The Cloak of Invisibility: Challenges and Applications

Is it possible to create a cloak of invisibility—a flexible artifact that can make anything inside it invisible and preserve invisibility despite mobility and deformation? Exploring the algorithmic and technological challenges involved reveals tantalizing information.

Franco Zambonelli and Marco Mamei
Università di Modena e Reggio Emilia

This is an unusual article. It does not fall neatly into any of the typical genres of papers published in this magazine, such as a research report, retrospective, survey, or tutorial. Rather, it is a carefully reasoned technical speculation on an intriguing idea—the use of sensor-impregnated garments to achieve invisibility through real-time scene manipulation. Although the paper is pure speculation, the Associate Editors in Chief and I felt that *IEEE Pervasive Computing*'s readership would be well served by including this thought-provoking article in this issue. To provide balance, we include brief comments by vision experts Martial Hebert of Carnegie Mellon University and Steve Shafer of Microsoft Research.

—M. Satyanarayanan, Editor in Chief

Anyone who's read the Harry Potter books by J.K. Rowling understands the concept of an invisibility cloak. Furthermore, humans have dreamt of what invisibility might mean since the beginning of civilization, and this dream persists in today's literature and culture. However, although more improbable methods of invisibility will remain unrealized, an invisibility cloak could be feasible in the future through technology. In this article, we attempt to show this via technical "pencil and paper" arguments and a conceptual implementation.

Generally speaking, we propose a fabric of small computing devices that can receive and retransmit light emissions in a directional way as well as interact with each other in a wireless amorphous network.[1] While someone or something is inside the cloak, the cloak's devices would cause external observers to perceive the light configurations they would see if the wearer wasn't there. Sensors on the side of the cloak not facing the observers could receive such configurations and, via distributed coordination, communicate them to emitters on the observer's side for retransmission.

Building a solid wall exhibiting such properties for a nonmoving observer might be difficult. Even more challenging problems arise when preserving such properties

- For any observer in any position
- For any shape of the cloak
- Despite movement of the cloak's fabric

Although we haven't identified fully fledged technical solutions for all these complex problems, you could collectively exploit numerous recent findings in different research areas (such as mobile computing,[2,3] distributed and peer-to-peer coordination,[4,5] self-organization,[6,7] and MEMS (microelectromechanical)[8] and sensor networks[1]) to sketch promising solutions. Following that, you might then achieve the cloak's actual implementation and numerous related, innovative artifacts.

For the sake of readability, we present our arguments incrementally. For each step, we discuss the

1536-1268/02/$17.00 © 2002 IEEE

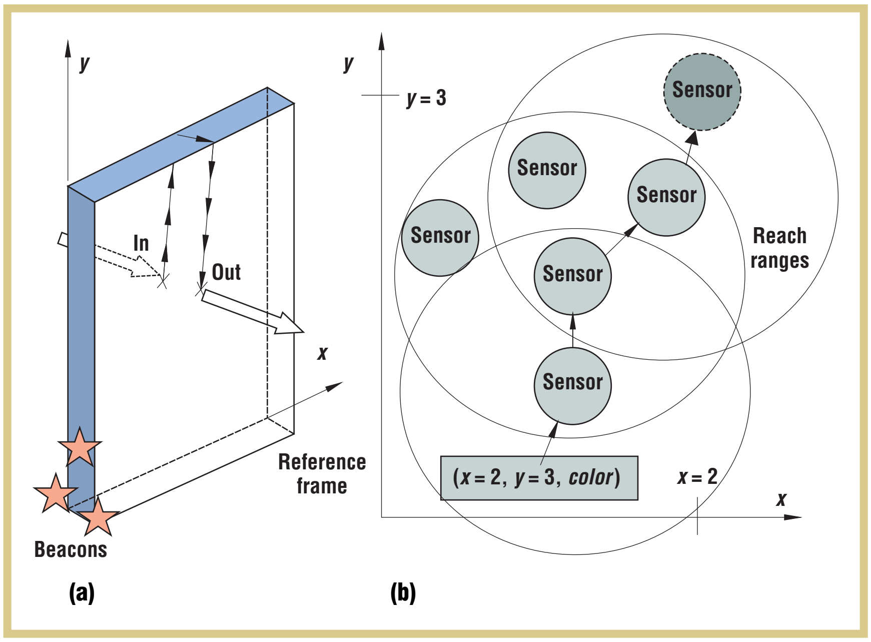


Figure 1. The invisible wall: (a) a global view and (b) local routing of a tuple toward a specific point.

associated hardware and software challenges and the artifacts you might create after resolving the challenges.

Step 1. The invisible wall

Let's consider the basic scenario of making a rigid, flat wall invisible to an unmoving observer at a known position (for example, centered in front of the wall at a known distance from it). You could do this simply by having a camera capture everything behind the wall (from the observer's known fixed perspective) and project it on the front side.

Alternatively, we propose building the invisible wall by using a network of small computer-based devices. (This serves as a basic building block toward defining the invisibility cloak, for which we will remove the constraint of a flat, rigid shape and of a single, fixed observer.) For this, we will need

- *In devices*, sensors that can perceive light emissions and transform them into digital signals, such as a digital camera's CCD sensors
- *Out devices* that can appropriately emit light on the basis of specific signals received, such as LCD displays, LEDs, and microlasers

Of course, you might also have devices acting as both In sensors and Out emitters.

To create the wall, we would deploy densely packed In and Out devices on the wall, one type on each side. We do not consider devices placed in regular, possibly wired, grids as in cameras and monitors. Instead, we consider the wall randomly filled with unwired devices to avoid placement and wiring efforts and to enable, say, "painting" or spraying the wall with transparent glue containing sensors and emitters. Such a choice, while increasing flexibility and fault tolerance, requires devices to have short-range wireless communication capabilities (optical- or radio-based)[9] and to be either internally self-powered (for example, via solar energy or a light battery)[1] or laid on a conductive substrate and fed with external power.[10]

Once applied, each In device must record its local light information on the wall and transmit it to the corresponding Out device on the opposite side, to globally reconstruct the image. Such transmissions, owing to short-range communication capabilities—and to avoid limiting the wall's size and thickness—must occur in multiple steps, by properly routing messages across the network to their destination (see Figure 1a). Of course, this requires both sides of the wall to be seamlessly part of the same network, so that a message can pass from one side to the other by continuously traveling over the network. For example, in Figure 1a, we assume that devices also fill the wall's gray lateral parts.

Software issues

Deploying sensors and emitters without a predetermined layout can dramatically cut the costs of building our wall. However, it complicates software design. In particular, when dealing with the software to control such an artifact, we must ask,

- To properly establish the In and Out pairs, how can devices determine their location?
- How do you route data across the network from In to Out devices?

For the first question, because we have spread the devices randomly on the wall, they do not have any a priori knowledge about their position. However, fundamentally, each device will find its mate by looking for the device on the opposite side of the wall having the same coordinates. We can't use Global Positioning System technology because it lacks the required accuracy and is costly.[3] Alternatively, thanks to recent research in localization technologies and algorithms,[6,11] we can let each device determine its own position in the wall, according to a relative set of coordinates, by relying on a few devices with known positions.

Such localization algorithms can rely on the geometrically intuitive fact that you can uniquely determine a point's position on a surface by measuring its distance from three nonaligned reference points, *beacons*, through *triangulation*.[3,11] So, for the invisible wall, we can establish the global coordinate system in four main steps.

First, we choose at least three beacons—either via an external stimulus or by a leader election algorithm—defining a coordinate's frame (in our case, we might choose devices along the wall's edges, as in Figure 1a).

Second, each beacon must produce some

Simply Difficult

Developing large arrays of microsensors on flexible surfaces is an interesting area of research that could lead to many applications. In that respect, the article investigates interesting concepts for implementing such devices. However, although the goal of invisibility is quite catchy and sure to motivate much discussion, the claims are at odds with standard results in computer vision and computer graphics, and it would be wise to motivate work on sensor and processor networks without resorting to Harry Potter.

Although it's possible, in principle, to project the image received by an array of input sensors through another array of output sensors, the extrapolation from an invisible wall with a fixed observer to a general cloak overlooks several known facts in the geometry and photometry of image formation. More precisely, serious issues arise when you consider the case of an observer in an arbitrary position in a 3D world.

From a geometric standpoint, assuming that the observed world has a general 3D structure—as opposed to a flat world—it is well known that, because of parallax effects, you cannot reconstruct exactly the view that an observer in an arbitrary position would see. Even if you reconstructed the 3D structure from the input sensor—something the authors do not propose—you still could not reconstruct the view because parts of the scene that are occluded from the input sensors' view, but visible to the observer, cannot be reconstructed. A vast literature on scene reconstruction and image-based rendering addresses these issues. Shuffling the set of input rays into a set of output rays will not generally generate a correct image.

Ignoring the geometric issues, faithful reconstruction from an arbitrary observer viewpoint assumes implicitly a simple reflection model for the objects observed in the environment. Specifically, as soon as the light reflected from an object depends on the surface orientation and viewing direction, you can no longer recover in a simple manner the intensity that you would see from a different direction. —*Martial Hebert*

Martial Hebert is a professor at Carnegie Mellon University's Robotics Institute. His interests include computer vision (particularly object recognition and 3D model reconstruction) and mobile robotics (particularly perception for mobility and autonomous driving).

sort of signal to let other devices estimate their distance from the three beacons and eventually, via triangulation, their positions.[3] Devices can estimate distance by exploiting either a physical property of the signal the beacons emit (for example, its attenuation, although this method's accuracy at such small scales might be inadequate) or a geometrical property (for example, the devices' average density coupled with expanding ring hop count messages[6]).

Third, the triangulation process spreads to further devices—as soon as they know their position with respect to the coordinate system—and diffuses across the whole surface.

The last step is to execute a distributed, iterative algorithm[11] on the surface to improve the accuracy of the estimation, which could have produced notable errors during the diffusion process, until reaching the required accuracy. Although this might take some time, the process's execution time is not relevant because it only takes place at bootstrap.

Given a set of coordinates' availability, the same for both sides of the wall, you can restate the second question as: "How can you route the data sensed by an In device at coordinates (x_{In}, y_{In}) to the Out device positioned at the (x_{In}, y_{In}) coordinates on the wall's opposite side?" Or, better, because an Out device might not necessarily exist at the same coordinates (or because such a device could be dead or temporarily unreachable), you might say "How can you route such data to the Out device closest to (x_{In}, y_{In})?" In this case, the answer is rather simple (see Figure 1b). Each In device must propagate a message (that is, a tuple) in the form (x_{In}, y_{In}, *color*), representing the image information it captured at its own coordinates. You can then route such a tuple from device to device toward the wall's closest edge. Because a device knows its own coordinates and can store its neighbors' coordinates (as obtained during localization), it also knows whether it must propagate a message and where, on the basis of simple Euclidean considerations. Once the tuple has reached the wall's opposite side, routing proceeds from device to device toward the (x_{In}, y_{In}) coordinates. Propagation stops when the tuple reaches the Out device at (x_{In}, y_{In}) or when no emitter closer to the goal can be found in the neighborhood.

Optical and hardware issues

At what size do the In and Out devices provide a reasonable visual rendering? Following the Listings-Donders model of the human eye[12] (see Figure 2), we can define θmin as the minimum angle for which an observer would perceive points *A* and *B* separately. This angle is approximately 1/60 degree, the minimum at which two light rays hit two distinct cone cells separated by an unhit cone cell.

Thus, considering Figure 2, you would perceive *A* and *B* separately only if

$$l > d \cdot \tan(1/60) \approx \frac{d}{3400}.$$

For a good image, the distance between two devices must be less than *d*/3400 (for example, you would perceive two objects separated by 1 mm as a single object from a distance of 3,400 mm, or 3.4 m).

For instance, to rend invisible a 1 m^2 wall from a distance of 10 m, you'd need 115,600 devices on each side. In fact, the allowed maximum distance between two devices to provide the impression of smoothness is 10/3400 = 2.9 mm. So, each device must

Kind of Cool

Is this paper a joke or a serious contribution to the literature? This much I know for sure: These authors have an overactive imagination! And that is probably what we all need because the field of pervasive computing is on the whiz-bang edge of computer science to begin with.

On purely technical grounds, do I believe this proposal can work? Not really. The optical sensors and emitters required would be impossibly dense, especially considering the issues involved in each unit's directionality, determination of the viewer's position, intermingling sensors and emitters on the surface, determination of sensor and emitter positions, and so on. Do I believe we're going to invent cloaks of invisibility soon? No.

But it would be kind of cool, wouldn't it? There's something to be said for papers that go way outside the bounds and inspire people to wild feats of imagination. In that regard, I actually wish this article were more about cool concepts and less about how (not) to achieve them. Maybe we can't make an invisibility cloak, but we could make a "mirror" that does something particularly cool. The "maybes" here are endless.

So, is this paper to be taken seriously? Not by me. But who cares? If I were teaching, I might give it to my class anyway and ask them to think about it. This might not be possible in my lifetime, but could be in theirs. —*Steve Shafer*

Steve Shafer is a senior researcher in ubiquitous computing at Microsoft Research. His current work is in location awareness for mobile devices and intelligent environments; his previous work was in computer vision, modeling optical properties of materials, and cameras.

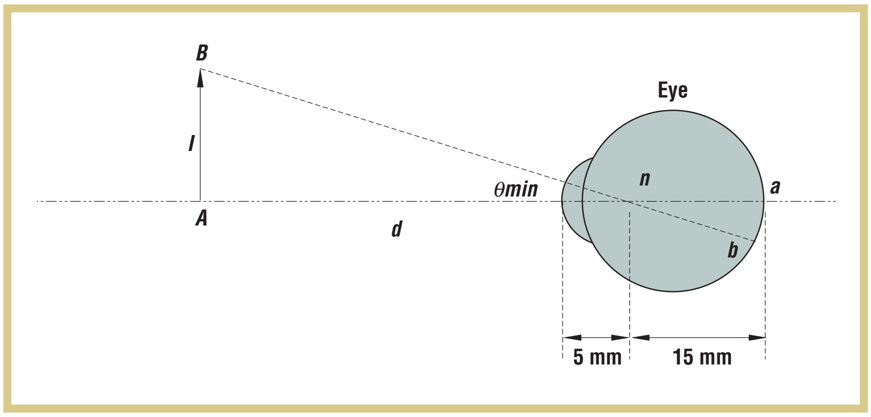


Figure 2. The Listings-Donders model of the human eye.

be approximately $(2.9\ \text{mm})^2 \approx 8.4\ \text{mm}^2$ wide; a linear meter must have at least 340 devices and a square meter 115,600 devices (340^2).

The above requirements appear feasible with regard to the state of the art in optical MEMS technologies. For instance, three years ago, the Smart Dust project at Berkeley developed optical (laser-based) internally powered computer-based sensors and emitters of $1\ \text{mm}^2$, which could well serve our purpose.[13]

Even the amount of data that each device and its wireless communication channels must deal with is not a challenge. If we assume each device should record and transmit a 24-bit value for the color information (16 million colors) with two 16-bit values representing device coordinates, this results in a 56-bit tuple (24 + 16 + 16). You must update these values 30 times per second, the normal television frequency. For a $1\ \text{m}^2$ wall, with devices approximately 2.9 mm apart, and assuming that the communication range lets a device connect only to its closest neighbors (a very strict hypothesis), a tuple's routing process takes 170 steps on average and, in the worst case of centrally located pairs, 340 steps. Thus, devices located at the wall's edges (the ones dealing with the highest traffic) will take charge of routing information for the 170 other devices on the line between them and the wall's center. All this considered, a single device's wireless link bandwidth should be $(170 * 56\ \text{bits}) * 30\ \text{Hz} \approx 286$ kbits/sec to sustain such peaks, which is low enough for modern micro devices to sustain.

We emphasize that we are considering here the human eye's physiological limits. For several applications, less strict constraints (and coarser renderings) might suffice.

Applications

Because the described wall can render the image only from a fixed perspective, it cannot render invisibility: the image produced by the Out devices would not change as an observer moves, making the illusion vanish. Still, a transparent wall can have several potential applications in cases where you'd like to observe without being observed—for example, in therapy and investigation settings as well as for entertainment. This installation has an advantage over more traditional technologies, such as using cameras or mirrors, in its self-containment. It also doesn't require specific infrastructures or skills.

This technology would also allow building further interesting artifacts. Specifically, you could use an analogous amorphous network to produce a paintable television or monitor—that is, a television sold as a

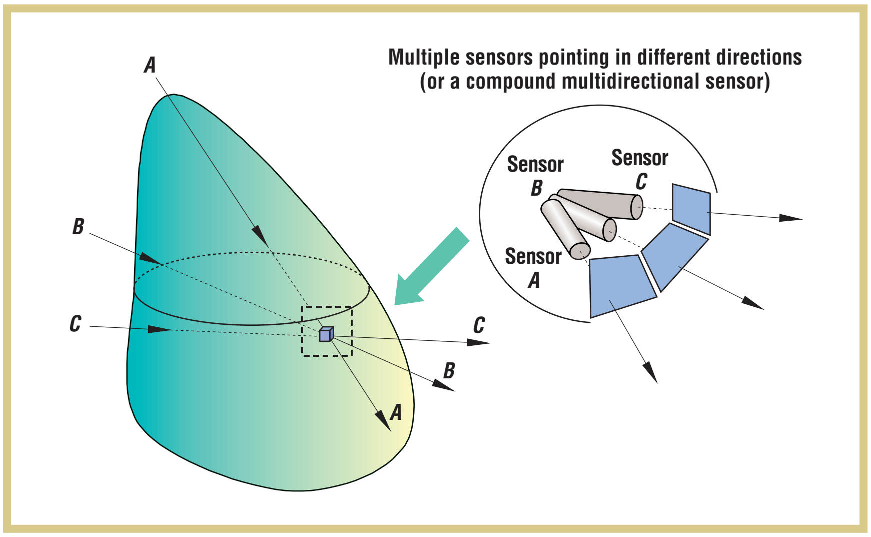

Multiple sensors pointing in different directions (or a compound multidirectional sensor)
Sensor B
Sensor C
Sensor A
A
B
C
C
B
A

Figure 3. The invisible object.

paint that, once painted, works as a normal flat-screen television. In this case, you would need a TV-signal receiver at one of the painted wall's edges to also act as a beacon for the network coordinate system. Once all emitters are localized, the receiver can transmit tuples in the form (x, y, *color*), to be propagated in the network of emitters to render the TV image.

Step 2. The invisible object

You can only enable the real power of invisibility, and a larger class of related applications, by making it possible to paint invisible objects of any shape and by enhancing image rendering to relax the constraint of the fixed and single point of observation. You want the object be completely covered by a sensor network so that, for any point of its surface, a ray of light incident from any direction gets properly captured and retransmitted on the opposite side.

By assuming that one or more observers can move around the object and see all its sides, you must integrate the In and Out devices in a single device or, they must be both densely painted on the whole surface so that you no longer separate the object into In and Out sides. Moreover, if we want the object to show what's behind it independent of the observers' position, each portion of the surface should be able to retransmit different light configurations in different directions, to virtually extend a reasonable number of light rays on the object (see Figure 3). For this purpose, we can consider a device as a compound object capable of acquiring different light information (In function) and of firing different light rays (Out function) in different directions. Alternatively, we can consider a device as a monodirectional sensor or emitter. By distributing these monodirectional devices densely on the object surface with a random orientation, any portion of the surface will have, with high probability, sensor and emitters pointing in all directions

Software issues

In this case, implementing invisibility closely resembles the case of the previously described invisible wall: each In device must provide light information to the Out device on the object's "opposite" side—that is, to the emitter on the surface that is the closest (in surface position and orientation) to the virtual extension of the light ray that the sensor captures. However, in this case, answering how devices determine their position and the In-Out pairs and how you can route data from In to Out devices is more challenging.

For our explanation, it's important to distinguish between *extrinsic* and *intrinsic* coordinates. Extrinsic coordinates identify a device's position and orientation with respect to a 3D frame attached to the object (see Figure 4a). You could represent a device's extrinsic coordinates, for instance, by its (x, y, z) coordinates and by the two angles (θ, ϖ) determining its orientation. Intrinsic coordinates specify devices' positions in the object surface. In other words, they are 2D coordinates mapped on the surface, establishing a frame on the object's surface (see Figure 4b).

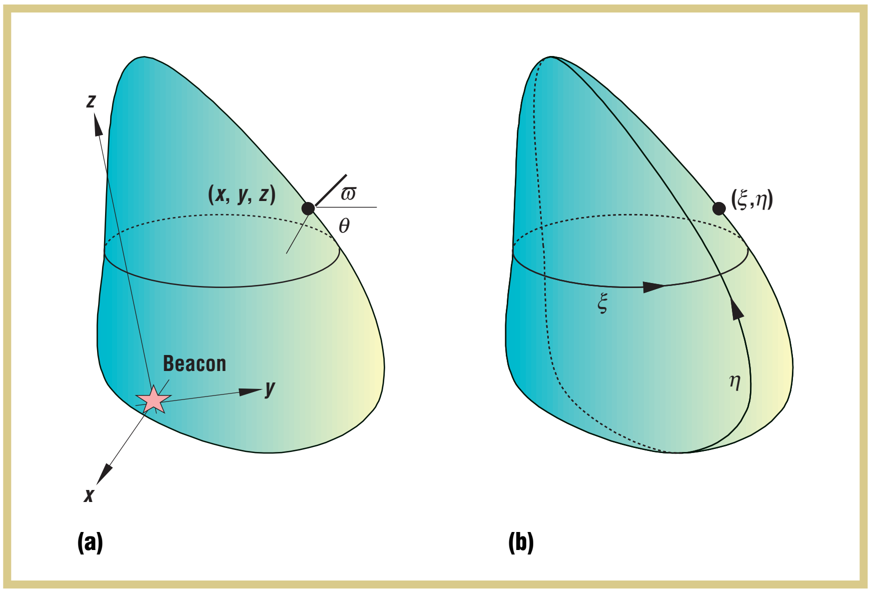


Figure 4. Coordinates: (a) extrinsic and (b) intrinsic.

A device's extrinsic coordinates are fundamentally important in that they unambiguously determine the coefficients of the specific light ray associated with the device—that is, the light ray an In device receives and blocks or the light ray an Out device reproduces. So, you must establish an In-Out pair between two devices whose extrinsic coordinates identify the same straight line—that is, the light ray to be reproduced. Therefore, each device must know its extrinsic coordinates to establish such a pair.

Determining a device's extrinsic coordinates in a distributed way from local information can take place by extending the beacon-based localization mechanism to consider the object's local curvature and the devices' orientation (see Figure 4a). You can determine local curvature from within the surface in a completely distributed manner by taking into account the following geometrical property: While on a plain surface, the ratio of the circumference to a circle's radius is always 2π. On a curved surface, the ratio of the circumference to a circle's "radius," as measured on the surface, decreases as the curvature increases. This is because the measured radius is actually an arc on the surface. Starting from this property, each device can measure the local curvature of the object on which it is located by measuring the circumference and the radius of a small circle centered on itself. This can operatively take place by having each device probe the neighborhood, determine the number of devices at a given distance (the circumference) and the number of devices on the shortest path from the central device to a neighboring device (the radius), and then examine how far the ratio of these numbers is from 2π.[7]

Once you do this, you can get the device's orientation on the surface by the curvature information and by comparing the beacon's orientation with other devices' orientations epidemically. To this end, you must equip each device with a system capable of determining relative orientations, such as the Cricket Compass.[14] Once you've gathered the curvature and devices' orientations, you can easily obtain the extrinsic coordinates by a simple variant of the triangulation procedure.

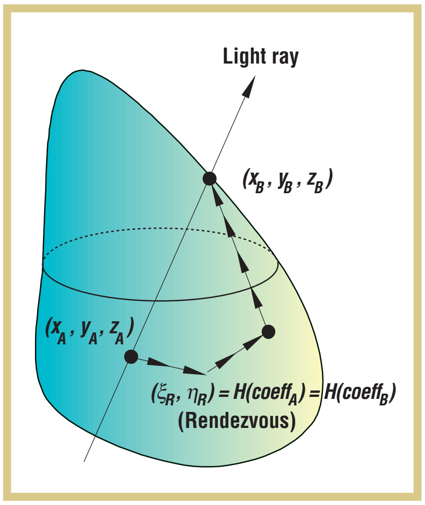


Figure 5. Rendezvous communication.

Unfortunately, even once each device knows its extrinsic coordinates, this knowledge doesn't help establish the correct In-Out pairs that would enable a light ray to be reproduced. In fact, if an In device starts propagating a tuple reporting its extrinsic coordinates and the color to be reproduced ($x, y, z, \theta, \varpi, color$), you cannot exploit this information to properly route the message toward the corresponding Out device. In fact, extrinsic coordinates do not indicate where a light ray entering the object will exit, this being dependent on the object's shape. Without local information about the object's shape (which is impossible to store locally unless the object is very regular), you cannot know a priori where the tuple should eventually arrive or what the right direction is to approach the destination.

To solve this problem without flooding tuples across the whole network, we need a strategy to route information without explicit knowledge of mates' extrinsic coordinates. This is where intrinsic coordinates come in. Evaluating intrinsic coordinates is perfectly analogous to the flat wall case. Also, you can effectively exploit intrinsic coordinates to route tuples toward a specific point in the surface, by making a tuple progressively approach the needed destination.

We propose that each device, once it has determined its extrinsic coordinates, can determine the coefficients *coeffs* of the straight line that coincides with the light ray incident on it. Of course, two In and Out devices are mates if and only if they compute the same (or very close) *coeffs*. Now suppose that all devices agree on a continuous hash function H that maps an equation's coefficients in intrinsic coordinates (ξ, η). An In device A can then send the tuple containing the color information to the device at intrinsic coordinates $H(coeff_A)$. An Out emitter B, on its side of the wall, can try to collect color information from the device at intrinsic coordinates $H(coeff_B)$. So, if A and B are mates, the calculated intrinsic coordinates are $H(coeff_A) = H(coeff_B)$, identifying a unique device closest to those coordinates. This device will act as a rendezvous point (see Figure 5) to establish the pair and exchange the needed information. Of course, the message must carry the extrinsic coordinates to deal with *hash collisions*—that is, to check the correctness of a forming pair in the case of multiple pairs using the same rendezvous node.

You can optimize this process: only the first communication between an In and Out pair occurs via the rendezvous, after which the two devices become known to each other and can interact via direct tuple routing. However, such a solution is less reliable: it does not consider that a device can die or run out of power, a problem that the rendezvous solution, establishing In-Out couples dynamically, does not experience.

Internet-scale peer-to-peer computing and content-based routing have already addressed similar problems. There, peers forming an unstructured and dynamic community must exchange data and messages based not on an IP addressing scheme but rather on the messages' content (for example, "I need the mp3 of 'Hey Jude,' no matter who can provide it to me"). Data structuring in a peer-to-peer community defines a sort of extrinsic coordinate scheme: it is one on which messages must be delivered but provides no help in routing them. So, to avoid Gnutella's flooding approach, various proposals[5,15] exploit the common idea of relying on a virtual overlay network, defining an intrinsic coordinate scheme facilitating routing between peers. If you need to send or receive a message on the basis of its content, you can hash the message's content into the intrinsic coordinates and then route it toward the rendezvous point in the overlay network.

Optical and hardware issues

If you consider each sensor and emitter a separate device, the number of directional devices involved in rendering an object invisible might be very high. In fact, the object surface must be densely filled with sensors or emitters oriented toward all possible directions of the object's outer space.

To provide some quantitative data, we can reapply considerations made previously to a 1-m-diameter sphere. Consider again for a moment a fixed point of observation; in this case, to render invisibility at a 10 m distance, approximately 372,000 devices, each 8.44 mm^2 wide, must cover the sphere's surface. We derive this number from the Listing-Donders model: from a 10 m distance, a device of 8.44 mm^2 is seen as a single point, and we can easily calculate that a 1-m-diameter sphere has room for 372,000 such devices. When you have to preserve such a property for any direction of observations, however, the 8.44 mm^2 previously occupied by a single device must now include numerous sensors and emitters pointing in different directions. As a first approximation, we can say that in creating a smooth 3D view of an object, the number of directional sensors and emitters in the 8.44 mm^2 area must be at least one-half the total number of such areas—that is, 186,000 (372,000/2). This is because, to acquire and display a coherent image from all possible points of view in each of the 8.44 mm^2 areas, you need at least one sensor or emitter to support any of the other sensors and emitters directly visible from a common viewpoint (approximately the number of sensors and emitters in half a surface—for example, a hemisphere).

To fill a 8.44 mm^2 area with 186,000 sensors and with a similar number of emitters, each device should occupy an area smaller than 5 μm ∗ 5 μm. It's hard to imagine a single standalone computer-based device of that size. Still, you can think of packing in multiple optical microdevices pointing in different directions and controlling them via a single compound device. Efforts such as those at Texas Instruments, where they've produced microdisplays made up of electrostatically actuated mirrors of a few μm, and at Philips Research Laboratories,[16] showing the possibility of growing μm-scale LCD cells on any type of surface, demonstrate such an approach's potential feasibility.

For bandwidth requirements, we can apply calculations similar to the ones in the previous section. Coding the ($color, x, y, z, \theta, \varpi$) information requires 104 bits (24 + 5 ∗ 16 bits). Each tuple must carry its extrinsic coordinates to avoid hash collisions; you can dynamically recompute at each hop the intrinsic coordinates required for routing. Considering that, in the worst case, you must route each message for 1.57 m (one-half of a maximum circle in the sphere), an In-Out communication occurring through a rendezvous device would require, in the worst case, 3.14 m—that is, 1,082 hops. (We assume that the wireless-communication range is long enough to transmit over the 8.44 mm^2 area.) So, for a single directional facet, or monodirectional device, the required bandwidth to route 30 messages per second for a maximum worst case of 1,082 other devices is 1,082 ∗ 104 bits ∗ 30 Hz ≈ 3.4 Mbits/sec (although the average required bandwidth for sensors might be lower). If you pack all the devices contained in the 8.44 mm^2 area (372,000 devices) in a single compound device, the devices' bandwidth requirements increases tremendously (to approximately 1.2 Tbits/sec). As challenging as this might be, it's not technically impossible, especially when you consider recent advances in terahertz-band technologies.[9] Additionally, you could also exploit AI and image-recognition technologies to have sensors track observers' positions and reduce the sensors' efforts by rendering invisibility from a limited set of viewpoints.

Applications

Perhaps the most natural applications for invisible objects fall in the military market (for example, invisible cars and tanks) and in the nonintrusive study of natural ecosystems.

You could also paint the interior of an object, such as a room, instead of its exterior. For example, you could produce realistic, immersive virtual reality environments, freeing users from wearing intrusive hardware[17] and from staying immobile at a specific position in a specific room where specific displays are placed.[18] A related application with a great potential market in southern Europe would be to produce paintable windows. Tall buildings, very dark inside due to both narrow windows and narrow streets, characterize southern European cities. Because local laws forbid changing these buildings' structures, painting virtual windows could greatly improve the quality of living there. From the inside, they would look like large, real windows; from the outside, you wouldn't perceive changes to the building.

More generally, you could effectively exploit the outlined technology to produce visibility despite occluding objects. For instance, appropriately painting trucks would improve their rear visibility. Also, you could improve limited visibility in mountain streets by painting portions of the occluding slopes.

Step 3. The invisibility cloak

The last constraint we have to remove to build the cloak of invisibility (or more generally, a comfortable cloth) is rigidity. We need to deploy our network of devices on a flexible fabric, which you can reform accordingly for unpredictable dynamics and shapes, due to factors both external (for example, wind) and internal (for example, the wearer's movements).

Software issues

Unlike the previous cases, a flexible cloak's In-Out pairs must be continuously reestablished to reflect the cloak's movements. In particular, although the device's intrinsic coordinates remain fixed, extrinsic coordinates can change continuously. So, the routing problem becomes particularly challenging, and you must account for the overhead in maintaining an overlay coherent structure over the cloak's amorphous network. We have two strategies for solving this problem.

First, devices could reevaluate their extrinsic coordinates continuously (for example, 30 times per second) to account for cloak reshaping. The communication would then proceed with the rendezvous approach. Unfortunately, this strategy imposes a notable computational and communication overhead on devices, leaving little room for the activities related to image rendering. Also, because the localization process might require several iterations to reach satisfying accuracy, we can't predict whether this process would be fast enough for execution at 30 times per second.

Alternatively, you could rely on a small, rigid portion of the cloak (for example, a belt or a necklace) as a central point for geometrical references. Both In and Out devices, without being forced to know a priori their extrinsic coordinates, could send a partially undefined tuple (containing only the color information) toward the rigid reference point. While being routed from the source toward the cloak's rigid section, each of the intermediate devices could dynamically compute the geometric information related to the path that the tuple followed and include this information in the tuple before propagating it. Once the tuple arrives at the cloak's rigid section, it can exploit the information collected during its travel to discover the source's extrinsic coordinates. At this point, it can apply the hash-based rendezvous mechanism to meet its mate. A drawback is load imbalances, produced by concentrating computational and communication activity in (and in the proximity of) the cloak's rigid area.

We do not exclude that better strategies and algorithmic solutions might exist. You might find inspiration from mobile ad hoc networks, which focus on enabling mobile peers to communicate with each other, despite the networks' continuously changing topology.[2]

Optical and hardware issues

Cloak flexibility does not change the devices' optical and size characteristics. What might change instead is their bandwidth requirement, as induced by the additional communication overhead to support cloak deformations and dynamic reforming of the In-Out pairs.

If we consider the first solution proposed (dynamic recomputing of positions and orientations), optimistically assuming that this solution is computationally feasible, the bandwidth requirements will likely increase dramatically, owing to the relocalization process's iterations. If we consider the second proposed solutions, the bandwidth requirements do not seem to even double. In fact, before a device could route a message to the appropriate rendezvous place, the message must travel toward the rigid reference point (for example, the belt). So, each message on a flexible cloak travels for a total distance that would be less than twice the distance it would have traveled for a rigid object. However, you must carefully check such considerations against the load imbalances introduced.

Additionally, energy recharging can give a mutable, wearable cloak a distinct advantage over a wall or rigid object. Even when lacking an external power supply,

- You can use inertial forces induced by cloak movements to mechanically recharge the devices
- The human body's thermal energy can provide an alternate energy source for wearable computing systems (for example, Infineon Technology offers thermal-body-powered clothes)

A final note relates to the artifact's cost. The Institute for Defense Analysis has predicted that large-scale production of MEMS computer-based systems will reduce their cost, in the near future, to well below 1 Euro each.[8] A cloak of invisibility of 3 m^2 would require approximately 372,000 compound (multidirectional) devices to be invisible at a 10m distance, implying an overall cost well below a half-million Euros.

Applications

Of course, you could move from cloaks to any type of clothing, applying the same technology. They could significantly impact military operations and investigations, although raising questions of ethics. We'd rather see invisible clothes appear in fashion and entertainment markets. For fashion, you could enrich small (and cheap) portions of clothes and accessories (for example, T-shirts, bracelets, or necklaces) with invisibility frames. Moreover, an invisibility cloak could effectively augment virtual reality games in theme parks such as Disneyland.

Another application we have considered, but aren't sure if it's feasible, is internal body monitoring. You could have a patient drink (or be injected with) a set of sensors and let them disperse in a zone of the body (for example, the stomach). Then, via an emitter-based cream that you'd paint on the patient (for example, the belly), you could see the body's interior, despite the internal movements of the body and of the sensors within it. If impossible in a human body, you could be probably do it in other types of "interiors" characterized by dynamic internal activity, such as an underground river or a complex pipeline.

Recent advances in MEMS and wireless communication technologies, together with conceptual advances in distributed coordination, let us envision the possibility of building an invisibility cloak in our lifetimes, possibly accordingly to our conceptual implementation.

Our research group currently faces issues related to distributed coordination in pervasive and mobile computing scenarios, which brings us closer to realization of the invisibility cloak and related applications.

First, we are studying the impact of environmental dynamics on the local states of sets of locally connected sensors and on their global behaviors.[19] We've found that you can effectively exploit environmental dynamics (for example, the type and dynamics of the stimuli received) to achieve a globally organized behavior in a large set of distributed sensors. For an invisibility cloak, you could use this to let In sensors in a region of the cloak perform some sort of dynamic and distributed data compression, to drastically reduce the data traveling over the cloak.

Second, we are studying a novel approach to content-based routing, aimed at letting data flow toward its destination in a complex network by relying on distributed *computational fields*, providing dynamic, application-specific views of a region of physical space.[4] You can then achieve content-based routing by letting data follow the shape of such fields, which will lead them to their destinations despite dynamic environmental changes. For the cloak, you could effectively use such computational fields to dynamically route information from In to Out devices despite the cloak's changes in shape.

These approaches, together with approaches from other areas (for example, amorphous computing, multiagent systems,[20] and sensor networks) will likely improve the cloak's conceptual design and shorten its time to market and related costs.

ACKNOWLEDGMENTS

We thank Tim Kindberg, Roy Want, and the anonymous referees for helpful comments and corrections on this article's earlier drafts.

REFERENCES

1. D. Estrin et al., "Connecting the Physical World with Pervasive Networks," *IEEE Pervasive Computing*, vol. 1, no. 1, Jan.–Mar. 2002, pp. 59–69.

2. J. Broch et al., "A Performance Comparison of MultiHop Wireless Ad Hoc Network Routing Protocols," *Proc. ACM/IEEE Conf. Mobile Computing and Networking*, ACM Press, New York, 1998, pp. 85–97.

3. J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *Computer*, vol. 34, no. 8, Aug. 2001, pp. 57–66.

4. M. Mamei, L. Leonardi, and F. Zambonelli, "A Physically Grounded Approach to Coordinate Movements in a Team," *Proc. 1st Int'l Workshop Mobile Teamwork*, IEEE CS Press, Los Alamitos, Calif., 2002, pp. 373–378.

5. A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," Lecture Notes in Computer Science, no. 2218, Springer-Verlag, Berlin, 2001, pp. 329–350.

6. R. Nagpal, "Programmable Self-Assembly Using Biologically-Inspired Multiagent Control," *Proc. 1st Int'l Conf. Autonomous Agents and Multiagent Systems*, ACM Press, New York, 2002, pp. 418–425.

7. O.C. Thorpe, *Computing Curvature Using Amorphous Computing*, tech. report, MIT Artificial Intelligence Lab, Cambridge, Mass., 1998.

8. K.S.J. Pister, "On the Limits and Applicability of MEMS Technology," Defense Science Study Group Report, Inst. for Defense Analysis, Alexandria, Va., 2000.

9. R. Kohler, "Terahertz Semiconductor-Heterostructure Laser," *Nature*, vol. 417, no, 6885, 9 May 2002, pp. 156–159.

10. J. Lifton et al., "Pushpin Computing System Overview: A Platform for Distributed, Embedded, Ubiquitous Sensor Networks," *Proc. 1st Int'l Pervasive Computing Conf. 2002*, Lecture Notes in Computer Science, no. 2414, Springer-Verlag, Berlin, 2002, pp. 139–151.

11. A. Howard and M.J. Mataric, "Relaxation on a Mesh," *Proc. IEEE/RSJ Int'l Conf. Robots and Systems*, IEEE CS Press, Los Alamitos, Calif., 2001, pp. 740–745.

12. J.F. Fulton, *Howell's Textbook of Physiology*, W.B. Saunders, Philadelphia, 1946.

13. J.M Kahn, R.H. Katz, and K.S.J. Pister, "Emerging Challenges: Mobile Networking for Smart Dust," *J. Comm. and Networks*, vol. 2, no. 3, Sept. 2000, pp. 188–196.

14. N.B. Priyantha et al., "The Cricket Compass for Context-Aware Mobile Applications," *Proc. ACM/IEEE Conf. Mobile Computing and Networking*, ACM Press, New York, 2001, pp. 1–14.

15. I. Stoica et al., "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM Special Interest Group on Communications Conf.* (SIGCOMM 2001), ACM Press, New York, 2001, pp. 149–160.

16. R. Penterman et al., "Single-Substrate LCDs by Photo Enforced Stratification," *Nature*, vol. 417, no. 6684, 2 May 2002, pp. 55–58.

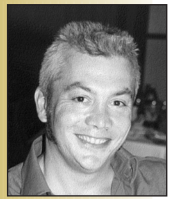
17. W. Piekarski and B. Thomas, "ARQuake: the Outdoor Augmented Reality Gaming System," *Comm. ACM*, vol. 45, no. 1, Jan. 2002, pp. 36–38.

18. J. Jacobsson and Z. Hwang, "Unreal Tournament for Immersive Interactive Theater," *Comm. ACM*, vol. 45, no. 1, Jan. 2002, pp. 39–42.

19. F. Zambonelli, M. Mamei, and A. Roli, "What Can Cellular Automata Tell Us about the Behaviour of Large MultiAgent Systems?" *Proc. 1st Int'l Workshop Software Eng. for Large Agent Systems*, to be published in Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2003.

20. F. Zambonelli and V. Parunak, "From Design to Intentions: Signs of a Revolution," *Proc. 1st Int'l Joint Conf. Autonomous Agents and Multiagent Systems*, ACM Press, New York, 2002, pp. 455–456.

the AUTHORS



Franco Zambonelli is a professor of computer science at the University of Modena and Reggio Emilia. His research interests include distributed and pervasive computing, multiagent systems, and agent-oriented software engineering. He obtained his Laurea in electronic engineering and his PhD in computer science from the University of Bologna. He is a member of the management committee of the European Network of Excellence "Agentlink II" and, within the same network, is the coordinator of the Special Interest Group on Methodologies and Software Engineering for Agent Systems. He is a member of the IEEE, ACM, AIIA (Italian Association for Artificial Intelligence), and TABOO (Italian Association on Object-Oriented Technologies). Contact him at Dipt. di Scienze e Metodi dell'Ingegneria, Univ. di Modena e Reggio Emilia, Via Allegri 13, Reggio Emilia, Italy; franco.zambonelli@unimo.it.

Marco Mamei is PhD student in computer science at the University of Modena and Reggio Emilia. His research interests include distributed and pervasive computing, complex and adaptive systems, and multiagent systems. He has a Laurea in computer science from the University of Modena and Reggio Emilia. He is a member of the AIIA (Italian Association for Artificial Intelligence) and TABOO (Italian Association on Object-Oriented Technologies). Contact him at Dipt. di Scienze e Metodi dell'Ingegneria, Univ. di Modena e Reggio Emilia, Via Allegri 13, Reggio Emilia, Italy; mamei.marco@unimo.it.