Cloudlets: at the Leading Edge of Mobile-Cloud Convergence

(Invited Paper)

Mahadev Satyanarayanan[†], Zhuo Chen[†], Kiryong Ha[†], Wenlu Hu[†], Wolfgang Richter[†], Padmanabhan Pillai[‡]

[†]Carnegie Mellon University and [‡]Intel Labs

***Abstract*—As mobile computing and cloud computing converge, the sensing and interaction capabilities of mobile devices can be seamlessly fused with compute-intensive and data-intensive processing in the cloud. *Cloudlets* are important architectural components in this convergence, representing the middle tier of a mobile device — cloudlet — cloud hierarchy. We show how cloudlets enable a new genre of applications called *cognitive assistance* applications that augment human perception and cognition. We describe a plug-and-play architecture for cognitive assistance, and a proof of concept using Google Glass.**

I. Introduction

"Augmenting Cognition" [20], a thought piece written in 2004, imagined a world with real-time cognitive assistance for mobile users:

> "Looking toward the future, we can envision computing technologies converging in tantalizing ways to augment cognition. For example, imagine a wearable computer with a head-up display in the form of eyeglasses and with a built-in camera for continuous face recognition. This would offer the essentials of an augmented-reality system to aid cognition."

A decade is a long time in computing. When this futuristic world was imagined in 2004, many of the technologies that we take for granted today did not exist. Smartphones, Google Glass, and elastic cloud computing had yet to be invented. What has not changed are human limitations. Already scarce in 2004, human attention is even scarcer today. Mobile real-time cognitive assistance offers a powerful antidote to high-distraction environments. By unobtrusively and spontaneously guiding a user's attention, such a system can reduce the impact of external distractions and help maintain full situational awareness. It can also help with just-in-time learning of new skills, which has already become a meta-skill of enormous value in an ever-changing world.

Today, we have the building blocks of technology needed to achieve our decade-old dream. Available to us are wearable computers, ubiquitous wireless networks, cloud computing resources, and cognitive algorithms that equal or exceed human accuracy and speed in tasks such as computer vision, speech recognition, natural language translation, and question answering. How do we put these together to create systems that can provide real-time cognitive assistance for mobile users? That is the focus of this paper. We describe a system architecture and a proof-of-concept demonstration that point the way to many rapid advances in the coming decade. A critical insight is that cloud computing itself will have to change architecturally in order to support the low-latency resource-intensive computations that occur within the innermost loops of cognitive assistance. The need to support cognitive assistance within very tight time bounds will inspire many innovations in mobile and cloud computing over the next decade. Indeed, real-time cognitive assistance may become the "killer app" that shapes mobile computing in the next decade.

II. Mobile Real-Time Cognitive Assistance

In his visionary essay *"As We May Think,"* Vannevar Bush imagined the existence of a device called a "Memex" that would extend and amplify human thought [5]. Written in 1945, this is the earliest recognition that computing might be harnessed to augment human cognition. Prior to Bush, computing devices had been seen primarily as engines to reduce the drudgery of laborious mathematical calculations. Today, one can view the Internet and the World Wide Web as a collective Memex for society. The Memex was an inspired early answer to the question, *"How can computers help humans be smarter?"* This question assumes renewed urgency today. The improvements to intellectual productivity that began at the dawn of computing and continued through the advent of personal computing, the Internet and the World Wide Web have now plateaued. How will we re-energize and extend our quest for intellectual productivity?

We start by recognizing that the scarcest resource in a computing system is no longer its processor, memory, storage capacity, network bandwidth, or even battery life. Rather, it is *user attention* — the ability of the human user (who is the most essential part of the whole system) to stay focused on the task at hand, ignoring all distractions. Herb Simon's 1971 observation that "...a wealth of information creates a poverty of attention..." [22] is truer today than ever before. Mark Weiser made the same point in a different way in 1991 [24]: "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it." The Aura project [9] had low distraction as its central theme.

This line of thinking leads to the question, *"How do we augment human cognition in a way that is minimally distracting?"* A modern GPS-based car navigation system offers some important clues. You start by giving it high-level information about the destination. From that point onwards, the system requires no babysitting. Occasionally, it offers you helpful just-in-time voice-synthesized guidance about upcoming actions that you need to take. If you ignore a suggestion (e.g., by missing an exit), the system recognizes this promptly and adapts to your behavior by reformulating its route and offering you new guidance. Most of the time, it remains silent but alert. The complex technology needed to achieve this simplicity (e.g., satellites, wireless communication, GPS receiver chips, route planning and optimizing algorithms, and voice synthesis

Ron was wounded in Afghanistan and is slowly recovering from traumatic brain injury. He is often unable to remember the names of friends and relatives. He also forgets to do simple daily tasks. Fortunately, his cognitive assistance system offers hope. When Ron looks at a person for a few seconds, that person's name is whispered in his ear along with additional cues to guide Ron's greeting and interactions; when he looks at his thirsty houseplant, he hears "water me"; when he looks at his dog, he hears "take me out." Ron's cognitive assitance system travels with him, and is always ready to help him.

(a) Disabled Veteran

For Sara, transitioning from her EMT training to the field is a daunting experience. Here she is with her first emergency, a middle-aged woman who has fallen to the ground after being stung by a wasp at a state fair. With her own pulse racing, amidst the distraction of the milling crowd, Sara tries to remember the protocol for allergic reaction when her cognitive assistance system says, "Check for respiratory distress." When Sara responds "present," the voice says, "epinephrine injection device – thigh." Point by point, Sara is led through her protocol until the woman's wheezing begins to subside.

(b) Medical Training

John has been really good on his new diet for three weeks. He has been strongly advised by his doctor to lose weight, because he is on the verge of Type-2 diabetes. Today, at a dinner with friends, John's resolve has been sorely tested but he has been able to resist all temptations so far. Alas, the dessert course is his downfall. When the mouth-watering tray of delicacies is brought before him, he can hold back no longer. As he reaches for the 1200-calorie pecan pie with whipped cream, his cognitive assistance system screams in his ear and stops him cold. John has been able to stay on his diet for another day.

(c) Health and Wellness

On a busy travel day, a People Mover at the airport is down. It is exhibiting jerky motion and behaving in an unsafe way not seen before. Jane, the lead engineer has run out of ideas. Nothing in her extensive experience matches this situation. Sensing her stress, her cognitive asisstance system suggests that she check a recent incident report from an Asian airport with the same People Mover. This proves to be a winning idea. The symptoms appear to be similar, so Jane tries the same fix: reverting a recent software upgrade on the wayside computer. Within minutes the problem is solved and the People Mover is back in service.

(d) Industrial Troubleshooting

Fig. 1. Hypothetical Cognitive Assistance Scenarios

algorithms) are totally invisible to the user. The system has transformed the difficult task of navigating unfamilar terrains into a trivial exercise in following directions.

Can we generalize this metaphor? Can we transform unfamilar and difficult-to-learn tasks in professional and personal settings into simple just-in-time guidance from a system that is tolerant of human errors and limitations? The emergence of wearable computers such as Google Glass is a powerful catalyst and game-changer. By combining the rich sensing capabilities of such devices with the plentiful resources of cloud computing and algorithmic advances in the building blocks of human cognition, we can fundamentally transform the learning of new skills and the reinforcement of existing skills. Figure 1 illustrates the future we are trying to create. In the rest of this paper, we will abbreviate the phrase "mobile real-time cognitive assistance" to just "cognitive assistance."

III. Challenges and the Role of Cloudlets

A. Crisp Interactive Response

Humans are acutely sensitive to delays in the critical path of interaction. This is apparent to anyone who has used a geosynchronous satellite link for a telephone call. The nearly 500 ms round-trip delay is distracting to most users, and leads to frequent conversational errors.

Normal human performance on cognitive tasks is remarkably fast and accurate. Lewis et al. [14] report that even under hostile conditions such as low lighting and deliberately distorted optics, human subjects take less than 700 milliseconds to determine the absence of faces in a scene. For face recognition under normal lighting conditions, experimental results on human subjects by Ramon et al. [18] show that recognition times range from 370 milliseconds for the fastest responses on familiar faces to 620 milliseconds for the slowest response on an unfamiliar face. For speech recognition, Agus et al. [1] report that human subjects recognize short target phrases within 300 to 450 ms, and are able to detect a human voice within a mere 4 ms. Ellis et al. [6] report that virtual reality applications that use head-tracked systems require latencies less than 16 ms to achieve perceptual stability.

More generally, assistive technology that is introduced into the critical paths of perception and cognition should add negligible delay relative to the task-specific human performance figures cited above. End-to-end delays of more than a few tens of milliseconds will distract and annoy a mobile user who is already attention challenged. Note that it is not sufficient to just match human performance in recognition speed. It is necessary to be superhuman in speed without sacrificing accuracy in order to leave time within a very tight budget for additional processing to compose user guidance.

B. Need for Offloading

Cognition is not just latency-sensitive but also compute-intensive and memory-intensive. Tasks such as object recognition and natural language translation require server class hardware today. While wearable devices will continue to improve over time, they will always be resource-poor relative to server hardware of comparable vintage [19]. Figure 2, adapted from Flinn [8], illustrates the consistent large gap in the processing power of typical server and mobile device hardware over a 16-year period. This stubborn gap reflects a fundamental reality of user preferences. To be successful in the marketplace, mobile devices have to exploit Moore's Law very differently from how it is exploited by server class hardware. The most sought-after features of a wearable device are light weight, small size, long battery life, comfort and aesthetics, and tolerable heat dissipation. System capabilities such as processor speed, memory size, and storage capacity are only secondary concerns.

Year	Typical Server Processor	Speed	Typical Handheld or Wearable Device	Speed
1997	Pentium® II	266 MHz	Palm Pilot	16 MHz
2002	Itanium®	1 GHz	Blackberry 5810	133 MHz
2007	Intel® Core™ 2	9.6 GHz (4 cores)	Apple iPhone	412 MHz
2011	Intel® Xeon® X5	32 GHz (2x6 cores)	Samsung Galaxy S2	2.4 GHz (2 cores)
2013	Intel® Xeon® E5	64 GHz (2x12 cores)	Samsung Galaxy S4	6.4 GHz (4 cores)
			Google Glass OMAP 4430	2.4 GHz (2 cores)

Fig. 2. Evolution of Hardware Performance (adapted from Flinn [8])

Response time and battery life can often be improved by offloading the execution of cognitive algorithms from a wearable device over a wireless network to the cloud. The rich sensing capabilities of a mobile device (such as accelerometer, gyroscope, microphone, and camera) can then be combined with compute-intensive or data-intensive cloud processing. The Apple Siri voice recognition system, the Google Goggles augmented reality system, and the Amazon Silk browser are examples of commercial systems that use this approach.

C. Squaring the Circle: How Cloudlets Can Help

It is difficult to simultaneously satisfy the need for crisp, low-latency interaction and the need for offloading processing from a wearable device. The obvious solution of using commercial cloud services (such as Amazon EC2) over a wide-area network (WAN) is unsatisfactory because WAN round trip times (RTTs) are too long. As a user travels, his mobile device experiences high variability in the end-to-end network latency and bandwidth to cloud data centers. In their study of cloud services, Li et al. [15] report an average RTT of 74 milliseconds from 260 global vantage points to their optimal Amazon EC2 instances. The RTT distribution on WANs tends to be heavy-tailed, with many individual round trips taking hundreds to thousands of milliseconds. The situation is unlikely to improve on its own, because commercial service providers are focused on improving network bandwidth for video streaming rather than lowering end-to-end latency. In today's Internet, how do we enable cognitive assistance applications that need cloud resources at consistently low end-to-end latency?

We can introduce a new degree of freedom into this overconstrained design space by using *cloudlets* [21], as shown in Figure 3. A cloudlet is a new architectural element that represents the middle tier of a 3-tier hierarchy: mobile device — cloudlet — cloud. It can be viewed as a "data center in a box" whose goal is to "bring the cloud closer." As a powerful, well-connected and trustworthy cloud proxy that is just one wireless hop away, a cloudlet is the ideal offload site for cognitive assistance. Although widespread commercial deployment of cloudlets is not yet a reality, there is growing commercial investment in cloudlet-like infrastructure. For example, Nokia recently announced the availability of RACS (Radio Access Cloud Platform) [17] for use in 4G cellular systems. A number of companies including Dell, AOL, and Huawei have introduced hardware for *micro data centers* [16] that could easily be repurposed as cloudlets.

We envision cloudlets being used as follows. The user's cognitive assistance device (such as Glass) discovers and associates with a nearby cloudlet. It then uses this cloudlet for all offload processing. Optionally, the cloudlet may reach out to the cloud for various services such as centralized error reporting, usage logging or prefetching of data. However, all such cloudlet-cloud interactions are outside the critical latency-sensitive path of device-cloudlet interactions. As the mobile user departs from the proximity of this cloudlet, a mechanism analogous to cellular handoff is invoked. This seamlessly associates the user with another cloudlet for further travel.

IV. The Plug and Play Gabriel Platform

The domain-specific knowledge and software development effort involved in creating and validating an implementation of a cognitive algorithm are large. There is rapid innovation in this space by many different teams worldwide, often motivated by use cases unrelated to cognitive assistance. Many algorithm designs and implementations have been developed and refined over the years for tasks such as face recognition [25], activity recognition in video [23], natural language translation [2], OCR [10], and question-answering technology [7]. These *cognitive engines* are written in a variety of programming languages and use diverse runtime systems. Some of them are proprietary, some are written for closed source operating systems, and some use proprietary optimizing compilers. Each cognitive engine typically represents many tens to hundreds of person years of effort by experts in that domain. To the maximum extent possible, we would like to reuse this large existing investment without any rewriting of software, or imposition of rigid standards. We would also like to stimulate the creation of new cognitive engines that leverage new results or new software components. In colloquial terms, we would like to support "plug and play" simplicity in reusing existing cognitive engines or adding new cognitive engines.

To address these requirements, we have created an open source platform for cognitive assistance called *Gabriel* that runs on cloudlets. Gabriel uses virtual machine (VM) encapsulation of cognitive engines and interconnects these VMs using a publish-subscribe (PubSub) mechanism for efficient sharing of sensor data that is streamed from a wearable device. Figure 4 illustrates Gabriel's back-end processing structure on a cloudlet. An ensemble of *cognitive VMs*, each encapsulating a different cognitive engine, independently processes the incoming flow of sensor data from a Glass device. A single *control VM* is responsible for all interactions with the Glass device. The sensor streams sent by the device are received and preprocessed by this VM. For example, the decoding of compressed images to raw frames is performed by a process in the control VM. This avoids duplicate decoding within each cognitive VM. A PubSub mechanism distributes sensor streams to cognitive VMs. At startup, each VM discovers the sensor streams of interest through a UPnP discovery mechanism.

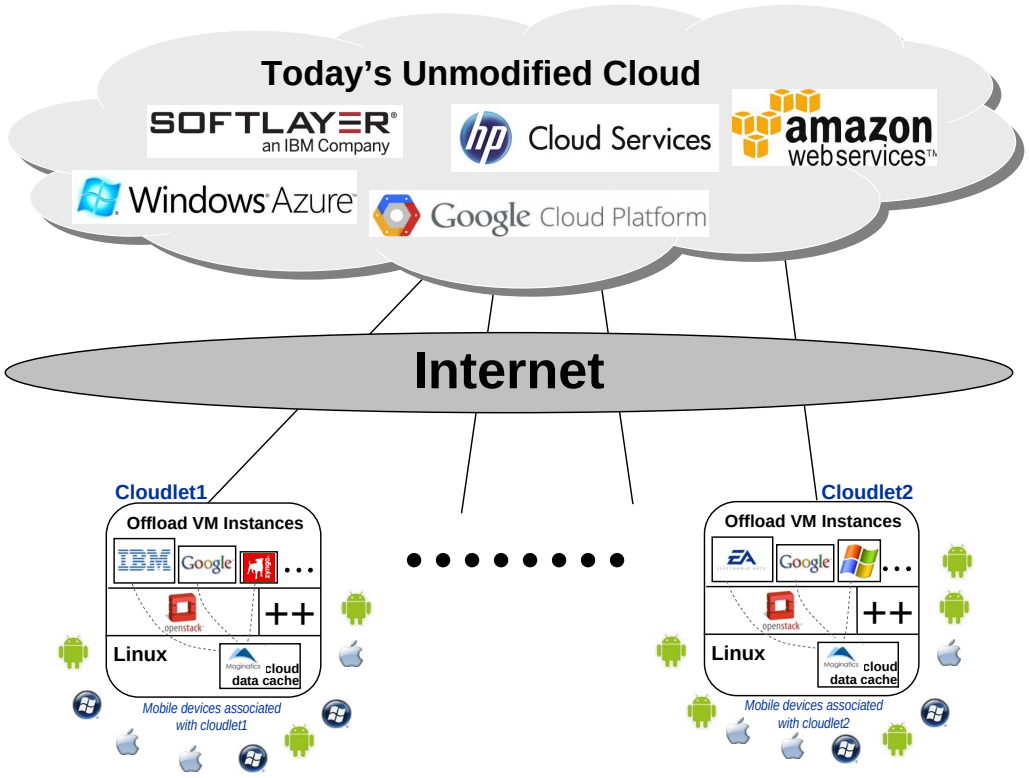


Fig. 3. Two-Level Cloud-Cloudlet Architecture

The outputs of the cognitive VMs are sent to a single *User Guidance VM* that integrates these outputs and performs higher-level cognitive processing. From time to time, this processing triggers output for user assistance. For example, in an assistive application for Alzheimer's patients, a synthesized voice may say the name of a person whose face appears in the Glass device's camera. It may also convey additional guidance for how the user should respond, such as "John Smith is trying to say hello to you. Shake his hand." As Gabriel evolves, we envision significant improvement in user experience to come from sophisticated, higher-level cognitive processing in the User Guidance VM.

Context-sensitive control of sensors on the Glass device is achieved through a context inference module in the Control VM. Significant improvements in battery life and usability are possible if high-level knowledge of user context is used to control sensors on a wearable device. For example, consider a user who falls asleep in his chair at home while watching TV. While he is asleep, his Glass device does not have to capture video and stream it for cognitive assistance. In fact, offering whispered hints during his nap might wake him up and annoy him. When he wakes up of his own accord, cognitive assistance should resume promptly. The challenge is, of course, to reliably distinguish between the user's sleeping and waking states. This is the classic problem of activity recognition from body-worn sensor data, on which significant progress has been made in the recent past. These results can be extended to infer context and then use it for adaptive control of sensor streams.

This problem has many subtle aspects. In the above example, if the user falls asleep in a bus or metro rather than in his living room, the cognitive assistance system should give him timely warning of his approaching exit even if it wakes him up from his pleasant nap. In this case, location sensing will need to remain turned on during his nap even though other sensors (such as the camera) may be turned off. More generally, tight coupling of sensing and context inference in a feedback loop is valuable in a cognitive assistance system.

V. Proof of Concept: 2D Lego Assembly

To gain initial validation of the cognitive assistance concept and to obtain first-hand experience with its implementation challenges, we have built a full end-to-end prototype of a very simple cognitive assistance application. Simplicity was essential for keeping the implementation effort of this proof of concept modest. Our application uses Google Glass and the Gabriel architecture to guide a user in assembling 2D models (such as those shown in Figure 5) using the Lego product *Life of George* [13]. In addition to colored Lego blocks, the product also includes a board with a special black dot pattern and a color palette, as shown in Figure 6. Our processing uses the dot pattern, but does not use the color palette. The restriction to two dimensions simplifies the computer vision aspects of this application. A YouTube video of our implementation can be found at `http://youtu.be/uy17Hz5xvmY`.

Video from the camera on the Glass device is streamed to the cloudlet. Processing of video frames and creation of guidance are done entirely on the cloudlet, as shown by the workflow in Figure 7. There are two major phases in the workflow. In the first phase, the current video frame is analyzed to extract a *symbolic representation* of the current state of the Lego task. This phase has to be tolerant of considerable real-world variation in the image due to variable lighting

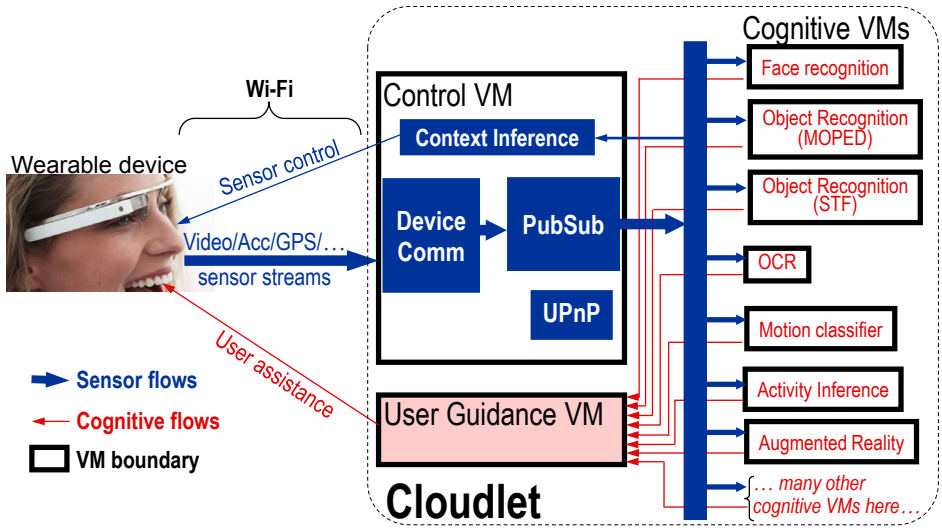


Fig. 4. The Gabriel Architecture for Cognitive Assistance (Source: Ha et al [11])

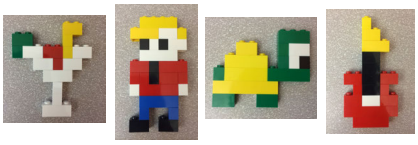


Fig. 5. Example Lego Models from *Life of George*

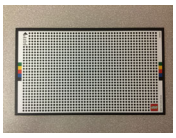


Fig. 6. Empty Board

levels, varying light sources, varying positions of the viewer with respect to the board, task-unrelated clutter in the image background, and so on. The symbolic representation is an idealized representation of the input image relative to the Lego task and excludes all irrelevant detail. One can view this phase as a task-specific "analog-to-digital" conversion of an input image — the enormous state space of the input image is simplified to the much smaller state space of the symbolic representation. Technically, of course, all processing is digital.

The second phase operates exclusively on the symbolic representation. After extraction, the symbolic representaton is compared to the expected task state. Based on this comparison, user guidance for making incremental progress on the task is generated. This guidance has video and plain text components that are streamed back to the Glass device. The video guidance is shown on the Glass display, and accompanying audio guidance is provided using the Android text-to-speech API. We provide more detail on the two phases in the sections below.

A. Extracting the Symbolic Representation

As shown by the example in Figure 8(m), we use a two-dimensional matrix to represent the abstract state of the Lego assembly task. Each matrix element is a small integer that represents the color of the corresponding Lego brick. Values from one to six map to different colors of Lego bricks (black, white, red, green, blue, and yellow). A value of zero indicates the background (i.e., no brick is in that position).

Although based on well-known computer vision techniques, our implementation for extracting the symbolic representation is nontrivial. It relies on the Lego board, whose distinctive black border and black dot pattern (Figure 6) make it easier to detect a Lego model that is placed on it. Our algorithm recognizes the Lego model only if it is placed on the board. As shown on the left side of Figure 7, the implementation consists of a 3-stage pipeline: board detector, Lego localizer, and the generator of the matrix that corresponds to the symbolic representation (which we abbreviate to "matrix generator"). The pipeline is implemented using the OpenCV image processing library, and is described below using Figure 8(a) as a working example.

1) Board detector: The first step in the pipeline is to find the board. As shown in Figure 6, the board has a black border and black dots on its surface. We assume this pattern to be unique and use a black detector to find such an area in the input image. A black, roughly rectangular shape close to this area is then considered to be the board. We verify the density of black dots to make sure that the board is correctly detected.

Reliable implementation of a black detector is harder than it may seem. Because of variation in lighting conditions, the absolute color values of the pixels may vary considerably. A simple threshold on brightness and saturation fails to be robust. Our implementation subtracts the original image by a blurred version of the image. The result is essentially relative brightness of each pixel compared to its neighbors. We then apply a threshold on this relative brightness value. This approach gives us a robust black detector (Figure 8(b)).

After we have an accurate boundary of the board (Figure 8(c)), we find the four corners by doing line detection of the board contour (Figure 8(d)) and intersection of the four lines. We then perform perspective transformation to restore the rectangular shape of the board (Figure 8(e)). The resulting image is the basis of all further processing.

2) Lego Localizer: We next extract the Lego model from its background. Our algorithm is based on the observation that the Lego image has far fewer edges than the background. We perform an edge detection on the whole board image (Figure 8(f)). Following this by dilations and erosions results in an image from which we can simply extract the Lego image by finding the largest blob near the center of the board (Figure 8(g)).

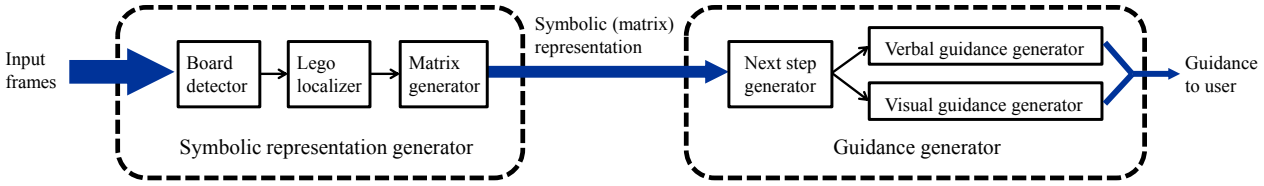


Fig. 7. 2D Lego Task Workflow on Cloudlet

Unfortunately, sole reliance on edge detection is not robust. The sides of Lego bricks (especially the strongly colored ones such as red, yellow and blue) have more texture than the surface, leading to frequent errors. To correct for this, we use color detection to find these sides, and add them to the Lego shape obtained by edge detection. This leads to a shape with both surface and side parts included (Figure 8(h)). We then perform erosion to remove the side parts (Figure 8(i)), with the amount of erosion being calculated from the perspective transform matrix.

Our color detection mechanism converts the image to HSV color space, and then thresholds the individual channels. However, lighting conditions again complicate the processing. Simple thresholds on HSV values are not robust, especially when the incident light is strongly colored (e.g. sunlight through a stained glass window). We therefore use the grey world color normalization method [3], [4] to correct the white balance of the image before doing the color detection.

In addition to the main technique described above, we also apply other techniques such as removing all areas close to black dots. These tend to perform worse than the main technique, but the combination of all of the techniques tends to produce robust results.

3) Matrix Generator: At this point in the processing, we have found all the pixels associated with the Lego model. Before we generate a symbolic representation, we have to perform a final set of corrections. We first rotate the image to an upright orientation (Figure 8(j)). The degree of rotation is calculated by line detection of the Lego image and a majority vote on the line directions.

Using color detection on the color-normalized Lego model, each pixel is quantized to one of the Lego brick colors (Figure 8(k)). The magenta color represents uncertainty. We then partition the image into blocks, with each block corresponding to a 1x1 sized Lego brick (Figure 8(l)). Final assignment of brick color is done by a weighted majority vote of colors within the block, with pixels near the block center being assigned more weight. Figure 8(m) shows the final matrix representation. Figure 8(n) is synthesized from this matrix, and is thus a visualization of the symbolic representation.

B. Generating Guidance

A task in our system is represented as a linked list of Lego states, starting from the beginning state (nothing) to the target state (the user's goal). In this initial prototype, the state list is manually generated for each Lego model supported by our application. Figure 9 shows an example of a task, with each state being illustrated by the synthesized image of the matrix that represents that state.

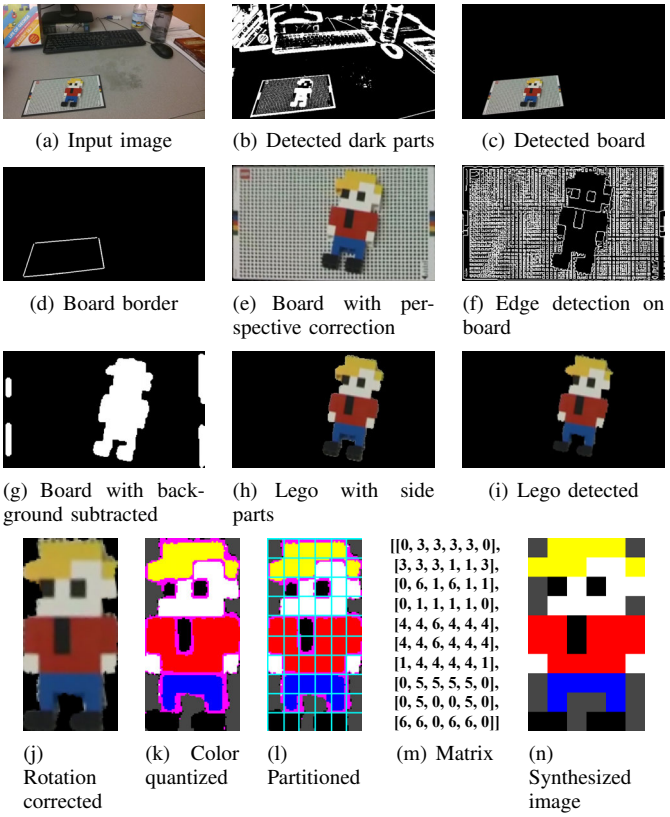


Fig. 8. Workflow for Extracting Symbolic Representation

Based on the matrix representation of the current Lego state, our system tries to find an optimal next step towards the task target. A naive way of doing this is to compare the current state to all the states in the task. If there is a match, then we simply tell the user what the next state should be. Based on the linked list representation, we can easily give users step-by-step instructions as he follows the guidance. However, if the user fails to follow the instructions (that is, the user's current state is not in the pre-defined state list) the system will not be able to give any useful guidance. Our implementation tries to be smarter in such cases. Guidance generation consists of four logical steps:

1) We check if any state in the task representation is strictly one brick more than the current user state. If so, we ask the user to add the brick to the existing Lego model; otherwise go to step 2.
2) We check if the current user state can match any state in the task representation by moving an existing brick. If so, we ask the user to move the brick, otherwise go to step 3.
3) We check if any state in the task representation is strictly one brick less than the current user state. If

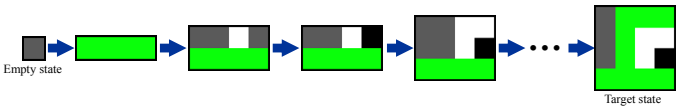


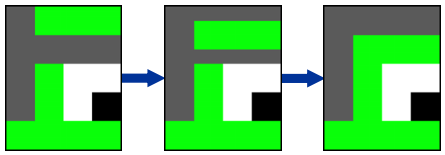
Fig. 9. Example State List Representation of Task

"As a first step, please find a 1x4 green piece and put it on the board."

"Great! Now find a 1x2 piece and put it on the top left of the current model."

"You are quite close. Now slightly move the 1x1 yellow piece to the right by 2 brick sizes."

(a) Verbal Guidance



(b) Visual Guidance

Fig. 10. Example of Guidance Provided to User

so we ask the user to remove the brick; otherwise go to step 4.

4) At this step, the system does not know exactly what brick to ask the user to add, move, or remove, which means the user has done something quite different from our instructions. We will then ask the user to revert back to a state that he has built earlier, and go through the steps 1-3 again.

Generally, the instruction to the user consists of three parts, the action the user needs to perform (add, move, or remove), the location information about the action (e.g. where to add the brick or which one to move), and the description of the brick (e.g. size and color). We use both verbal and visual guidance. The verbal guidance is a complete instruction whispered into the user's ears through text-to-speech technology. The visual guidance is an animation displayed on the Glass screen, showing how the bricks should be placed. Figure 10 shows examples of both verbal and visual guidance.

C. Prototype Status

We have implemented the whole pipeline described above in our prototype. The computer vision processing is usually robust under a wide range of lighting conditions. However, it is not perfect especially under very strong light or when the Lego shape is too complex. Currently we require two consecutive video frames from Glass to generate the same Lego matrix before we attempt to generate guidance. This increases our confidence that the extraction of the symbolic representation is correct, but comes at the cost of increased latency.

Our implementation uses a Dell OptiPlex 9010 (4-core 3.4 GHz Intel® Core™ i7 processor and 32 GB memory) as a cloudlet. Since our initial focus has been on creating a fully functional proof-of-concept, we have made no effort to optimize performance. On the unoptimized cloudlet pipeline, it takes between one-half and one second to process one video frame from Glass. If we discover that a frame is defective (e.g. the scene is too blurry, or the Lego board is occluded), we discard it as early as possible in the pipeline and start to process the next frame. Optimizing the cloudlet pipeline and improving the end-to-end responsiveness of the system are important next steps in this work.

D. Future Work

To the best of our knowledge, the 2D Lego application described here is the first working example of a cognitive assistance system. While the application itself is simplistic, its implementation demonstrates the feasibility of the concepts discussed in this paper. It shows that cognitive assistance is not some distant and ethereal vision, but one that can be made a reality in the near future.

The current implementation can be improved in many ways, and these improvements will be our near term focus. The processing pipeline in the cloudlet offers ample scope for optimization. Bringing the processing latency down from the current 0.5-1.0 second to tens of milliseconds will enable a much more fluid and responsive user experience. The complexity of the models and the sophistication of the guidance can both be increased, while still remaining within the 2D Lego domain. This will give us more experience in extracting task-specific symbolic representations from video streams, which is likely to be a very important long-term capability.

Beyond these early improvements, we can expand the sophistication of the target application and include a wider range of sensors. For example, the microphone and the accelerometer provide additional sensor streams that can be used to enrich and possibly simplify the analysis of the video stream from the camera. The metaphor of assembling a target object from a kit of parts has very broad applicability. For example, manufacturers such as IKEA provide furniture kits and expect their customers to perform the assembly from printed directions. This is often an error-prone and frustrating experience for the customer, and sometimes requires troubleshooting via telephone hotlines. Videos on how to perform the assembly can help, but a far superior approach would be to provide a kit-specific cognitive assistance application that provides step-by-step guidance with prompt error detection and correction, as described in this paper. Many other everyday human activities, such as cooking from recipes and repairing home appliances, can also benefit from this kind of step-by-step guidance based on analyzing the actual state of progress towards task completion. Further afield, and necessarily further out into the future, come scenarios such as those of Figure 1.

A major catalyst and accelerator would be the emergence of a marketplace such as the iTunes store or the Google Play store for cognitive assistive applications. Each such cognitive assistive application would need to provide task-specific extraction of symbolic representations from sensed data (e.g. task-specific computer vision) and task-specific user guidance (including detection and correction of error states). By providing a simple way for entrepreneurs to easily monetize effort, such a marketplace would lead to the emergence of a rich ecosystem for cognitive assistance. The open-source Gabriel architecture of Figure 4, layered on top of the open cloudlet architecture of Figure 3, can serve as the foundation for such an ecosystem.

VI. Conclusion

Mobile real-time cognitive assistance rests on three pillars. First, it relies on mobile-cloud convergence embodied in cloudlets for low-latency, compute-intensive processing. Second, using techniques similar to those used by robotic systems, it leverages real-time sensor stream analysis and fusion to extract symbolic representations of the real world. Third, in a manner reminiscent of the learning sciences [12], it generates task-specific user guidance from symbolic representations of task state and deep task knowledge. The pursuit of cognitive assistance will force each of these pillars to be strengthened, thus driving much new research.

As discussed in Sections I and II, user distraction is a growing concern today. The type of cognitive assistance system that we have described does not reduce external distractions. Rather, it overcomes the consequences of distraction relative to a specific task by proactively guiding the user, and by discovering and correcting errors before they have propagated too far. In this sense, the analogy to GPS-based navigation is apt. We may not be able to reduce distraction, but we can reduce its impact. Given this goal, providing step-by-step guidance without any omissions is important. However, one can envision a more ambitious type of cognitive assistance system whose goal is to foster learning of task-specific skills. Such a system would have to be significantly more sophisticated in its task guidance than the kind of system we have described here. To encourage learning, it may be necessary to supress much guidance as a user gains experience on a task. It may also be necessary to allow a user to recover from errors on his own, and to offer help only when he fails to make progress. Implementing such a system will require deep understanding of human cognition and learning, and can be viewed as a type of *cognitive tutor* in the parlance of the learning sciences [12]. Until now, all cognitive tutors have focused on symbolic tasks such as arithmetic or algebra. They have completely avoided the difficult problem of extracting a task-specific symbolic representation in real time from sensor inputs. The latency and processing requirements for solving this problem, combined with the additional processing requirements for cognitive tutoring, make cloudlets an absolute necessity. Thus, cloudlets are truly the leading edge of mobile-cloud convergence — they are the enablers of an entirely new class of mobile computing systems that will shape our future.

Early in this paper we posed the rhetorical question *"How can computers help humans be smarter?"* Every few decades, a fundamentally new approach to answering this question arises. Offering cognitive assistance to mobile users by transforming difficult tasks into simple step-by-step guidance is the newest of these approaches. It is the potential for enormous societal and commercial benefits from this approach that lead us to predict that cognitive assistance will be the "killer app" for mobile computing in the next decade.

Acknowledgements

We wish to thank Bobby Klatzky and Dan Siewiorek for the scenarios in Figure 1(b) and Figure 1(d) respectively. Rahul Sukthankar, Jan Harkes, and Benjamin Gilbert helped us in designing the computer vision processing described in Section V. Some material in Sections III and IV have been reused from our paper "Towards Wearable Cognitive Assistance" [11].

This research was supported by the National Science Foundation (NSF) under grant number IIS-1065336, by an Intel Science and Technology Center grant, by DARPA Contract No. FA8650-11-C-7190, and by the Department of Defense (DoD) under Contract No. FA8721-05-C-0003 for the operation of the Software Engineering Institute (SEI), a federally funded research and development center. This material has been approved for public release and unlimited distribution (DM-0000276). Additional support for cloudlet-related research was provided by IBM, Google, Bosch, and Vodafone. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and should not be attributed to their employers or funding sources.

References

[1] T. Agus, C. Suied, S. Thorpe, and D. Pressnitzer. Characteristics of human voice processing. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, Paris, France, June 2010.

[2] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, Mar. 1996.

[3] G. Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin Institute*, 310(1):1–26, July 1980.

[4] J. M. Buenaposada and B. Luis. Variations of Grey World for face tracking. In *Image Processing & Communications 7*, 2001.

[5] V. Bush. As We May Think. *The Atlantic*, July 1945. http://www.theatlantic.com/magazine/archive/1945/07/as-we-may-think/303881/.

[6] S. R. Ellis, K. Mania, B. D. Adelstein, and M. I. Hill. Generalizeability of Latency Detection in a Variety of Virtual Environments. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 48, 2004.

[7] D. Ferrucci, A. Levas, S. Bagchi, D. Gondek, and E. T. Mueller. Watson: Beyond jeopardy! *Artificial Intelligence*, 199:93–105, 2013.

[8] J. Flinn. *Cyber Foraging: Bridging Mobile and Cloud Computing via Opportunistic Offload*. Morgan & Claypool Publishers, 2012.

[9] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste. Project Aura: Towards Distraction-Free Pervasive Computing. *IEEE Pervasive Computing*, 1(2), April 2002.

[10] Google. tesseract-ocr. http://code.google.com/p/tesseract-ocr/.

[11] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan. Towards Wearable Cognitive Assistance. In *Proceedings of MobSys 2014*, Bretton Woods, NH, June 2014.

[12] K. Koedinger and A. Corbett. Cognitive Tutors: Technology bringing learning science to the classroom. In K. Sawyer, editor, *The Cambridge Handbook of the Learning Sciences*, pages 61–78. Cambridge University Press, 2006.

[13] Lego. Life of George. http://george.lego.com/, October 2011.

[14] M. B. Lewis and A. J. Edmonds. Face detection: Mapping human performance. *Perception*, 32:903–920, 2003.

[15] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: comparing public cloud providers. In *Proceedings of the 10th annual conference on Internet measurement*, 2010.

[16] D. McGrath and A. Banathy. What Can A Micro Data Center Do For You? *The Journal from Rockwell Automation and Our PartnerNetwork*, October 2012. www.rockwellautomation.com/rockwellautomation/news/the-journal/.

[17] Nokia Solutions and Networks (NSN). *Liquid Applications System Description*, July 2014. DN09139526.

[18] M. Ramon, S. Caharel, and B. Rossion. The speed of recognition of personally familiar faces. *Perception*, 40(4):437–449, 2011.

[19] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, Ottawa, Canada, 1996.

[20] M. Satyanarayanan. Augmenting Cognition. *IEEE Pervasive Computing*, 3(2):4–5, April-June 2004.

[21] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, 8(4), October-December 2009.

[22] H. A. Simon. Designing Organizations for an Information-Rich World. In Martin Greenberger, editor, *Computers, Communication, and the Public Interest*. The Johns Hopkins Press, Baltimore, MD, 1971.

[23] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241, 2011.

[24] M. Weiser. The Computer for the 21st Century. *Scientific American*, September 1991.

[25] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face Recognition: A Literature Survey. *ACM Computing Surveys*, 35(4):399–458, December 2003.