2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

A Framework for Dynamic Sensory Substitution

Artashes Mkhitaryan and Darius Burschka

Abstract— **In this paper we present a framework for dynamic substitution of different sensory modalities with existing physical sensors. Our system is capable of finding the most optimal set of mathematical and physical transformations between two modalities of physical and virtual sensors. It allows a creation of new virtual sensors from given set of physical sensors. The virtual sensing may extend to new sensing modalities for which no direct physical sensors exist. The framework optimizes for a minimal error and optimal observation in the resulting fusion. It is processing the chain for a given spatial measurement and measurement range. The framework is capable of increasing the reliability of acquired data in multi-sensor systems by being able to asses the amount of accumulated errors. We give two examples of real-world applications of this framework in robotic environments.**

I. INTRODUCTION

Perception becomes an increasingly important module in autonomous mobile systems that need to create their own mission plans from sensor data. The increasing demand on various sensing modalities requires an increased number of physical sensors on the plaforms. Similarly, smartphones are fitted with a variety of basic sensors like, monocular camera, gyroscope, GPS, accelerometer, microphone and much more. An approach that allows the fusion of sensory data from different modalities does not only increase the reliability of the acquired data by adding redundancy to the system, but widens the spectrum of sensible modalities, using virtual sensors. An example of this is illustrated in Fig. 1, where a virtual force sensor is created by means of an optical camera observing the deformations of an elastic membrane. This example will be discussed in detail in section III-A. Although current cognitive neuroscience research treats Sensory Substitution as visual input substitution by either acoustic or tactile modalities, the goal of this paper is to provide a framework that generalizes this concept to any sensing modality.

Our framework estimates the optimal chain of transformations from the sensing domain of the original sensor(s) to the desired sensing modality, using a set of physical laws and mathematical operators. It takes into account not only the modality that the initial sensor operates in but also its operating range and the expected error. The chain of transformations is established by applying the weighted Dijkstra's search algorithm on a connected graph, whose nodes are the available transformations. In situations where the initial operating range of the original sensors changes, our framework dynamically reconfigures the graph to the current conditions.

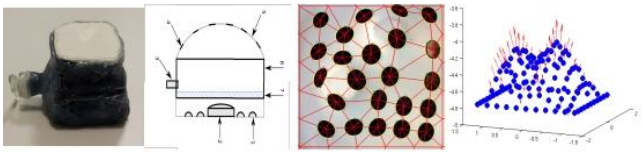


Fig. 1. An example of a virtual force sensor. It is realized by an optical camera, which registers the deformations of a plastic membrane.

According to the type of the considered sensory signal, the topic of sensory substitution can be divided into corresponding domains, i.e. haptic, visual, audio, etc. While there are currently many approaches that specialize on substitution of individual classes of sensory signals, in general a method that unifies all the types into one framework is missing. The two largest parts of the mentioned research focus on haptic sensory substitution, that is aimed at the development of prosthetics, and visual sensory substitution that is aimed for the compensation of visual impairment.

Haptic sensory substitution can be further divided into subcategories: methods that inherit the electro-mechanical approach and those that adopt the vision based approach. Damian et. al. [1] present an artificial skin for prosthetic limbs, which is designed to help detect slippage. It uses the electro-mechanical approach to compute the 2 dimensional friction forces occurring over its surface. Another example of electro-mechanical force sensing is described in [2]. The authors present a force sensor for teleoperation of remote limbs. Former contains a number of hollow cylinders with different heights, that are enclosed in one another eventually forming a conical pyramid. Based on the number of shifted cylinders and the magnitude of the shifts, the authors are able to estimate the perpendicular force vector and the approximate area of the contact. Both of the previous methods are only providing partial information regarding the contact, and the forces acting on it.

More recent works specializing in haptic sensory substitution are concentrating on vision based approaches. In their paper [3], [4], [5], present a sensitive fingertip sensor that consists of a CCD camera mounted on one side of a clear silicon, and two rows of colored spherical markers that are located within the clear silicon. Based on the observed shifts between the two rows of the markers, the authors present an algorithm for three dimensional force distribution reconstruction. Another method for vision based haptic sensory substitution is described by Mkhitaryan et. al.

This work has been supported by an internal grant "Real-Time Perception and Exploration with Collaborating Agents" of the German Aerospace Center (DLR)

Artashes Mkhitaryan and Darius Burschka are with Faculty of Informatics, Technical University of Munich, Boltzmannstrae 3, 85748 Garching bei Mnchen, Germany {mkhitary|burschka}@in.tum.de

[6]. The sensor consists of a CCD camera mounted on one side of a hollow frame and a rubber membrane mounted on the other side of the hollow frame in an airtight manner. The authors estimate the acting forces and the contact shape based on the three dimensional reconstruction of the membrane. Most of the works described above are concentrating on one or two aspects needed for full tactile sensory substitution. The need for a framework that will allow the combination of different aspects into one is apparent.

The methods for visual sensory substitution are less structured than the ones for haptic sensory substitution, however some categorization can be done here as well. One set of those methods can be classified as vision based. Johnson et. al. [7] describe a device for improving the navigation of visually impaired people. The device consists of a stereo pair, a processing unit and a tactor belt. It operates by converting the visual information into vibrations that are performed by the belt. The authors where successful in obstacle avoidance in sparse environments. In [8] another device is described for vision based visual sensory substitution. The authors use a camera mounted on the head of the human user. Latter is connected to a processing unit, which converts the images into 144 low-voltage impulses that are sent by means of a ribbon cable into the mouth of the user. As a result human operator was able to catch a rolling ball solely based on the input from this device.

Many papers in biology discuss the mechanics behind the sensory substitution for humans and animals, that have lost their vision or never had it. In his paper, Rauschecker performs experiments on auditory localization between cats that where blind since birth and sighted cats. The results have shown that as a compensation to visual impairment the blind cats where better at auditory navigation. In his work Windsor [10] describes the navigation mechanism used by the Blind Mexican cave fish (Astyanax fasciatus). Here, the fish uses the reflections of the waves coursed by its motions to successfully avoid obstacles, and form an idea regarding the surrounding terrain.

Other cases of sensory substitution in nature can serve as strong basis for sensor development and substitution in man made devices. In [11] authors describe the logistics behind auditory localization performed by snakes. Snakes lack a tympanic membrane and the external ear openings, but are equipped with a perfectly functioning inner ear. They can localize the pray based on the vibrations of the sand by placing their jaw on it, which allows the transfer of vibrations to the inner ear. Farnosch [12] et. al. describe in their work the model by which frogs detect their pray. Using many of it's lateral organs it can not only determine the direction and the nature of the motion occurring in distance, but also can distinguish between two different sources of motions. In [13] the authors describe the mechanics by which the snakes equipped with a poor infrared sensor determine their pray.

II. THE FRAMEWORK

In this paper we present a framework for dynamic sensory substitution. Our framework is able to dynamically establish a link between the modalities in which the initial sensors operate and the modalities which need to be sensed through a chain of transformations. The latter are represented as nodes in a connected graph, with the initial sensors marked as starting points, and the target sensing modality is marked as the goal Fig. 2. We call this graph, *transformation graph*. The edge weights of the transformation graph are computed based on the inputs of the corresponding node and the error due to the transformation. An optimal chain is then obtained by applying the extended, weighted Dijkstra's search algorithm on the aforementioned graph.

A. Creation of the Connected Graph

As we already mentioned, we define the available mathematical and physical transformations as the nodes of our connected graph. The nodes are described based on their input and output arguments, see Fig. 3. Here the left block represents the input arguments, i.e. the arguments required for the transformation, and the right block represents the output arguments, the arguments that result due to the transformation. Further we connect all the inputs and outputs with the corresponding arguments. An example of a completed transformation graph is given in Fig. 2.

Based on the type of input arguments, that the transformation requires we distinguish between three classes of input blocks, basic, iteration dependent and high-order.

1) Basic Input Blocks: Basic input blocks belong to transformations that require one unique input argument, such as a simple multiplication by a scalar. Fig. 3 (**A**) illustrates a case with a basic input block. This could be used to compute force from the output of an accelerometer.

2) Iteration Dependent Blocks: These input blocks belong to transformations that along side with the current argument require the value of the argument from previous iterations, see Fig. 3 (**B**). An example of a transformation that has an iteration dependent input could be a computation of a derivative, e.g to determine velocity from displacement.

3) High-order Input Blocks: The third class corresponds to an input block that belongs to a transformation, which requires multiple consecutive values of its input argument, see Fig. 3 (**C**). An example of a transformation requiring a high-order input block could be sub-sampling a signal.

The difference between the high-order and the iteration dependent input blocks is that the latter store the current value of the argument for use in the next iterations, whereas the high-order input blocks do not maintain any state.

The edges of the transformation graph are weighted based on the class of the input block they connect to. The weight of each edge is defined as the combination of two parts, an additive and a multiplicative. The numbers written in dashed circles Fig. 3 represent the weight of each type. Here the left number is the additive and the right number is the multiplicative.

1) Additive: The additive weight represents the expected error that would occur due to the transformation. Fig. 4 illustrates a simple case for computing the additive weights. The initial sensor operates in $[0; 6.6]$ range, without errors.

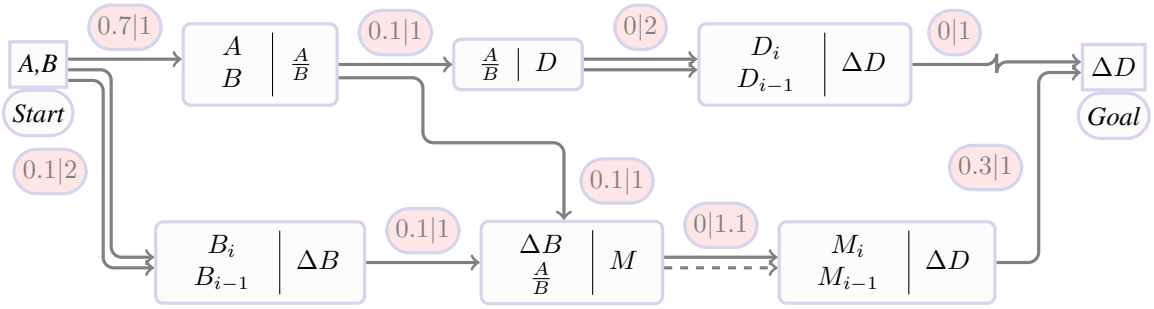


Fig. 2. Depiction of the generic transformation graph, described in Sections II-A and II-B.

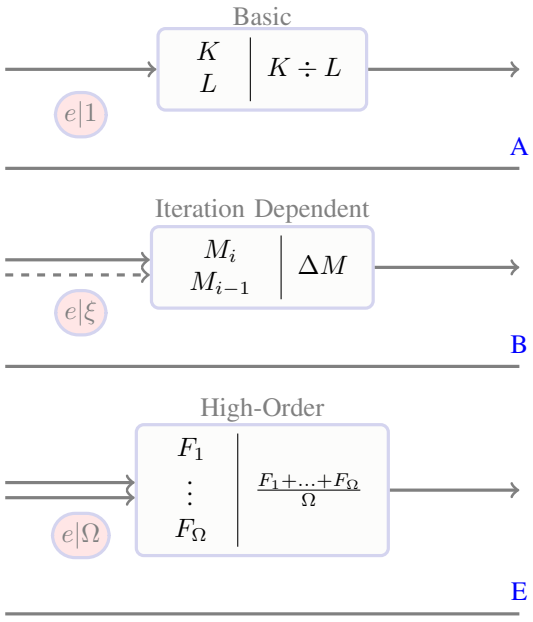


Fig. 3. Different classes of transformation blocks. (A) a transformation with a basic input block. Those are linear transformations that require only one value of its input argument. (B) a transformation with an iteration dependent input block. Those represent the first class of non linear transformations that require sequential values of the input argument from the previous and current iterations. Here, ξ is the multiplicative of the weight, that is higher than 1, due to the non linear nature of the transformation. (C) Transformations with high-order input blocks. Those require multiple consecutive values of the input argument, and do not retain state. Here Ω is the multiplicative weight, that is equal to the order of the input block.

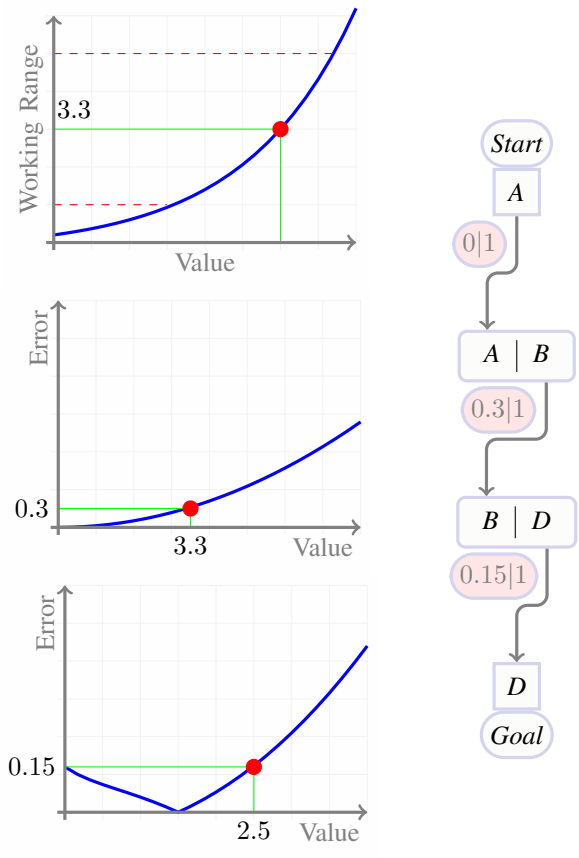
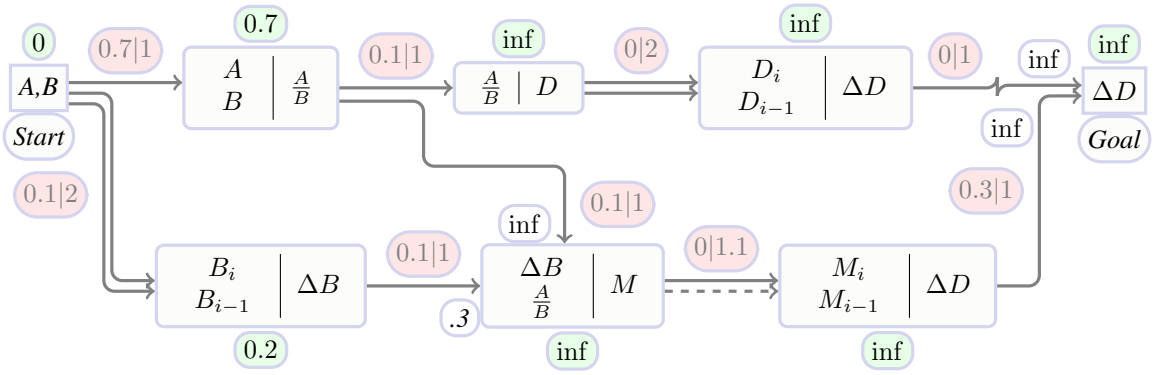


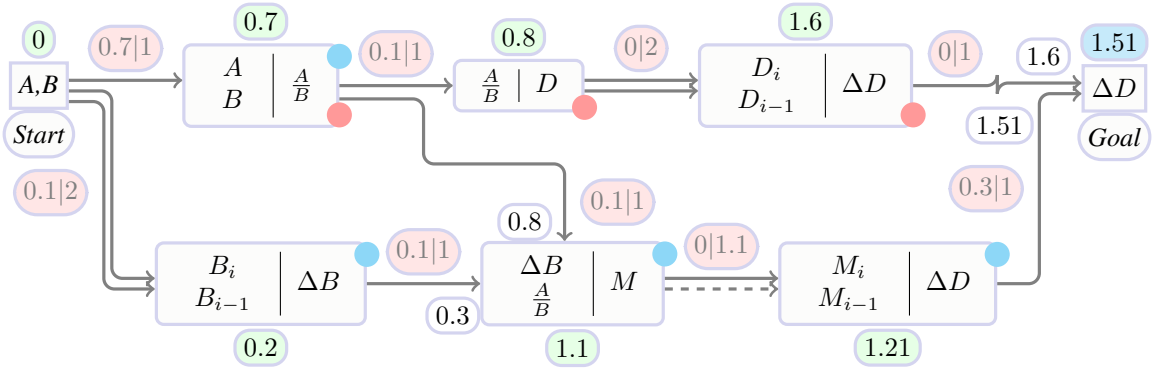
Fig. 4. Illustration of the process of assigning additive weights to the edges.

Thus, assuming normal distribution the expected value is 3.3. Since the sensor operates without any errors the additive weight of the first edge would be 0. According to the error profile of the first input block the expected error is 0.3. Therefore additive weight of the second edge would be 0.3. Further, due to the first transformation the expected value would change to 2.5. Henceforth, using the new expected value and repeating the same steps as before the additive edge of the third edge would be 0.15. If there are further transformations available along the chain this process is repeated over and over again until the goal is reached.

2) Multiplicative: The multiplicative weights represent the amount of transformations that where performed until the current node was reached. Their main role is to emphasize the accumulated transformation errors from previous steps, especially for those with non basic input blocks. Thus, the weights of the edges depend on the class of the input block they connect to. Since the basic input blocks require as an input just one unique argument. It will travel through the transformation chain only once. Hence we assign the multiplicative weights of the edges connecting to basic input blocks to 1. The transformations with iteration dependent input blocks require the values of the argument from current and previous iterations to perform. In comparison to transformations with basic input blocks, here the transformation is performed on two values with the same error order. Thus error accumulation will not be linear. Therefore the edges



(a)



(b) Final result of the search

Fig. 5. 5(a) a depiction of an intermediate step of the search algorithm. The step was selected to illustrate the handling of multiple inputs of a transformation. 5(b) the final result, after all the distances to the goal have been computed.

connecting to iteration dependent blocks have multiplicative weights that are higher than 1. Lastly, the higher-order input blocks require multiple consecutive values, all of which have to travel through the chain of transformations up to that point. Therefore the amount of the accumulated error would be directly correlated to the amount of times the value of the argument had to travel through the chain. Thus the multiplicative weight for the edges connecting to high-oder input blocks is set to the order of the corresponding block.

B. Extended Dijkstra's Algorithm

In this section we discuss a generic example to illustrate all the steps of our framework, as well as show the significance of the assigned weights. The graph contains transformations with all the mentioned input block classes. As already mentioned, we use an Extended Dijkstara's algorithm for finding the optimal set of transformations. We amended the assignment of weights, as well as the computation of the path distance value. The steps of Dijkstara's search algorithm are as follows:

Fig. 2 illustrates the generic graph that was constructed according to the rules described in the section above. Since the graph is generic the additive weights are chosen in a way that helps to emphasize the effects of transformation blocks and in the scopes of this example they do not carry any physical meaning behind them. For simplicity reasons we defined all the high-order input blocks as 2nd order. Also,

Algorithm 1 Dijkstra (Graph, *source*, *target*)

```
for all V ∈ Graph do
    dist[V.index] = inf;
    visited[V.index] = false;
end for
dist[source] = 0;
C = Graph[start];
label: 1;
if C.index = target then
    return dist[C.index];
end if
for all N ∈ V.neighbors do
    if not visited[N.index] then
        d = distance(N,C);
        if dist[N.index] > d then
            dist[N.index] = d;
        end if
    end if
end for
visited[C.index]=true;
C = Graph[index(min(dist[]))];
goto 1;
```

the transformation with the iteration dependent input block represents a numerical derivative. In this case the derivative is computed according to (1), where the M_{i+1} and M_i are the values of the argument from current and previous iteration steps, and h is the step. Since the operation between the two values is a scaled subtraction, the error accumulation due to this transformation is not large. However, it is still an iteration dependent transformation and has a disadvantage to basic transformations, therefore we set its multiplicative to 1.1.

$$M^{'} = \frac{M_{i+1} - M_i}{h} \tag{1}$$

Here we are using one initial sensor that is marked as *start*. It is able to sense in [A;B] domain, and has its distance set to 0. The sensor is capable of sensing in [B] dimensionality with relatively small error, however in contrast to that the sensed errors in [A] dimensionality are quite high. The original sensor is connected to two transformations, of which one has a basic input block and the other has a high-order input block. Since the transformation with the basic input block requires both high error [A] and low error [B] arguments as its input, the additive weight is set to a relatively high value of 0.7. On the other hand the transformation with the high-order input block, only requires [B] as its argument, therefore its additive weight is set to a low value of 0.1. However since the latter has a 2nd order high-order input block its multiplicative is set to 2. We apply the above described Dijkstra's search algorithm to compute both of the distances to the goal. At each node we use the following formula to compute the distance to its neighbors:

$$n = (c + a) * m \tag{2}$$

Where n is the distance value of the neighboring node, c is the distance value of the current node, a is the additive and m is the multiplicative.

Fig. 5(a) depicts an intermediate case where one of the transformations requires two inputs that are provided from different sources. In this case we compute the distance to the considered node as a summation of two distances. According to the illustration the distance to one of the inputs is already computed and is set to 0.3, however the distance of the second input is not computed yet, and is infinite. Therefore the global distance to the node is set to $0.3 + inf = inf$.

The final result of the search is illustrated in Fig. 5(b). Here we have two possible chains to reach the goal. One of which requires 4 sequential transformations (marked with cyan), and the other one requires 3 transformations (marked with red dots). Note that, even though the chain marked with cyan dots has more transformations than the other one, it still has a shorter distance. The reasons is that in contrast to the shorter chain, in this case the transformation with high-order input block is located at the very beginning. This means that the error accumulation does not have as much of an impact as in the case where the high-order transformation is located nearly at the end of the chain. Also note that, even though the transformation with the iteration dependent input block has

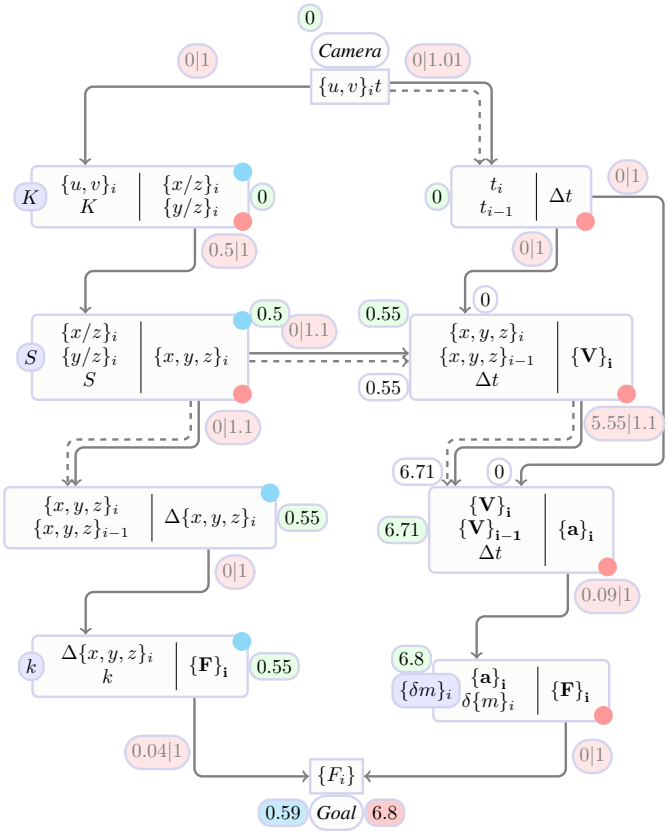


Fig. 6. Transformation graph, for a sensitive fingertip sensor. Here the system is setup to sense forces using a regular CCD camera. The graph contains two possible transformations chains. The firs chain (marked with cyan dots) contains the set of transformations necessary to sense forces using Hook's law. The second chain (marked with red dots) contains the set of transformations necessary for sensing forces using Newton's second law.

and additive weight of 0, it still has an impact on the distance calculated. Where the transformations with basic inputs and 0 additive weights do not affect the distance at all.

III. APPLICATION EXAMPLES

In this section we will be discussing two applications, where our framework is used. In the first application we developed a new sensitive robot fingertip [6], that is able to sense the applied forces over a fingertip. It consists of an elastic membrane (the fingertip), a frame and a CCD camera. The latter is computing the applied forces based on the observation of the deformations of the elastic membrane (Fig. 1). Second example involves a modern smartphone, that is equipped with an accelerometer and a camera. Here the task is to compute the accelerations of the cellphone. The system makes a decision in favor of either using the raw accelerometer data or computing the acceleration based on the visual information from the camera. The decision is made based on the current operating range.

A. Example of a Vision Based Force Sensor

We developed a force sensor that estimates the force field that is applied on an elastic membrane. Our sensor

consists of a CCD Camera that performs a marker based 3D reconstruction of the membrane. Later the system approximates the membrane by a grid of springs and based on its physical transformations estimates the acting forces. There are two main ways for estimating the acting forces (Fig.6). The first is using Hook's law (marked by cyan dots), where the system estimates the acting forces on each node of the grid based on the stretches of adjacent springs. The second one is using Newton's second law (marked by red dots), where the acting forces on each node are estimated based on their accelerations and mass. This method is at a disadvantage to the one based on Hook's law since it requires a dynamic system. Nevertheless we will discuss both the cases to illustrate the nature of the error accumulations and the reliability of the results by either one of them.

Fig.6 illustrates the transformation graph of our setup. Here the main sensor (CCD camera) operates in $[u, v, t]$ dimensions. Where (u, v) are the pixel coordinates on the image plane of the camera, and t is the time of image registration. The chains for both of the methods start with the same two transformations. Those are, the conversion from image coordinate system to world coordinate system, and the 3D reconstruction of the markers. The marker detection algorithm performs with sub-pixel accuracy thus, the error is up to $.5px$ (first transformation from the left), and the errors emerging from the 3D reconstruction are small and can be neglected. The chain splits after the 3D reconstruction transformation. The chain describing the method based on Newton's law proceeds with transformations computing the velocity based on spacial change in time, and acceleration based on velocity change in time accordingly. While the chain describing the method based on Hook's law proceeds by computing stretches of the the approximated springs and then the forces accordingly.

Note, the final distance to the goal of the chain based on Hook's law is significantly smaller than the second chain. There are couple of reasons behind it. The large errors that result due to the transformations in the beginning, belong to both of the chains. However the chain describing the method based on Newton's law is longer than the other one. Therefore the error accumulation in it will amount to a larger value. The second reason is that due to their dynamic nature, the transformations responsible for computation of acceleration based on the spacial change, result into two iteration dependent transformations in a row. The latter also adds large additive errors due to the time discretization.

As a second result, we constructed the sensitive fingertip sensor. The latter is capable of dynamically measuring the force fields over its fingertip, and has a relatively small error of $0.04N$. More detailed information regarding the implementation and the operation of the sensor can be found in [6].

B. Optimal Path for Computing Acceleration

Here we consider a modern cellphone that is equipped with a camera and an accelerometer, the task of this system is to compute the best acceleration estimate of the cellphone.

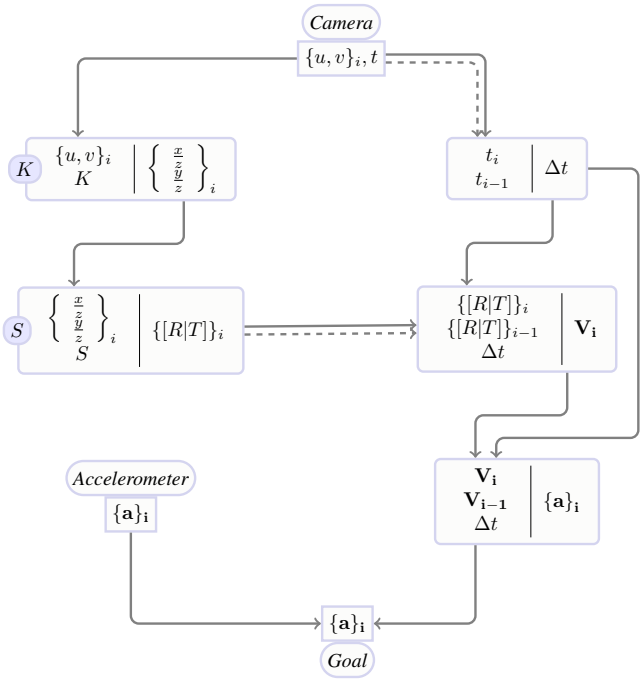
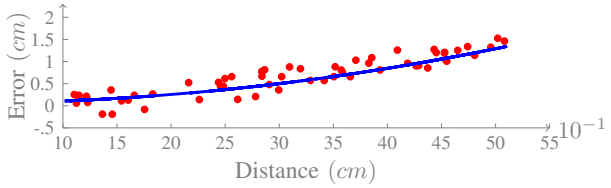
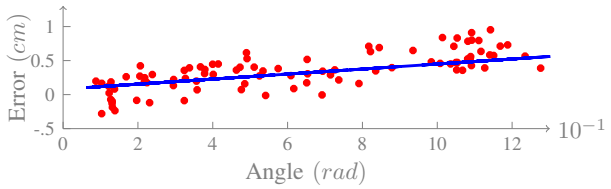


Fig. 7. Depiction of the transformation graph designed to estimate the best acceleration value of a cellphone. There are two ways to perform. First, by reading the data directly from the accelerometer located on the cellphone. Second, by computing the accelerations based on the images acquired by the camera located on the cellphone.



(a)



(b)

Fig. 8. 8(a) Illustrates the error dependency of the 3D reconstruction algorithm to the distance between the marker and the camera. 8(b) Illustrates the error dependency of the 3D reconstruction algorithm to the angle of view. Here, the presented angle is the out-of-plane angle between the cellphone and the surface of the marker, note that the in-plane rotation angels are not presented since they do not add any errors.

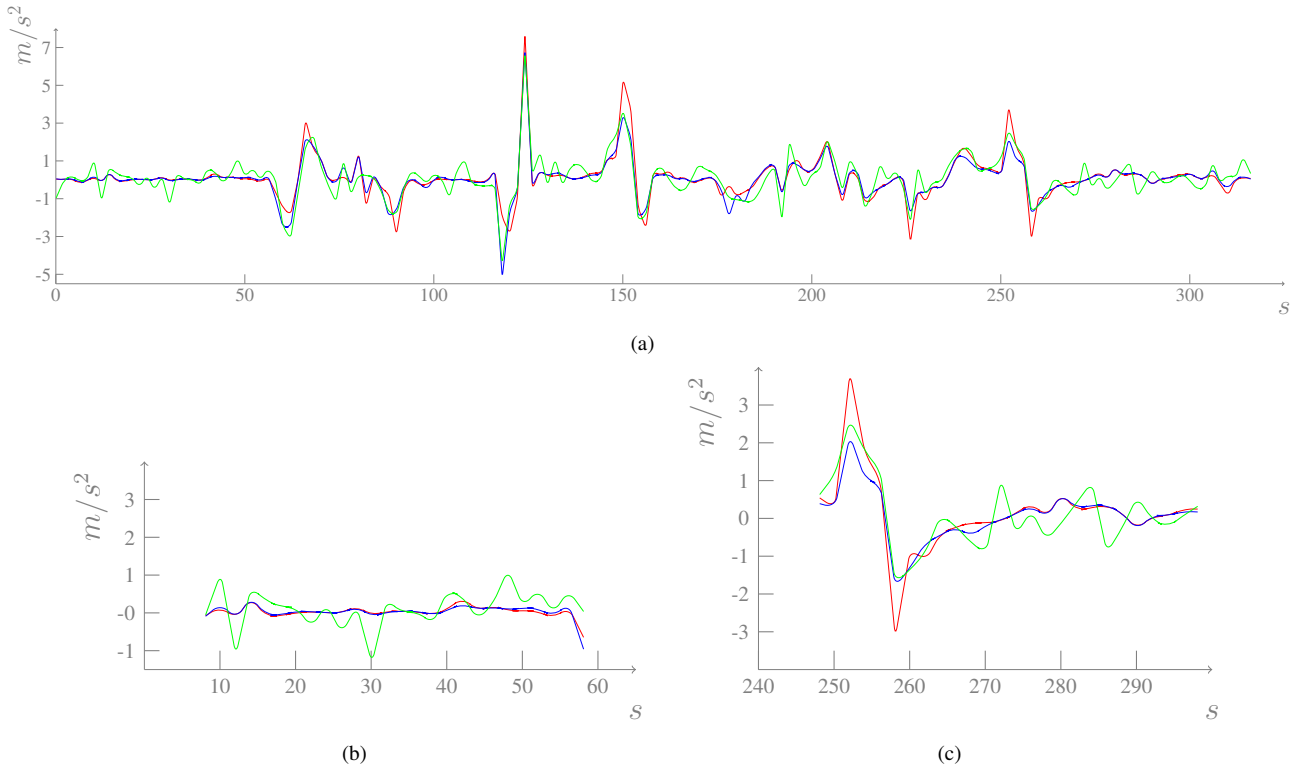


Fig. 10. 10(a) Illustration of the registered accelerations. Here the data registered from the industrial accelerometer (XSens) is plotted in blue. The reconstructed accelerations from the cellphone-camera are plotted in red, and the accelerations registered from the accelerometer located on the cellphone itself are in green. Note that for lower accelerations the error from the cellphone accelerometer is larger than the error from the camera, however for large accelerations this changes. To ensure a better observability of the previous statement we provide two zoomed sections 10(b), 10(c) of the 10(a) graph.

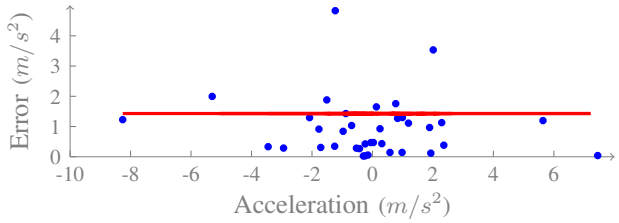


Fig. 9. Error profile of the accelerometer that is embedded in the cellphone. Here the red line illustrates the standard deviation of the errors, that is $1.43m/s^2$.

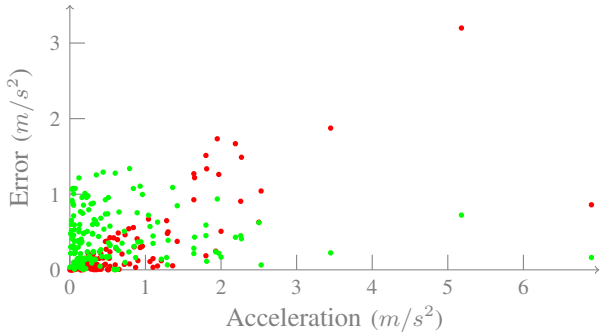


Fig. 11. Error analysis for the two acceleration registration methods. Here the green dots represent errors of the data registered by the accelerometer located on the cellphone and the red dots represent the errors of the accelerations computed from the camera images.

Since accelerometers that come with cellphones usually have quite a low signal to noise ratio, the data they provide is quite unreliable for small accelerations Fig. 9. Note, that the standard deviation of the error for the measured accelerations is about $1.5m/s^2$. This means that the measurements of the accelerometer for accelerations below $1.5m/s^2$ are unsound. On the other hand accelerations computed from the camera have lower errors for smaller accelerations, but tend to be unreliable for large acceleration. This is due to the fact that the derivation error due to time discretization is directly proportional to the magnitudes of acceleration and jerk eq. (3).

$$E_T = \frac{f''(\xi) * \Delta t}{2} \tag{3}$$

Where E_T is the error due to the numerical derivation, $f''(\xi)$ is the second order time derivative of the function and ξ is just a number in the range $[t_0; \Delta t]$ (t_0 being current time). Fig. 7 illustrates the transformation graph of our system. There are two ways of obtaining the accelerations of the cellphone, either directly through the accelerometer or through a set of transformations performed on the camera image. For each iteration, our system is computing the acceleration based on both methods, and depending on the obtained distance, it is deciding which value is more reliable. Here the additive errors are computed dynamically

based on the error profiles of each transformation. Since the camera is only registering images with timestamps, the first transformation (left) is conversion from image coordinate system to world coordinate system. Here (u, v) are the image coordinates and K is the intrinsic camera matrix, $\{x/z\}_i$ and $\{y/z\}_i$ are the pixels in world coordinate system. The additive errors in this step are constant and occur due to the discretization of the space by the camera itself. The second step in the chain is the reconstruction of the camera pose (second transformation left). Here we are using a marker based approach to obtain the rotation and translation $[R|T]$ pair that describe the pose of the camera, S is a scalar describing the size of the marker, and $(\{x/z\}_i, \{y/z\}_i$ are the rays pointing at it. Fig. 8 illustrates the error dependency of the pose reconstruction to the distance between the camera and the marker, and the viewing angle, note that since the error does not depend on in-plane rotations the provided dependency is only for out-of-plane rotation angles. The third step in our chain (second transformation from the right) is the computation of the velocity (V_i) of the camera. This is done based on the current and previous poses of the camera and the time that elapsed between their registration. The error in this case can be computed analytically using eq. (3). The final step (third transformation right) is the computation of the acceleration (a) of the camera, which is done similar to previous step, where the error can also be computed using eq. (3). We tested the system using an industry grade accelerometer (XSense), that we attached to the cellphone to measure the ground truth. Fig. 10 illustrates the results for registered accelerations with all three methods. Here the blue graph represents the ground-truth, the green line depicts the accelerations that are read from the accelerometer located on the cellphone, and the red line shows the reconstructed accelerations from the camera. Fig. 11 illustrates the error analysis for the same case, here the green dots are the errors of the accelerometer and the red dots are the errors of the reconstruction. Overall our system was able to pick the most accurate solution in around 75% of the cases, the decisions where mostly inaccurate where the errors from both methods had similar magnitudes.

IV. CONCLUSIONS

We have presented a framework for dynamic sensory substitution. Our framework is capable of dynamically establishing optimal connections between different sensor modalities through a set of physical and mathematical transformations. Latter allows the creation of virtual sensors that are able to sense in modalities that the original sensors were not designed to operate in. The set of transformations is selected not only based on their amount, but also on the expected errors that occur due to the transformation, as well as the operating range of the sensor. We have shown exemplary that our framework is capable to select the best measurement in cases where there is more than one initial sensor available. It selected the best possible solution in 75% of the cases and underperformed only in cases where both of the possible solutions had very close values.

REFERENCES

[1] Dana D. Damian and Harold Martinez and Konstantinos Dermitzakis and Alejandro Hernandez-Arieta and Rolf Pfeifer "Artificial Ridged Skin for Slippage Speed Detection in Prosthetic Hand Applications" IEEE/RSJ International Conference on Intelligent Robots and Systems October 18-22, 2010, Taipei, Taiwan

[2] Antonio Bicchi, Enzo Pasquale Scilingo, Danilo De Rossi "Haptic Discrimination of Softness in Teleoperation: The Role of the Contact Area Spread Rate" IEEE TRANSACTIONS ON ROBOTICS & AUTOMATION

[3] Katsunari Sato and Kazuto Kamiyama and Naoki Kawakami and Susumu Tachi "Finger-Shaped GelForce: Sensor for Measuring Surface Traction Fields for Robotic Hand" IEEE TRANSACTIONS ON HAPTICS, VOL. 3, NO. 1, JANUARY-MARCH 2010

[4] Kazuto Kamiyama and Kevin Vlack and Terukazu Mizota and Hiroyuki Kajimoto and Naoki Kawakami and Susumu Tachi "Vision-Based Sensor for Real-Time Measuring of Surface Traction Fields" IEEE Computer Graphics and Applications

[5] Kazuto Kamiyama and Hiroyuki Kajimoto and Masahiko Inami and Naoki Kawakami and Susumu Tachi "A Vision-Based Tactile Sensor" ICAT 2001 December 5 Japan

[6] Mkhitaryan Artashes and Darius Burschka. "Vision based haptic multisensor for manipulation of soft, fragile objects" In IEEE SENSORS, 2012.

[7] Lise A. Johnson and Charles M. Higgins "A Navigation Aid for the Blind Using Tactile-Visual Sensory Substitution" Proceedings of the 28th IEEE EMBS Annual International Conference New York City, USA, Aug 30-Sept 3, 2006

[8] Paul Bach-y-Rita and Stephen W. Kercel "Sensory Substitution Human-Machine Interfaces", TRENDS in Cognitive Sciences Vol 7. N. 12, December 2013 ELSEVIER

[9] Josef P. Rauschecker "Compensatory plasticity and sensory substitution in the cerebral cortex" TINS Vol. 18, No. 1, 1995 ELSEVIER

[10] Shane Windsor "Hydrodynamic imaging by blind Mexican cave fish" Thesis (PhD–Biological Sciences)–University of Auckland, 2008.

[11] Paus Friedel and Bruce A. Young and J. Leo van Hemmen "Auditory localization of Ground-Borne Vibrations in Snakes" PhysRev Letters PRL 100, 048701 (2008)

[12] Jan-Mritz P. Franosch and Marion C. Sobotka and Andreas Elepfandt and J. Leo van Hemmen "Minimal Model of Pray Localization through the laterl-line systems" PhysRev Letters Volume 91, Number 15 (2003)

[13] Andreas B. Schert and Paul Friedel and J. Leo van Hemmen "Snake's Perspective on Hear: Reconstruction of Input Using an Imperfect Detection Systems", PhysRev Letters 97, 068105 (2006)