Re-Live the moment - Using run visualizations to provide a positive feedback loop for workouts

UROP 2014 – Project Proposal

This project aims to help people revisit their positive experiences in order to bring about behavior change. In particular, we are planning to focus on scenarios of people doing sports/running.

We all are aware of the benefits of doing sport in general, and weekly running in particular, but people often have a hard time "fitting" this into their busy schedules - especially since they are often tired from daily stress. The aim of this project is to help one remember the fun they had during their last run, in particular the post-run satisfaction they felt. The basic idea is to record audio-visual "impressions" of the run and later play them back to the user in order to encourage the next run.

The technical setup is planned like this: a body-worn miniature camera will take pictures and/or record videos while running. A custom-designed app installed on the user's smartphone will allow the user to record the basic run parameters (route, distance covered, speed) while listening to music. In addition, the app will also occasionally prompt the user through a headset on how the person feels, with feedback coming in through the microphone and/or a simple body worn feedback device (e.g., a simple custom-built button feedback). The collected data will be (manually) processed into a short video and then played back to the user at a later time in order to make him/her remember the fun of the overall experience.

The student should create an Android App that supports basic run monitoring capabilities (in the long run, this could be extended to cover other exercises). Upon starting a running session, the App should be configured to record the start time, GPS location data, data from the accelerometer sensor, and the names of the songs being played. Periodically, it should trigger audio prompts on how the person feels. Last but not least, at the end of each running session, the App should show up a pre-configured questionnaire to enable the user assess the session, potentially giving an audio-visual feedback in the form of a "selfie" video.

The App should then be deployed and tested with a small number of volunteers over the summer, in order to prepare for a large scale deployment in September.

Optional tasks:
- The student can also make use of the Speech Recognition interface in Android and try to recognize the recorded audio feedback from the user and identify how the user was feeling e.g. bad, good or excited.
- The App should also allow the user to share some running performance statistics (e.g. total distance, average speed, songs listened) in social networks.
- We can also retrieve the feedback that the user will get from his/her friends after publishing the performance in social networks and try to use it when making him/her revisit the fun experience of running.

Required skills:
Good programming knowledge in Android, or strong Java skills and willingness to learn Android Programming. Knowledge in using the API of social network services is a plus and beneficial for the second optional task.

Professor: Marc Langheinrich
TAs: Bexheti Agon and Niforatos Evangelos
Imagine you have just arrived in a new city. What will your most immediate information needs be? Perhaps a map of the city's subway system, tips of where to go for a quick bite to eat or recommendations of what to see and do. This project aims to investigate the possibility of pre-empting these information needs so that by the time the traveler arrives, the most appropriate search results will already have been sent to their mobile device.

With the increase in use of mobile devices in recent years - and with it a commensurate growth in the number of mobile searches - there is now a large amount of information available describing what people search for and where. Furthermore the ever-increasing popularity of microblogging, particularly when using a mobile device, provides another potential source of geocoded data to mine.

We envision a system which first detects when a user is issuing queries or tweeting away from home, i.e. where the GPS locations associated with the user's logs suddenly appear in a new city, particularly when this is far from their "home base." By identifying a large number of such queries for each location we can construct lists of useful URLs, Tweets and searches for each location. In a final step, the suggested resource list could be improved by considering the interests of the user and re-ranking them by taking these interests into account. We have access to a large corpus of geo-tagged mobile search requests that might form an initial starting point for exploring "away from home" queries.

This is no doubt a difficult task and as such we envision this project to be a first approach towards this problem, perhaps even an investigation into the feasibility of this idea without the production of a working system.

**Required skills:**
Good background in information retrieval, or willingness to learn.

**Professor:** Marc Langheinrich

**TAs:** Bexheti Agon and Niforatos Evangelos
DejaVu

UROP 2014 – Project Proposal

In today's fast paced world, the ability to recall life memories progressively deteriorates as new ones constantly accumulate. A sizeable body of research is dedicated on how memories are encoded and stored and how they can be invoked. Visual stimuli are found to enhance memory recall and are often associated with rich recollections.

In the DejaVu project we want to examine how visual recordings impact the ability to recall memorable events. One application area may be in the context of "behavior change" in order to remember the benefits of desirable behavior. For example, a user who wants to eat healthier may take short video recordings after having had healthy and unhealthy food. The student should develop a simple app that would allow for the recording of short (e.g., up to 6-seconds) video segments, similar to apps such as "Vine" or "Cinemagram". This should be integrated with a simple experience sampling (ESM) interface, where the user can with a simple click mark his or her experience or mood (e.g., "I feel great!", "I feel a bit sick").

A corresponding real-world experiment should then be drawn up in order to investigate the effects of remembering "feel good" vs "feel bad" events, in particular with a view to altering behavior and influencing desires (e.g., for healthy food). The short videos would be actively reviewed by the user, or played back in an ambient, at the regular intervals (e.g., at the end of the day or a week). The app may furthermore record location information and hence incorporate maps into the playback experience.

The project offers students the chance to perform real-world experiments in a multi-national, multi-disciplinary team within the context of the EU-funded RECALL project. Students are expected to play an active role during all stages of the project, i.e., app development, participant recruitment, study planning, and study execution. Mobile application development experience is a strong plus.

Professor: Marc Langheinrich

TAs: Bexheti Agon and Niforatos Evangelos
DejaTweet

UROP 2014 – Project Proposal

Twitter is incredibly popular and is becoming increasingly embedded in everyday life. Users often post about what they are currently reading, thinking and doing and research has shown that they often try to re-find these Tweets. However, due to the brevity of tweets and the sheer number created every day, this is in many cases an extremely difficult task.

Here we envision an app which records as much information as possible about the environment surrounding the user when a tweet is sent. Then when the user goes to relive a moment in the past via their tweets, we can bring this extra information to bear to both make the refinding processes easier but also to deliver a more interesting experience, possibly leading to a more complete and useful recall of the event.

The student should develop a simple Android or iOS app allowing the user to post a tweet and recording some video/audio at the same time. The recorded video could then be uploaded to a service such as YouTube or Vine and then linked with the Tweet to facilitate future recall. Additional information such as the location of the user can be obtained via appropriate use of the Twitter API. In a second step, a web service could be developed to view and manage the recorded data, after it has been linked to the appropriate tweets.

Required skills:
Good existing knowledge of Android/iOS application development, or strong Java skills and willingness to learn Android Programming. Knowledge in using the API of social network services, particularly Twitter, is a plus and beneficial for the second task.

Professor: Marc Langheinrich

TAs: Bexheti Agon and Niforatos Evangelos
Innovative compiler technologies for accessible massive parallelism in numerical simulation software

Dmitry N. Mikushin, Prof. Olaf Schenk

The KernelGen compiler being developed at the University of Lugano automatically transforms computationally intensive parts of the program for running on specialized accelerators – Graphics Processing Units (GPUs) or Many Integrated Cores processors (MICs). This method allows a scientist to develop his model code as a single body, concentrating on problem subject, instead of spending most of the time doing correct and efficient manual parallelization (Fig. 1). Given that the growth of energy-efficient raw compute power is nowadays largely provided by accelerators, KernelGen's important mission is to make their potential more accessible for rapid deployment of new experimental models and multi-million lines of legacy programs.

The technology engine of KernelGen is based on parts of two major open-source compilers – GCC and LLVM, extended by patches, method-specific code transformations and runtime library. All work is performed at the level of compiler's internal virtual assembler language (Fig. 3), which serves as the universal representation for all human-readable programming languages. More details can be found in [2] and [4].

The primary goal of this project is to make a reasonable contribution to one of the compiler development & testing tasks presented below.

The student's primary outcomes: personal track record in compiler development, and connections with compiler engineers from major software companies that use LLVM (Apple, ARM, Google, Intel, NVIDIA, Samsung, Sony, etc.).

1. Implement loop naming directive in C/C++ and Fortran and use it as internal GPU kernel name: learn

---

**Figure 1:** Transformation of single-body program into CPU and GPU parts performed by compiler
how high-level language is lowered to compiler IR and attach name to the loop representation.

```c
#pragma kernelgen loop(name)
for (int i = 0; i < n; i++) { ... }
```

This feature is needed for making stable (not dependent on line or basic block number) loops names for comparing performance reports against the reference output in the automated test suite.

2. Introduce basic implementation of SASS backend: learn how LLVM backends are designed and implement lowering of LLVM IR into native GPU assembly (SASS).

```c
MOV R1, c[0x0][0x44];
S2R R0, SR_CTAID.Z;
MOV R2, c[0x0][0x148];
S2R R3, SR_TID.Z;
IADD R27, R2, -0x2;
IMAD R0, R0, c[0x0][0x30], R3;
IADD R16, R0, 0x2;
```

This component will allow to bypass NVIDIA's PTX-to-SASS assembler – the only proprietary component used by kernelGen, and potentially to make additional optimizations to GPU kernels.

3. Evaluate the quality of basic multigrid [1] parallelization with KernelGen: learn how KernelGen analyses and transforms the code, track down the reasons of issues that may arise and try to fix them. In case of success, write an equivalent CUDA version and compare their performance.

4. Pick up some open problems from KernelGen's tasks and bugs [3].

Where to start

- Read the basic information about LLVM and try to build a working compiler on your local machine.
- Familiarize yourself with LLVM IR language and how to write an IR transformation pass. Write your own pass modifying something in the input IR.
- Specifically for task 2: Writing an LLVM Backend

References


