Lab 5 – Algorithms

To: Professor Mike Boctor
May 5th, 2016

By:
Group 5
Evan Kennedy
Brian Magnusen
Punleuk Oum

Algorithm 1 – Web Worker Continuously Polls for Large Requests

```
api('stitch', { sid: sid, timestamp: now, type: type }, function(data, xhr) {
  if(data.Location) {
    postMessage({'stage' : 'waiting', 'progress' : 'Loading Panorama...'});
    postMessage({'stage' : 'done', 'type' : type, 'result' : data.Location});
    return;
  } else if (data.errorMessage) {
    console.error(data);
    var prettyError = JSON.stringify(JSON.parse(data.errorMessage),null,'\t');
    postMessage({'stage' : 'fail', 'message' : ['Couldn\'t stitch:', prettyError].join('\n')});
    return;
  }

  var timeout = Date.now() + 300000; //300 seconds elapsed
  var len = 0;
  async.whilst(
      function () {
        return len === 0 && Date.now() < timeout;
      },
      function (callback) {
          setTimeout(
          api('user/list-user-files', {sid: sid, timestamp: now, type: type}, function (list, xhr) {
            var response = JSON.parse(xhr.response);
            if(response.errorMessage){
            callback(response);
            } else {
              len = list.length;
              callback(null, list[0]);
            }
          }), 1000); //wait 1 second

      },
      function (err, item) {
        if(err){
          console.error(err);
        var prettyError = JSON.stringify(err,null,'\t');
        postMessage({'stage' : 'fail', 'message' : ['Couldn\'t stitch:', prettyError].join('\n')});
        return;
        }
      if(len === 0) {
        postMessage({'stage' : 'fail', 'message' : 'Couldn\'t stitch: took too long'});
        return;
      }

      postMessage({'stage' : 'waiting', 'progress' : 'Loading Panorama...'});
      postMessage({'stage' : 'done', 'type' : type, 'result' : "https://s3-us-west-2.amazonaws.com/stitch-
output/" + item});
      }
  );
});
```

Algorithm 2 – Lambda Function Stitches Uploaded Images

```
exports.stitch = function(event, context) {
  for(var i=0; i<required.length; i++)
    if(typeof event[required[i]] == 'undefined')
      return context.fail('Parameter ' + required[i] + ' required.');

  var result = Date.now().toString(36) + '.png';

  var dir = getDirectory();
  fs.exists(dir, function(exists) {
    if(exists) console.log('Folder ' + dir + ' exists, skipping folder creation.');
    else fs.mkdir(dir);
  });

  var prefix = event.sid + '/' + event.timestamp + '/';

  async.waterfall([
    function listImages(next) {
      console.log('Accessing Source Images.');
      raw_source.listObjects({ Prefix: prefix }, next);
    },
    function saveImages(list, next) {
      console.log('Downloading Source Images.');
      async.each(list.Contents, function(info, callback) {
        var name = info.Key.split('/').slice(-1)[0];
        var name = info.Key.replace(/\//g, '_');
        var file = fs.createWriteStream(dir + name);

        file.on('error', callback);
        file.on('finish', callback);

        console.log('Downloading ' + info.Key);
        raw_source.getObject({
          Key: info.Key
        }).createReadStream().pipe(file);
      }, next);
    },
    function stitch(next) {
      console.log('Stitching Panorama.');
      fs.readdir(dir, function(err, files) {
        if(err) return next(err);

        for(var i=0; i<files.length; i++) files[i] = dir + files[i];

        var args = files.concat(['--warp', event.type, '--output', dir + result, '--rangewidth', '-1', '--
work_megapix', '1', '--features', 'surf', '--match_conf', '0.65']);

        execFile(prog, args, opts, function(error, stdout, stderr) {
          next(error ? {'error': error, 'stdout' : stdout, 'stderr' : stderr} : void 0);
        });
      });
    },
    function uploadPanorama(next) {
      console.log('Uploading Panorama.');
      stitch_output.upload({
        Key: prefix + result,
        Body: fs.createReadStream(dir + result)
      }).send(next);
    }
  ], function(err, data) {
    deleteFolderRecursive(dir);
    fs.exists(dir, function(exists) {
      if(exists) err.stderr = "Couldn't delete directory:" + dir + "\n" + err.stderr;
    });
    context.done(err, data);
  });
};
```