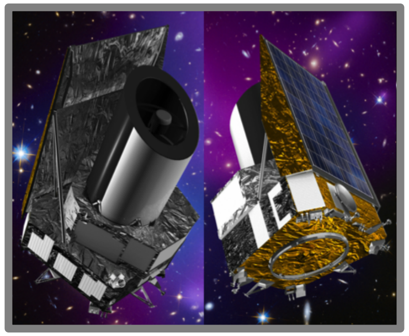
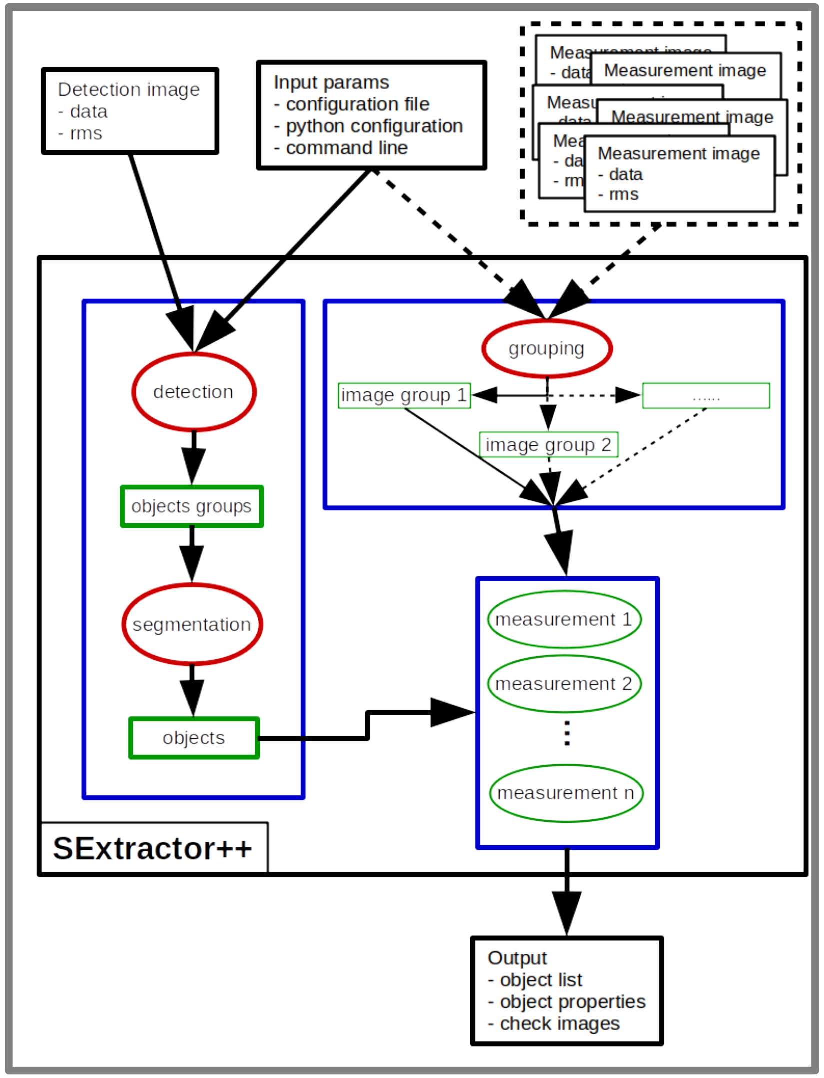


P1-6: Working with the SExtractor++ software

Martin Kümmel, M. Schefer, N. Apostolakos,
A. Álvarez Ayllón, P. Dubath, E. Bertin

See also talk by **E. Bertin** (**O9-5**) on Wednesday

What is **SExtractor++**



- Re-design and re-implementation of **SExtractor** in **`C++`**;
- Independent **`WCS`** for measurement images;
- Automatic grouping of measurement images;
- Flexible fitting engine;
- Extensible via **`plugins`**;
- Alpha release available at: https://github.com/astrorama/sextractorxx

SExtractor++ plugins

```
Class External : public Property {
  External(...) ... // creator
  ... get...()      // getter(s) for the
private:            // the source
properties
  ... // storage for the source properties
};
```

```
class ExternalConfig : public Euclid::Configuration::Configuration {
  ExternalConfig(long manager_id): Configuration(manager_id) {} //creator
  std::map<std::string, OptionDescriptionList>      // defines the parameter
                 getProgramOptions() override;     // names and help text
  void initialize(const UserValues& args) override;// reads in the values
  const ...& get...() const;                        // getters for
                                                    // the values
private:
  ... // storage for the parameter values
};
```

```
class ExternalTaskFactory : public TaskFactory {
 void reportConfigDependencies(Euclid::Configuration::ConfigManager& manager){
    manager.registerConfiguration<ExternalConfig>();
    ...                                         // define the parameter inputs and
  };                                            // dependencies for the plugin
  void configure(Euclid::Configuration::ConfigManager& manager) {
    auto external_config = manager.getConfiguration<ExternalConfig>();
    ... = external_config.get...(); // read in the parameter values
  };
  virtual std::shared_ptr<Task> createTask(const PropertyId& property_id) {
      return std::make_shared<ExternalSourceTask> // create the plugin task
                                   (m_external);`
private:
  ...                    // storage for the parameter values
};
```

```
class ExternalSourceTask : public SourceTask {
  virtual void computeProperties
               (SourceInterface& source) const {
    // compute and then set the property
    source.setProperty<External>(...);
  };
private:
  ... ;  // storage for the parameter values
};
```

```
class ExternalPlugin : public Plugin {
  virtual void registerPlugin(PluginAPI& plugin_api) {
    plugin_api.getTaskFactoryRegistry().registerTaskFactory<ExternalTaskFactory, External>();
    plugin_api.getOutputRegistry().registerColumnConverter<External, double>(
      ...   // define the name, type etc. of the output
    );
    plugin_api.getOutputRegistry().enableOutput<External>     // define the keyword to
                                         ("External");   // request the property
  }
  virtual std::string getIdString() const {return "plugin";} // identifier for the plugin
};
```

Plugin stubs for **everything:**
- parameters;
- output;
- property computation;

We are searching for a new name!!

- **“Yyytractor” → ?????????????**
- Project name and executable name
- **Requirements:**
  - less conflicting;
  - showing purpose and heritage;
  - compatible with:
    - github;
    - package naming;
    - framework naming;
    - C++ identifiers
- Check out: **https://github.com/astrorama/sextractorxx/issues/137**
  - rate existing suggestion (“like”, “dislike”, ...);
  - give new suggestions;