailed local map and controls the platform in real-time. We use the graph as described as the basis for the proposed visuo-haptic graph, which is a global scene representation.

Abstract representations of scenes have also been created using other approaches – for instance, [22] proposes a method based on map segmentation inspired from image processing. The approach presented in [23] combines depth data and laser scans to recognize scenes, thus representing the environment at a semantic level.

II. Visuo-Haptic Sensor

The sensor consists of a deformable material, such as plastic foam, attached to a robotic platform which explores obstacles in the environment haptically, i.e. by driving towards and pushing them. Contact points and forces are determined by visually measuring the deformation of the foam, using the known deformation characteristics of the material. A camera is mounted about 20cm above the deformable material pointing almost vertically. It shows the top surface of the foam, the floor and scene in its direct vicinity as well as a part of the platform itself (see Fig. 1, 3). We use a consumer Full-HD USB camera with a diagonal field of view of \( \pm 40^\circ \) – those devices exhibit a good image quality and are cheap due to the large proliferation of similar devices in smartphones. The focus is fixed by software and set to the foam, so there is only a slight blur of the nearby scene.

Furthermore, the platform exhibits a laser scanner, which scans a range of \( 240^\circ \) in a plane parallel to the floor. Range data are used for self-localization and building of 2D maps with a Visual SLAM system, see Sec. I-A4. Additionally, an inertial sensor on the platform serves as a motion prior for the SLAM system. The obtained maps are used in Sec. III.

On our platform the deformable part is a passive foam rod (orange part in Fig. 1) which is roughly 25cm long (major axis) and exhibits a cross section of \( w \times h = 2 \times 1 \text{cm} \). A standard PUR (polyurethane) foam is used which costs only a few cents and can be easily replaced in case of attrition. The cross section is chosen based on the deformability of the material and the required force range. One of the long sides is attached to a rigid mounting plate, which may be straight or curved, and the opposing side comes into contact with objects.

The direction of exploration, denoted \( x \) in Fig. 1, corresponds to the dominant forward motion of the platform on the floor. Forces act along vectors \( d(s) \) and deform the foam rod along its width \( w \). Deformation is expressed as a scalar function \( \delta(s) \) of displacement [in m], which is densely sampled along the major axis \( s \). The normalized strain of the material is \( \delta(s) / w^2 \). Shear forces parallel to the mounting curve are negligible in this setup and are not measured.

Calibration is performed by pushing a large plate onto the foam rod with a KUKA robot arm. The exact position is known from the arm controller, while the applied force (and thus the pressure) is measured with a JR3 force sensor. The process is repeated multiple times, yielding the data points shown in Fig. 2. There are some relaxation effects in the material – yet, they are negligible at the range of manipulation velocities used by our system. As expected from literature, the data points form a curve with three different regions, see
discussion in Sec. I-A2. Since foam manufacturers provide material tolerances, calibration must only be performed once for a certain material type. Here, we mainly rely on the plateau region for normalized strains in $[0,1,0.5]$, which corresponds to a reasonable range of forces for the application at hand. Additionally, this region allows for the most accurate measurements, since the sensitivity of the material to force changes is largest. Note also that the curve exhibits a strong hysteresis effect, depending on whether forces are increased (red curve) or decreased (blue points). Therefore, we only measure the displacement while forces are increased. The characteristic curve is repeatable for multiple experiments.

There are several boundary effects to be considered: Local material deformability increases towards the edge of the foam. The foam deforms equally along its entire length when it presses against an obstacle – therefore, calibration is performed on the rod with the final cross-section. Thus, this effect is already taken into account for the edges along the major axis. However, stress discontinuities along the major axis require special consideration: During measurement, if the stress applied to the foam is a step function along $s$ (e.g. the edge of an object), the front contour deforms smoothly beyond the contact area, see also Fig. 7(d). The smoothing is attributed to internal tension in the foam that results in additional stress at the contact area. This additional stress is approximated by assuming that the smoothed contour is actually caused by an enlarged contact area. During calibration there are no such discontinuities, since the length of the rod is cut such that it fits under the plate.

A third-order polynomial $f$ is fitted to the points obtained from calibration, yielding a phenomenological model for the strain-stress relationship, which is depicted as a red curve in Fig. 2. The curve fits the datapoints well, except for the strain-stress relationship, which is depicted as a red curve from calibration, yielding a phenomenological model for the fits under the plate.

Edges are tracked using the well-known concept of snakes, see Sec. I-A3, which consist of connected points “snapping” to image edges. We track points along the contour of the foam spaced about 3mm apart, which allows for an accurate representation of possible deformations. After initialization (see below), snake points move within a certain local neighborhood to iteratively minimize an energy term which consists of several parts, see Eqn. (2): First of all, the negative amount of edge strength $E_e$ tries to keep the points on image edges. In order to avoid snake points snapping to strong internal edges within an object, $E_e$ is limited to the average edge strength $\bar{\tau}$ of all snake points. A smoothness term $E_s$ accounts for the fact that the slope of $\delta(s)$ is limited. Furthermore, there are no edges within the foam rod, i.e. in between the inner and outer contour. Therefore, an energy term $E_j$ penalizes points jumping over edges along the path from the closest point of the reference snake $p_k^r$ to the current point. Finally, $E_c$ constrains point motion to the vector of deformation $d$, which is perpendicular to the reference contour of the foam rod. As this is the major direction of deformation, snake points stay on the same physical point on the foam, and the sampling density remains constant. The total energy is evaluated and minimized in a local neighborhood $\xi = [x, y]^T$:

$$E(\xi) = w \cdot [E_e, E_s, E_j, E_c], \quad \text{with} \quad E_e = -\min(\|\nabla G \ast i\|, \bar{\tau}),$$

$$E_s = (p_{k-1} - 2\xi + p_{k+1})^2(p_{k-1} - 2\xi + p_{k+1}),$$

$$E_j = \frac{1}{\bar{\tau}} \int_{x=p_k^r}^{\xi} \|\nabla G \ast i\| \, dx,$$

$$E_c = \begin{cases} 0, & \text{if } (\xi - p_k^r)\|\mathcal{P}^{-1}\|d(s) \leq \infty, \quad \text{otherwise} \end{cases}$$

Where $i$ – image, $\nabla$ – gradient operator, $G$ – Gaussian blur operator for noise reduction, $\mathcal{P}$ – projection operator, see Eqn. (3). Weights $w$ are set such that energy terms are in $[-1,1]$ within the search space. A 1D constraint is imposed by $E_c$, so the search for the optimum is fast even for large neighborhoods. Processing at frame rate (30 Hz) poses no problem to a mid-range Intel i5 platform. Note that it is not feasible to integrate shape priors in the energy term, as in more recent work using snakes. The contour of the foam is solely determined by the shape of the obstacles, and the correlation of close-by values of $\delta(s)$ is accounted for by $E_c$.

The approximate position of the foam rod in the camera image is typically known from a geometric robot model. Otherwise, it may be located using markers. First, the inner snake is initialized by adding points iteratively at a constant distance and having them snap to edge pixels. Points $p_k^r$ on the inner snake serve as a reference position of the sensor.
base – which might change slightly due to movement of the mounting plate or the camera. Next, points of the outer snake are initialized slightly outside of the inner snake. To allow for varying rod widths, these points are pushed away from the inner snake by an additional energy term until they reach the stable outer edge. The idle state of the outer snake is used as the zero reference of displacement \( \delta \).

Pixel positions on the snake are converted to real-world coordinates using the intrinsic matrix of the camera \( K \) and a coordinate frame \((P_0, R)\) at the center of the foam rod spanned by the exploration vector \( x \) and with \( y \) parallel to the floor. The mounting curve \( s \) and deformation vectors \( d \) lie on the \( x - y \) plane of this frame. In a robotic system \((P_0, R)\) are determined from the extrinsic camera parameters and the geometric robot model. Otherwise, the pose can be determined by applying markers. In this manner, it is sufficient to obtain 2D information from the camera. The projection operator \( \mathcal{P} \) converts a point in the image \( p = (x, y, 1)^T \) [in pixel] via the camera-centered coordinate frame \( P(X, Y, Z) \) [in meters] to a point \( P^F(X', Y', 0) \) on the \( x - y \) plane of the reference frame with normal \( n = R_{\cdot3}^T \):

\[
\begin{align*}
P &= P_0 + p' n', \quad p' = K \cdot p \\
\mathcal{P} : p &\rightarrow P^F
\end{align*}
\]

Visual measurements are noisy due to image noise, shaking of the camera and location uncertainty of the image edge. For individual snake points, noise with \( \sigma_1 = 70 \mu m \) is observed in the static case. Additionally, edge locations may be biased by large illumination changes or strong intensity discontinuities on objects close to the sensor. Bias effects depend on the surrounding scene and change with a much lower frequency than image noise. Taking into account these effects, at worst an edge uncertainty of \( \sigma_2 = 0.5 \) mm is observed.

### B. Fitting of Geometric Primitives

In typical household or office environments, a robot encounters obstacles such as walls, boxes, table legs, cylindrical trash bins or vases. As these artificial objects are aligned perpendicular to the floor and exhibit symmetry, a 2D “footprint” is often sufficient for tasks such as mapping or navigation. In case of contact with the sensor, the impression in the foam corresponds to a partial 2D contour of the object. Typically, a connected set of deformed snake points (contact region) corresponds to a single object.

Two-dimensional primitives are fitted to points \( P^F \) within the contact region. They serve as a rough approximation of the shape of an object beyond the contact points. The following primitives, which represent typical objects, are used: Lines (for walls, large boxes), line segments (small boxes), corners (boxes, table legs) and circles (trash bins, bottles). Line-based primitives are fitted into the set of contact points using least-squares minimization. A line segment is used if the center points of the deformation lie on a line, but the end points do not. Corners are described by two lines which intersect at the point of maximal deformation. Circles are fitted using the algebraic approach described in [24].

![Visual point features are tracked on the object surface (yellow tracks), as well as on the floor to measure motion in the respective image areas. Here, the object is just falling, and the track is lost due to motion blur. Furthermore, foreground areas in the top detected by the Gaussian mixture model are depicted in red, see text.](image)

Each candidate primitive is tested on how well it fits to the observed points \( P^F \). A score is calculated, based on the mean squared distance between observed points and closest model points as well as the number of points \( n \) used by the model:

\[
\gamma = \exp \left( -\frac{1}{|D|} \sum_{k \in D} \frac{1}{\sigma_2^2} d_M^2(P_k^F) \right) \cdot \frac{n}{|P^F|}
\]

Where \( d_M \) – distance of a point from model \( M \), \( D \) – deformation set. The primitive with the highest score and \( \gamma > 0.5 \) is chosen. It may be used for the refinement of a visual map, or for purely haptic mapping [1].

### C. Tracking Scene Motion

Besides tracking the foam deformation, the camera observes the vicinity of the robot platform – specifically the floor, approaching obstacles and objects in contact with the sensor. While the platform is pushing an object, its ego-motion is determined by tracking the floor, complementing the inertial sensors. Also, the surface of the object is tracked to determine its reaction, see Sec. II-D. Specifically, rotations or unexpected sudden motion, such as falling over, are of relevance.

Approaching obstacles are detected in order to allow the platform to slow down before touching them. Detection is again performed in the same camera image to obtain coherent data from the direct proximity of the sensor. Obstacles can be detected based on visual appearance differences or based on their height – since they are closer to the camera than the floor, they move faster. As feature tracking of the floor is problematic at higher speeds due to motion blur, we chose the former approach. An efficient way to model visual appearance is by using Gaussian mixture models, which are trained based on a set of expected images and detect contradiction to the trained model. Here, the approach from [25] is used, which handles the training process automatically and gradually adapts the model to the scene. The detector is trained automatically and quickly adapts to a change in the environment by learning the new floor model within a few frames. Obstacles are searched
for only in the upper part of the camera image where they appear first, see Fig. 3.

During contact, tracking of the floor and the object surface above the foam is performed using a Kanade-Lucas-Tomasi (KLT) tracker [26]. This feature extractor finds corner-like structures in the image, which are well-localized and thus suitable for tracking. In order to ensure reliable motion estimates, only stable features are selected for tracking within the appropriate regions of the image using the scores from [27]. Features for floor tracking are selected within small patches left and right of the foam and a mask is created for the object based on the fitted primitive. Features are tracked over several frames and connected by local search, yielding a sparse motion field of the floor and the object, see feature tracks depicted in Fig. 3. Tracking works well on textured surfaces, but the quality depends on the appearance of the scene. Therefore, visual features are used to complement scene knowledge – if tracking fails, only the force measurements from the sensor are available. Typical failure cases are transparent or entirely textureless objects. Floor tracking is more reliable, since household floors are usually textured, and even plain concrete surfaces, as shown in Fig. 3, exhibit features from material wear or imperfections.

The motion model of the floor is a 2D translation and rotation of the platform on the (known) floor plane. Its parameters (speed, rotation) are estimated from the feature tracks to obtain a visual motion estimate, which complements the inertial unit of the platform. Some feature tracks will be incorrect (outliers) and must be removed. Motion parameters are estimated using RANSAC [28], which finds the majority vote of features and is thus robust to outliers and local failures. Projection of the motion onto the floor plane yields the real-world motion of the platform, coherent to the sensor readings. That way, fixed objects and deformable objects can be detected, see Sec. II-D.

The object surface is located in the image area just above the foam (see Fig. 3). After foam deformation has reached its maximum, ideally the object, and thus the feature points on it, no longer move relative to the camera. Otherwise the object movement and especially its rotation are estimated using the motion of the features. Feature points are projected onto an estimate of the object surface. It is modeled as a plane which is perpendicular to the floor and spanned by a line which approximates the deformation region. Since the exact contact points are known and due to the small relative object motion, errors from model deviations remain low. Object motion is determined by homography estimation and decomposition [29], [30] of feature tracks. Rotation on the floor plane $H_R$ is most relevant, see Sec. II-D. Fall of an object usually occurs into the direction of the push and could be detected by fast rotation around the respective axis. However, tracking will not be able to follow such fast motion, and instead a sudden loss of track is taken to trigger a fall event.

Obviously a number of additional techniques could be applied to acquire visual scene knowledge. A depth camera would allow for stable extraction of the floor surface using plane fitting, and the object geometry could be accurately acquired. However, coherence needs to be ensured, and sensors like the Kinect exhibit a minimum range of $0.5m$. Furthermore, tracking could be performed with multiple image patches to increase stability and to obtain a homographic projection for each patch. The proposed system renounces additional techniques and relies only on simple feature tracking, since this showed to be effective. This approach keeps the computational load low as compared to algorithms processing 3D point clouds.

D. Acquisition of Haptic Tags

Modeling detailed haptic properties of an object – such as friction, deformability or roughness – can be very complex, especially since properties generally change along the surface. While it is possible to explore an object haptically with a robot arm in a time-intense process [6], a robot platform as presented here does not have that capability. The robot requires a prediction of object behavior for planning simple and standardized manipulation tasks. For these tasks, a detailed haptic object model is not required – it is sufficient to know some global or semi-global object parameters. In case of a standardized grasp, these parameters could be weight as well as the local friction and deformability at the contact area. For pushing operations, as considered here, friction forces, object stability and potentially inertia are of greatest relevance. These values are partially specific to the actuator, the grasp point and the manipulation task at hand. We refer to these properties as haptic tags, and they are discussed in detail below.

Haptic tags are acquired using the visuo-haptic sensor presented in Sec. II while the platform slowly drives straight towards the object, pushes against it and moves it a bit, if possible. In addition to the force profile from the foam deformation, the camera acquires visual cues from the environment to determine the reaction of the object on the pushing force. The haptic tag consists of the following properties:

- Static friction force $H_F[N]$
- Dynamic friction $H_F^d[N]$
- Deceleration time $H_I[s]$
- Sideward drift $H_D[m/m]$ 
- Deformability $H_D$ 
- Rotation on the floor plane $H_R[[rad/m]]$
- Fitted geometric primitive, see Sec. II-B
- Events Fixed/Fall
- Counters for Exploration/Failure

These properties allow for the prediction of manipulation tasks and object properties: The manipulability is determined from the events “fixed/fall”, and the friction force is an estimate for the required effort. If the manipulator cannot control lateral object motion, reliability is estimated based on drift and rotation. The expected reaction of an object is expressed by its static friction, deformability, drift, rotation and the “fall” event. Furthermore, the deformability allows for an estimate of the material type, which is important, for instance, to adjust grasping parameters. Simple articulated objects – e.g. a wing of a door with joints – are modeled based on rotation and deceleration time. Here we use the haptic tag for planning of pushing operations, see Sec. III-B.

Acquisition is separated into different phases, triggered by events: Event contact – phase (a) – event motion – (b)
– platform stops – (c). If an event does not trigger, the corresponding phase is skipped, and some haptic properties remain undefined. The visuo-haptic sensor measures distances between the robot, object and the environment (floor), see Sec. II-C. As soon as the foam starts to deform, i.e. $\delta$ starts to increase, the event contact is triggered. Once the deformation remains constant (or decreases) while the platform continues to move, the event motion is triggered. Data are stored during exploration and later processed on a per-phase basis.

Phase (a) — The static friction is the maximum force value measured during this phase. Furthermore, the object deformation is determined by $H_D = 1 - \frac{\text{foam deformation}}{\text{platform motion}}$. For rigid objects, the two distances are equal and $H_D = 0$. Distance and pressure values are taken when they are largest, and a linear compression model is assumed, thus $H_D = \text{const}$. $H_D$ is a relative measure which may be converted to the compressibility $\beta$ when volume change and local pressure are considered. Objects with a negligible static friction force, such as open wings of doors or balls, trigger the motion event immediately, skipping this phase.

(b) — Dynamic friction, drift and rotation are acquired. The dynamic friction is the average force while the object is moving. Drift $H_S$ is the shift of the central contact point perpendicular to the pushing vector, normalized by the pushing length. Object rotations indicate instability during the manipulation and are estimated using feature points tracked on the surface, see Sec. II-C. Rotation on the floor plane is comparable to the drift, while rotation around the y-axis of the camera indicates that the object falls over.

(c) — For most household objects friction forces dominate and their inertia cannot be directly observed. In this case, after the platform stops the measured force remains constant, and $H_I = 0$. Only some objects show an observable deceleration behavior and continue their motion after stopping. The duration of continued motion is taken as an estimate for deceleration. Object motion is determined from the change of foam deformation and by feature tracking. For instance, the wing of a door may continue to move after pushing, detaching from the foam to finally stop somewhere in front of the platform due to small friction in the door hinge. Finally, the two counters are used for an estimate of the success probability, when the object is manipulated several times. A failure is triggered when fast feature motion is detected, e.g. when the object falls or is destroyed.

Haptic tags are assignable to different visual representations of the object: Most importantly, they are assigned to the footprint of the object in the map at the contact point of exploration. This means that haptic tags stick to objects as long as they are not moved, even when the robot is in a different part of the building. Furthermore, haptic tags can be associated to object identities in an object database used for visual search with feature descriptors, e.g. [31]. That way, if a known and explored object is recognized visually anywhere in the room, its haptic tag can be used for manipulation planning without further exploration. This idea can be extended such that haptic tags are loosely associated to visual classes of objects, such as “bottles”. When an object instance is detected, the haptic tag allows for an estimate of its manipulation properties.

III. VISUO-HAPTIC NAVIGATION AND MANIPULATION

In the following an abstract graph representation is presented which jointly models navigation and manipulation decisions. Sec. III-A discusses how high-level navigation nodes are generated from a visually obtained occupancy grid map. In Sec. III-B, we present our approach to complement this visual navigation graph with manipulation nodes. These nodes represent obstacles which can be moved to open new paths. On the presented system, manipulation is limited to pushing objects away. Haptic tags (Sec. II-D) are used to estimate manipulation parameters. A path in the extended visuo-haptic graph translates to both navigation and manipulation tasks, depending on the node types along it. The former tasks are executed by a standard local planner [16], while the latter are discussed in detail in Sec. III-C.

A. Global Navigation Planning

This section describes the processing of occupancy maps to obtain the abstract visual representation, which is used for the visuo-haptic graph. We make use of established techniques which are reviewed in Sec. I-A4. The map is acquired using the laser scanner on the mobile platform and the Karto Visual SLAM system. A graph-based representation is obtained using the following steps: (a) Determination of a trivalent state (occupied/free/unknown) for each cell; (b) generation of a distance map $m_d$ that shows the distance to the nearest occupied cell (green in Fig. 4a); (c) calculation of a cost map $m_c$ that penalizes proximity to obstacles, see below; (d) thinning of the reachable map area, i.e. $m_c < \infty$, yields a skeleton (red paths in Fig. 4a); (e) conversion of the skeleton to a graph with edge costs $d \cdot m_c$, where $d$ – edge length. The costs $m_c$ of cells represent the inverse of the possible speed of the platform and are infinite for unreachable cells:

$$m_c = \begin{cases} \infty & m_d < r_{rob} \\ m_d & m_d > 3r_{rob} \\ \left[v_{\text{max}}^{-1} \ldots v_{\text{min}}^{-1}\right] & r_{rob} \leq m_d \leq 3r_{rob} \end{cases}$$

The obtained topological graph depicts the structure of the room and is further simplified: Short stub paths which the skeleton introduces are removed. Nodes with exactly two edges are connection nodes which can be deleted, except in cases where their removal would also remove a loop. Corresponding edges are merged together, summing their costs. The remaining nodes are located on the intersection points of the skeleton, see Fig. 4b. For a better visualization we keep some connection nodes at regular intervals, such as 1m. The edge costs are still based on skeleton segments in the center of the room and are thus in general overestimated. The platform motion is fastest and most reliable along such paths, yet there will be shorter paths in most cases. For a global planner, however, this approximation is feasible. The semantic graph is a much more efficient representation than a grid map, since it only exhibits one node for each choice around obstacles.

A mapping between occupancy grid cells and nodes of the semantic graph (and vice versa) is required to map a path in the semantic graph to the grid map. The assignment should be based on the shortest path between grid cells and nodes.
It must take the cost map \( m_c \) into account and especially the fact that obstacles may block free space from a close-by node. Each node is considered a representative of its assigned map segment, as depicted in Fig. 4b. A fast-marching approach \([18]\) with multiple start positions is utilized. The start positions correspond to the node locations and represent multiple minima in the potential field \( m_d \). Furthermore, a segment map \( M_R \) is obtained, which assigns each grid cell to a node id:

1) Initialize potential \( M_P \leftarrow \infty \), region map \( M_R \leftarrow 0 \\
2) \ M_P \leftarrow 0, M_R \leftarrow \text{node} \rightarrow \text{id} \text{ for cells at node locations} \\
3) \ u \leftarrow \text{Cells where } M_P \text{ has been updated} \\
4) \ \forall u, \text{ neighbors } n \text{ of } u: \\
\quad \text{if } p' < M_P(n), p' = M_P(u) + m_c \cdot d(n, u): \\
\quad \quad M_P(n) \leftarrow p' \text{ and} \\
\quad \quad M_R(n) \leftarrow M_R(u), \\
5) \text{If } M_P \text{ has changed, iterate to step 3} \\

The potential map \( M_P \) also allows for fast shortest-path planning from any point in the map to the closest node, simply by iteratively moving to the neighbor cell with the lowest potential. This detailed path is used by the local planner to reach the final goal.

B. Integration of Haptic Knowledge

So far, the map for the semantic graph is acquired solely based on visual data – i.e. each obstacle is considered to definitely block robot movement, regardless of its weight and shape. Using haptic knowledge about obstacles – specifically whether an object is movable by the robot platform – new paths which involve manipulation of obstacles may be found in the map. For instance, a toy ball blocking a passageway can be easily pushed away, adding a new edge in the graph. In order to keep a consistent representation, the visual semantic graph is left unchanged and extended by additional nodes (“manipulation nodes”) representing movable objects.

For the following process, a list of explored objects with haptic tags is assumed to be known together with an assigned region in the map, see Fig. 5a. The list of haptic tags is acquired as discussed in Sec. IV-C, which presents an exploration plan for obstacles. Sec. II-D discusses the properties in the haptic tag. The visual semantic graph is extended to a visuo-haptic graph as follows:

1) Set the occupied grid cells representing obstacle \( o_i \) to “free” in the original visual map
2) Using the updated map, recalculate the cost map \( m_c \) in the vicinity of \( o_i \) according to Eqn. (5)
3) Add a node \( n_i \) to the semantic graph \( G \), located at the center of obstacle region \( o_i \)
4) In region map \( M_R \), assign all unassigned cells connected to \( n_i \) and with \( m_c < \infty \) to \( n_i \) (yellow in Fig. 5b)
5) Determine all neighbor regions of \( n_i \) in \( M_R \) and add edges between the corresponding nodes and \( n_i \)
6) Calculate edge costs by running \( A^* \) between the \( n_i \) and its neighbors using the updated cost map
7) Remove those edges again which do not significantly reduce any path costs

Costs in the semantic graph provide a time estimate for navigation operations. The costs \( t_{mani} \) for a manipulation operation are expressed consistently, taking into account the following components:

(a) Platform-specific constant setup/finishing time, 
(b) manipulability given the current plan, 
(c) duration of object motion \( t_{push} \) based on the required effort and push length, 
(d) penalty for possible manipulation failures \( t_{fail} \).

Costs are calculated based on the haptic tag. The manipulability (b) is a binary decision based on events “fixed/fall” and lateral motion. The later is estimated based on estimated drift \( (H_S \cdot \text{push length}) \) and must not exceed a certain threshold. Failure probabilities are derived from the exploration counter in the haptic tag if the object was explored several times. The pushing speed and length yield \( t_{push} \). Speed is adapted to the required effort determined by the friction force and depends
on the platform capabilities. Any path via a manipulation node includes two edges connected to this node, so costs $\frac{1}{2}t_{\text{mani}}$ are added to all its edges. The time for approaching and leaving the movable object is already expressed in the edge costs calculated in step 6.

A path in the extended graph $G^H$ between the current position and a goal position can be planned as usual, using $A^\ast$. Alternative paths can be determined based on task-specific side conditions, such as most information gain about the scene, most reliable path or avoidance of manipulation tasks. Each navigation node within a path represents a high-level navigation task, e.g. “go left/right around obstacle” or “enter hallway”. A corresponding initialization is given to the local planner, which searches a feasible and short path within the regions associated to the nodes in $\mathcal{M}_R$. Each manipulation node represents a task to push an object out of the way. The local planner for this task is described in Sec. III-C and attempts to move the obstacle such that it no longer blocks the robot. After the manipulation the scene map is updated, and $G^H$ must be adapted to that change by converting the manipulation node to a navigation node and removing costs $\frac{1}{2}t_{\text{mani}}$ from the associated edges. From time to time, when the robot rests, the full process for graph generation is restarted. Local planners may fail – especially for the manipulation tasks – necessitating replanning on the global level, with costs set to $\infty$ for edges associated to the failure.

The benefit of a new haptic node $n_i$ is determined by the maximum shortening of paths between any of the adjacent nodes. This value is used to determine the order of exploration, see Sec. IV-C:

$$\delta t = \max \{ \text{costs}_{G^H}(n_k, n_l) - \text{costs}_{G}(n_k, n_l) \} \quad \forall k, l | k, l \text{ neighbors of node } n_i$$ (6)

C. Manipulation planning

The task of the manipulation planner is to find solutions for object pushing, given an object and a map. Possible solutions are defined by the parameters contact point, pushing path and pushing target, which are determined here based on the object’s footprint in the map. One solution is selected and sent to the platform controller. If a haptic tag is acquired during manipulation, the process described in II-D is started.

Possible contact points must fulfill multiple conditions – solutions $x$ (a) lie on a path circling the object, similar to the path $\pi$ discussed below, (b) are “free”, i.e. $m_d(x) > r_{\text{rob}}$, (c) can be reached by the robot, i.e. the path planner finds a path to $x$, and (d) allow for stable manipulation. The latter condition depends on the end-effector used. In the presented system, $v$ should go through the connecting line between the object’s center of mass and the floor, since the object would otherwise start to rotate. It is most stable if the normal of the object’s footprint at the contact point is parallel to $v$, thus the platform should approach the surface in a perpendicular fashion. The center of mass is assumed to be in the center of the footprint – if that assumption is wrong, the contact point can be corrected after the first exploration using $H_D$ and $H_R$ (haptic tag).

The push target point $y$ must be chosen such that the new location is reachable, unblocks the blocked path and does not introduce any new blockades. The shortest path $\pi$ on which the object can be circled is defined by all points which are $r_{\text{rob}}$ from the closest point of the object footprint. If, for a potential object location $y$ on the map, each point on $\pi$ has $m_d(\pi > r_{\text{rob}}$, there is also no other obstacle that blocks the platform, and $y$ is a valid target point. However, there are more solutions, such as pushing the object close to a wall or another static obstacle, which is very feasible in practice. To find these solutions, $\pi$ is split into subpath $\pi_i$ at points $\xi \in \pi$ where $m_d(\xi) = r_{\text{rob}} \land \frac{1}{2}m_d(\xi) \neq 0$. In case $y$ is close to only a single obstacle, there is exactly one subpath for which $m_d(\pi_i) < r_{\text{rob}}$. If multiple of such subpaths exist, location $y$ is close to two obstacles and would introduce a blockade of paths. A map of possible target points is generated for a local neighborhood within the manipulation range $r_{\text{mani}}$:

$$m_{\text{goal}}(y) = m_d(y) > r_{\text{obj}} \land |\{\pi_i | m_d(\xi) < r_{\text{rob}} \forall \xi \in \pi\}| \leq 1$$ (7)

Fig. 6 shows this map in green and exemplifies points on $\pi$ for several target points $y$ located in free space, close to walls or corners and in a narrow doorway. Note the different detected subpath $\pi_i$ for these targets. For simplicity of illustration, the footprint is approximated by the smallest enclosing circle, such that $\pi$ is a circle as well. It is not required to test the costs of any points between $\pi$ and $y$.

A set of path primitives – such as curves of different radii used in a local planner – is used for pushing path candidates. Their validity is verified by checking $m_d$ at all points along a path, and for the remaining paths it is checked whether they establish connections between pairs of contact points and target points. The resulting triples (contact point, path, target point) are possible solutions for the manipulation problem. However, since the presented platform does not provide any lateral stability when pushing an object, only a straight pushing path ensures reliable manipulation. The straight line connects between contact point, center of mass and target point. Such a feasible push plan is typically found if there is enough free space on opposite sides of the object.

IV. EXPERIMENTS

A. Sensor accuracy

In order to determine the accuracy of the proposed visuo-haptic sensor, objects are pushed into the foam with different forces using a KUKA lightweight robot arm. The applied force is measured using a commercial, factory-calibrated JR3 force sensor, which serves as the reference. At the same time forces obtained from the proposed sensor with Eqn. (1) are recorded and plotted against the reference force in Fig. 7a. The experiment is repeated with different object shapes as depicted in Fig. 7b–d – two cylinders with diameters 4cm and 12cm, a large plate as well as one with 4cm width. Note that the pressure applied to the foam by the small plate has strong discontinuities at the edges and resembles a step function. The impression in the foam is smoothed out due to internal tension, see also Sec. II.
A polynomial of order three is fitted through the obtained datapoints, yielding the characteristic curves in Fig. 7a. Ideally, the curves should follow the black dashed line – yet a small systematic error is observed, which depends on the object shape: Small forces are underestimated, especially for large objects, which may be attributed to the high slope of the sensitivity curve in that region. Large forces are underestimated for all object shapes. This error could be further reduced by a more accurate foam deformation model. Note that the curve for the small plate (d) is close to the ideal line, despite the fact that the foam impression extends significantly beyond the edges of this object. Thus, the modeling of such boundary effects, as discussed in Sec. II, proves to be valid.

Individual datapoints for object (d) are depicted with an x-symbol in Fig. 7a. The different colors represent four different experiments and show that the repeatability is within the value of the noise. The observed standard-deviation for the proposed sensor is $\sigma = 0.075N$ along the entire range. This is lower than the value observed for the JR3 reference, $\sigma_{ref} = 0.15N$. However, it must be noted that the JR3 sensor has a sampling rate of up to several kHz and might have been distorted by vibrations of the robot arm. Since the force Eqn. (1) is calculated from multiple points in the image, there is an inherent smoothing effect.

### B. Visuo-Haptic Sensor

First, the mobile platform equipped with the proposed visuo-haptic sensor is driven towards several different obstacles...
in a room, such as boxes, bottles, tables, doors and walls. Contact with an obstacle is detected when the foam rod starts to deform. The speed of the platform is reduced based on the visual proximity detector to avoid damage to the object. The movement is stopped completely if one of the following conditions becomes true: (a) the amount of deformation goes beyond an upper limit, i.e. the strain goes to the densification region, (b) the total force, see Eqn. (1), reaches the pushing capabilities of the robot, (c) the robot moves for a distance larger than the width of the sensor. In the latter case, a movable object is observed, and the measured force corresponds to the friction force of the object. In the other two cases, the explored object is fixed – at least for the capabilities of the platform.

For both kinds of objects, fitting of geometric primitives (Sec. II-B) is triggered during the halt of the platform. Measurements are most stable in the static case and allow for the best possible fit. Examples of some explored objects are shown in Fig. 8, together with the fitted geometric primitive. The correct primitives are fitted with high accuracy in images (b)-(e) and (g). The bottles in (e) and (f) are (partially) transparent and could thus not be detected by purely visual methods. The fitted circle primitives have the correct size for (e), but are slightly under- or overestimated for the two bottles in (f). This may be attributed to an imperfect fit of the primitive at the boundaries of the deformation region.

Finally, the platform moves back, and the haptic tag is generated according to Sec. II-D. Table I shows results obtained for various objects. The following important object properties can be obtained from the tag: Fixed, i.e. the platform did not succeed to move a heavy object (PC); falling objects (cleaner) react in a sudden movement when pushed and can thus not be reliably manipulated; deformable materials retreat when pushed by the foam (here the cushion is fixed to apply a large force). The remaining objects are movable, but require significantly different efforts. Note how a larger weight of the same object (paper bin, vase) is detected by an increased friction force. The door exhibits a large drift, since it rotates around its hinge and it continues its motion after the platform stops. Finally, a large drift is observed during the first exploration of the vase: Its rectangular footprint was touched at a corner, such that the object rotated during the push.

C. Exploration System

The proposed navigation and manipulation scheme is implemented on a robotic platform, integrating the proposed approaches, i.e. mapping, acquisition of haptic tags, generation of the semantic navigation/manipulation graph and corresponding task planning. At first, the platform is driven manually around the scene to have the Visual SLAM system acquire a map as outlined in Sec. I-A4. The map obtained from an office environment is shown in Fig. 9b. (Of course, the map can also be loaded from a previous exploration. An automatic approach for exploration planning based on the detection of unexplored scene parts may be used, e.g. [18].)

Next, the map is searched for explorable objects, i.e. objects which look like they could be movable by the platform. They are selected based on a simple scheme using the 2D map: First, connected regions corresponding to the footprints of objects are extracted. Since the visual map does not necessarily show the complete footprint, the convex hull is calculated. The main selection criteria is the size \( s_{x,y} \) of the footprint: Large objects are usually heavy and cannot be pushed by small platforms, while small objects either fall when they are pushed, or belong to larger structures, such as tables. Only regions with \( s_{x,y} \in [0.1m, 0.7m] \) are kept and explored by the platform one by one based on proximity and the possible cost benefit according to Eqn. (6). Fig. 9b shows the filtered regions in orange. Haptic tags are obtained for the selected objects (Table I) and associated to the map. Thus, the acquired haptic knowledge is stored and can be refined over time by additional explorations.

After haptic exploration, the visuo-haptic semantic graph is generated according to Sec. III-B. The navigation graph built directly from the map (Fig. 9c) shows that there is no direct connection between close-by nodes in the lower left part of the room – due to a blockade by the stool (\( \beta \)). The connecting path is thus quite long (depicted in green). Manipulation nodes are added to the graph (blue, Fig. 9d) for all candidate objects, with costs determined from the haptic tag. As the stool is detected to be movable during exploration, it is connected to other nodes with costs \( t_{manip} = 7s \). Now, a much shorter and less costly path is found between the same two nodes. The push plan obtained for the stool is depicted by a blue line in Fig. 9b. The lengths/costs for the two paths are 6.0m/26s for pure (visual) navigation (\( v_{max} = 0.3m/s \)) or 1.8m/14s for the combined navigation/manipulation plan, which includes 7s for manipulation. Manipulation costs include a constant of 4s which is typically needed to position the platform correctly.

In the top right of the room there are two detached nodes (Fig. 9c) because of a blockade by object (\( \delta \)). One of the feasible push plans (blue in Fig. 9b) can remove the obstacle and enable access to that part of the room. Note that the manipulation nodes for objects (\( \alpha, \gamma \)) exhibit a high connectivity to neighbors – yet, due to the manipulation costs, the associated edges would not be chosen by a path planner.

Visual processing for the proposed sensor runs at 30Hz camera frame rate even on larger images (1600 \( \times \) 896 pixels) using an i7 platform. This is due to the fact that tracking relies on interest points (on the foam contour or within the object region), instead of using the entire image. The high-level graph-based planning algorithms are implemented in MATLAB and require a few seconds for processing on larger maps as presented in Fig. 4b. Note, however, that there are no strict realtime requirements for high-level planning.

V. CONCLUSION

In this work a novel visuo-haptic sensor is presented for the exploration of obstacles in household and office environments. It consists of a deformable, passive material which is measured visually using an inexpensive standard webcam. At the same time, the camera tracks the vicinity of the manipulator to measure ego-motion and motion of close-by objects. This low-cost sensor is attached to a mobile platform to acquire haptic information about the environment.
into the visual map using a probabilistic approach. It may be feasible to integrate the haptic geometric primitives cues could be chosen adaptively—e.g. for textureless objects a be increased by relying on additional cues. The most adequate depends on the appearance of the environment, stability can be tested to allow for more accurate deformation tracking. For instance, decreasing the initial contact area by using a triangular cross section would increase the sensitivity of the foam material to low forces. Since the suitability of visual cues manipulation operations (pushing objects). In future work, different sensor materials and shapes will be tested to allow for more accurate deformation tracking. For instance, decreasing the initial contact area by using a triangular cross section would increase the sensitivity of the foam material to low forces. Since the suitability of visual cues depends on the appearance of the environment, stability can be increased by relying on additional cues. The most adequate cues could be chosen adaptively—e.g. for textureless objects a 3D tracker based on a depth camera could be chosen. Finally, it may be feasible to integrate the haptic geometric primitives into the visual map using a probabilistic approach.

The mobile platform is used to explore obstacles and measure their haptic tags—i.e. object-based haptic properties which are relevant for manipulation tasks. The tags are attached to the corresponding objects in a scene map. Later on they are used for manipulation planning, i.e. to push an obstacle out of the way in order to access another part of the map. A graph-based integrated representation for navigation and manipulation tasks in a building is presented. Paths in that graph represent a high-level task list for navigation and manipulation operations (pushing objects).

In future work, different sensor materials and shapes will be tested to allow for more accurate deformation tracking. For instance, decreasing the initial contact area by using a triangular cross section would increase the sensitivity of the foam material to low forces. Since the suitability of visual cues depends on the appearance of the environment, stability can be increased by relying on additional cues. The most adequate cues could be chosen adaptively—e.g. for textureless objects a 3D tracker based on a depth camera could be chosen. Finally, it may be feasible to integrate the haptic geometric primitives into the visual map using a probabilistic approach.

### REFERENCES


![Image](image_url)
two nodes is shown in green and involves a long detour.

Fig. 9: The proposed exploration system is tested in an office scene. (a) Photo of the scene; platform exploring the objects vase and stool, (b) Map of the scene with candidate objects for exploration (α − δ). Feasible manipulation plans are exemplified for β, δ. The photo location is depicted by a red triangle. (c) Navigation graph built from the map. The shortest path between two nodes is shown in green and involves a long detour. (d) Manipulation nodes are added to the graph (concentric circles, yellow regions) and allow for a much shorter and faster path.


Nicolas Alt holds a master degree with honors in Systems of Information and Multimedia Technology from the Technische Universität München, Munich, Germany (2008), and a master degree in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, GA, USA (2007). He is currently pursuing his Ph.D. degree at the Institute for Media Technology at the Technische Universität München. As a visiting researcher, he joined the Advanced Robotics and Autonomous Systems group at INRIA, Sophia Antipolis, France, in 2008 and 2012. His research interests include perception for cognitive robots with a special focus on visuo-haptic modeling and multimodal sensing.

Eckehard Steinbach (SM’08) studied electrical engineering at the University of Karlsruhe, Karlsruhe, Germany, the University of Essex, Essex, U.K., and ESIEE, Paris, France. He received the Engineering Doctorate from the University of Erlangen-Nuremberg, Germany, in 1999. From 1994 to 2000, he was a member of the research staff of the Image Communication Group, University of Erlangen-Nuremberg. From February 2000 to December 2001, he was a Postdoctoral Fellow with the Information Systems Laboratory, Stanford University, Stanford, CA. In February 2002, he joined the Department of Electrical Engineering and Information Technology, Technische Universität München, Munich, Germany, where he is currently a Full Professor. His research interests are in the area of audiovisual-haptic information processing and communication as well as networked and interactive multimedia systems.