Asynchronous

Opencv code

```
#include <napi.h>
#include <opencv2/core.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/imgproc.hpp>

void FindImageDifferenceAsync(const Napi::CallbackInfo& info) {
   Napi::Env env = info.Env();

   if (info.Length() < 3 || !info[0].IsString() || !info[1].IsString() || !info[2].IsFunction()) {
      Napi::TypeError::New(env, "Invalid arguments").ThrowAsJavaScriptException();
      return;
   }

   std::string imagePath1 = info[0].As<Napi::String>().Utf8Value();
   std::string imagePath2 = info[1].As<Napi::String>().Utf8Value();
   Napi::Function callback = info[2].As<Napi::Function>();

   Napi::AsyncWorker* worker = new Napi::AsyncWorker(callback, [=](Napi::Promise::Resolver
resolver) {
      cv::Mat img1 = cv::imread(imagePath1, cv::IMREAD_GRAYSCALE);
      cv::Mat img2 = cv::imread(imagePath2, cv::IMREAD_GRAYSCALE);

      if (img1.empty() || img2.empty()) {
         resolver.Reject(Napi::TypeError::New(env, "Could not open image file"));
         return;
      }

      cv::Mat diff;
```

```
    cv::absdiff(img1, img2, diff);

    // Convert the difference image to a buffer
    std::vector<uchar> buffer;
    cv::imencode(".png", diff, buffer);

    // Create a node.js buffer object from the encoded image
    Napi::Buffer<uint8_t> result = Napi::Buffer<uint8_t>::New(env, buffer.size());
    memcpy(result.Data(), buffer.data(), buffer.size());

    resolver.Resolve(result);
  });

  worker->Queue();
}
```

Bindin.gyp

```
{
  "targets": [
    {
      "target_name": "my-addon",
      "sources": [ "my-addon.cc" ],
      "include_dirs": [
        "<!(node -p \"require('node-addon-api').include\")",
        "<!(node -p \"require('opencv4nodejs/package.json').opencvInclude\")"
      ],
      "libraries": [
        "<!(node -p \"require('opencv4nodejs/package.json').opencvLibDir\")/opencv_core",
        "<!(node -p \"require('opencv4nodejs/package.json').opencvLibDir\")/opencv_imgcodecs",
```

```
            "<!(node -p \"require('opencv4nodejs/package.json').opencvLibDir\")/opencv_imgproc"
          ],
          "conditions": [
            ['OS=="windows"', {
              "defines": [ "_CRT_SECURE_NO_WARNINGS" ],
              "msvs_settings": {
                "VCCLCompilerTool": {
                  "AdditionalOptions": ["/Zc:__cplusplus"]
                }
              }
            }]
          ]
        }
    ]
}
```

Binding gyp with copies

```
{
  "targets": [
    {
      "target_name": "myaddon",
      "sources": [
        "myaddon.cc"
      ],
      "copies": [
        {
          "destination": "<(module_root_dir)/build",
          "files": [
            "file1.txt",
            "file2.txt"
```

```
        ]
      },
      {
        "destination": "<(module_root_dir)/build/folder",
        "files": [
          "folder/*"
        ]
      }
    ]
  }
 ]
}
```