Automatic extraction of figures from PDF documents

Piotr Praczyk (piotr.praczyk@gmail.com)

Invenio/INSPIRE

- Invenio – digital library software developed at CERN to manage the repository of documents created in the institution
- SPIRES – The digital library of preprints created at SLAC.
- Invenio + SPIRES = INSPIRE

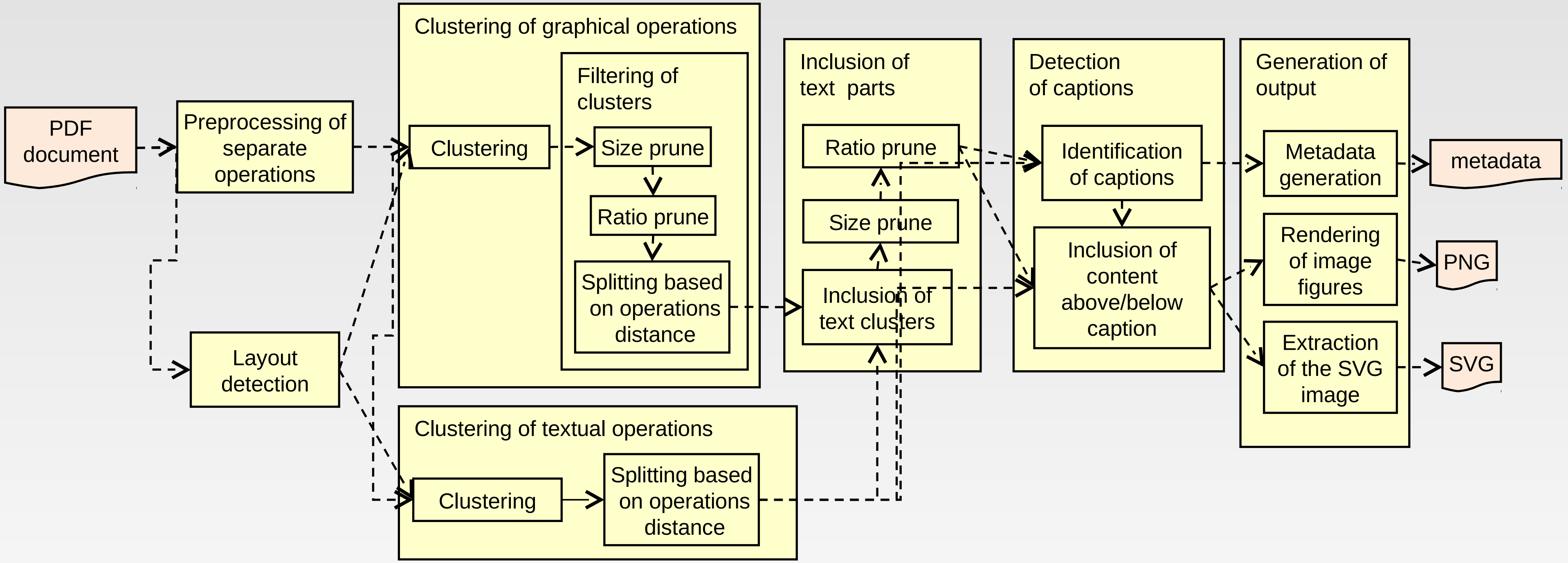
Possible sources of figures

- Extraction from LaTeX sources
  - Only figures attached via the \figure tag, constructed from a single file, without complicated transformations described in the LaTeX source.
- Extraction from PDF files
  - Heuristic estimation, where the figures are but an universal source of data
- Directly from authors (Web interface)
- Others

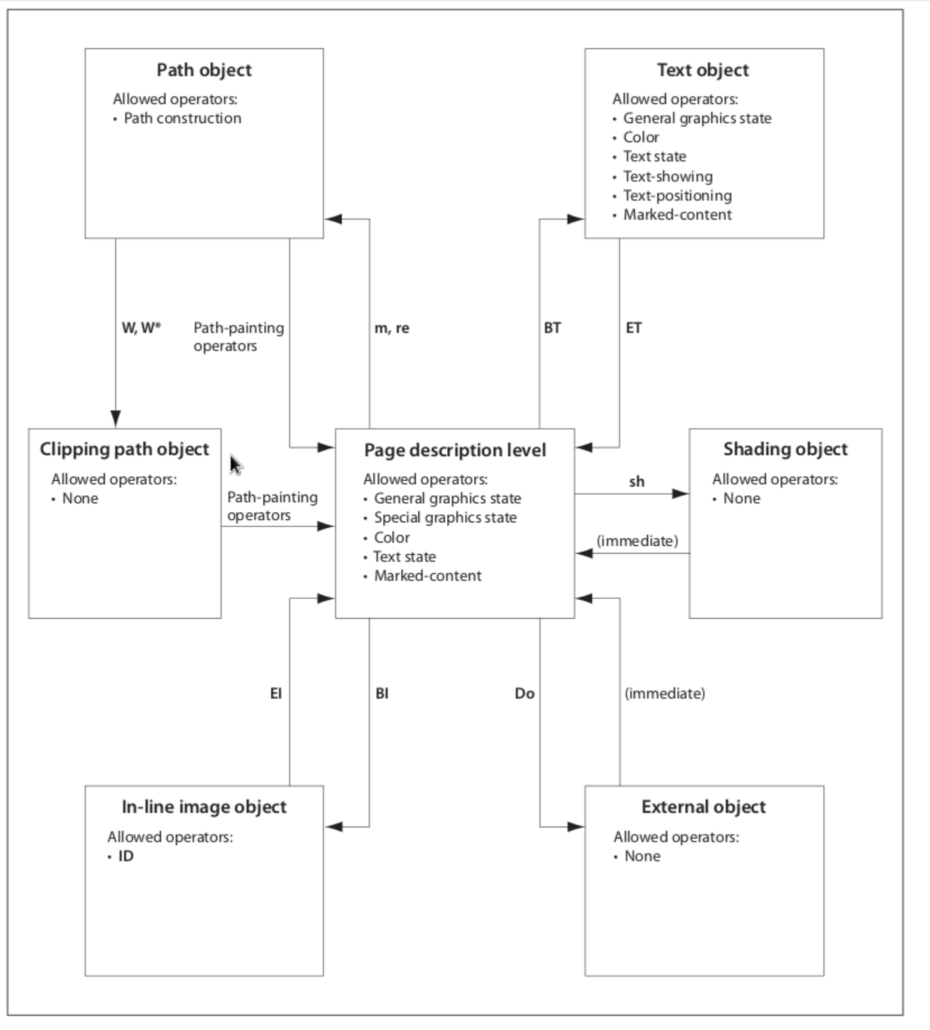
	Foreign code	PDF code
Referenced as external document	External objects	External forms
Included in the content stream	Inline Objects	Standard code

- We can not assume that external objects or forms will always contain figures

The complete process



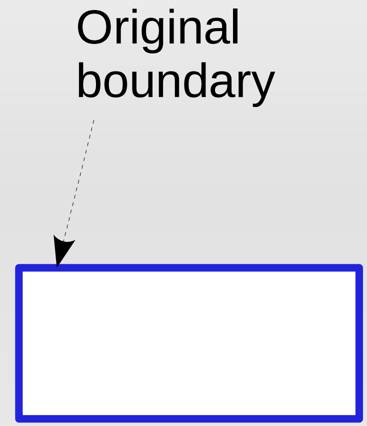
Analysis of the PDF stream content



- Operations modify the state of the interpreter
  - Current transformation matrix, current colour etc...
- 3 types: graphical, text, transformation
- Every operation can be annotated with the boundary of its effect in some canvas

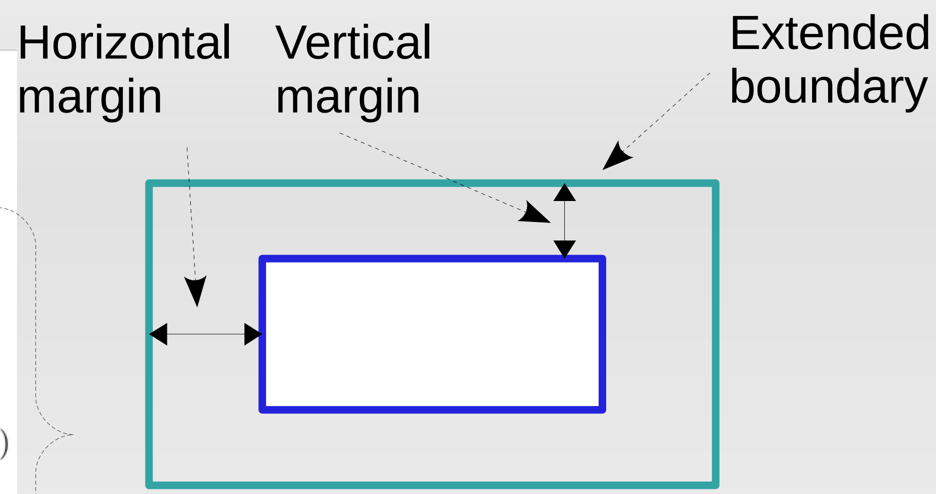
Pseudo-code of the clustering

```
t_x ← IntervalTree()
t_y ← IntervalTree()
for all op ∈ input_operations do
  boundary ← extend_by_margins(op.boundary)
  repeat
    int_x ← t_x.get_intersecting_ops(boundary)
    int_y ← t_y.get_intersecting_ops(boundary)
    intersecting ← int_x ∩ int_y
    for all int_op ∈ intersecting do
      bd ← t_x[int_op] × t_y[int_op]
      boundary ← smallest_enclosing(bd, boundary)
      parent[int_op] ← op
      t_x.remove(int_op), t_y.remove(int_op)
    end for
  until intersecting = ∅
  t_x.add(boundary, op), t_y.add(boundary, op)
end for
results ← {}
for all op ∈ input_operations do
  root_ob ← get_root(parent, op)
  rec ← t_x[root_ob] × t_y[root_ob]
  if results.has_key(rec) then
    results[rec].add(op)
  else
    results[rec] = [op]
  end if
end for
return results
```



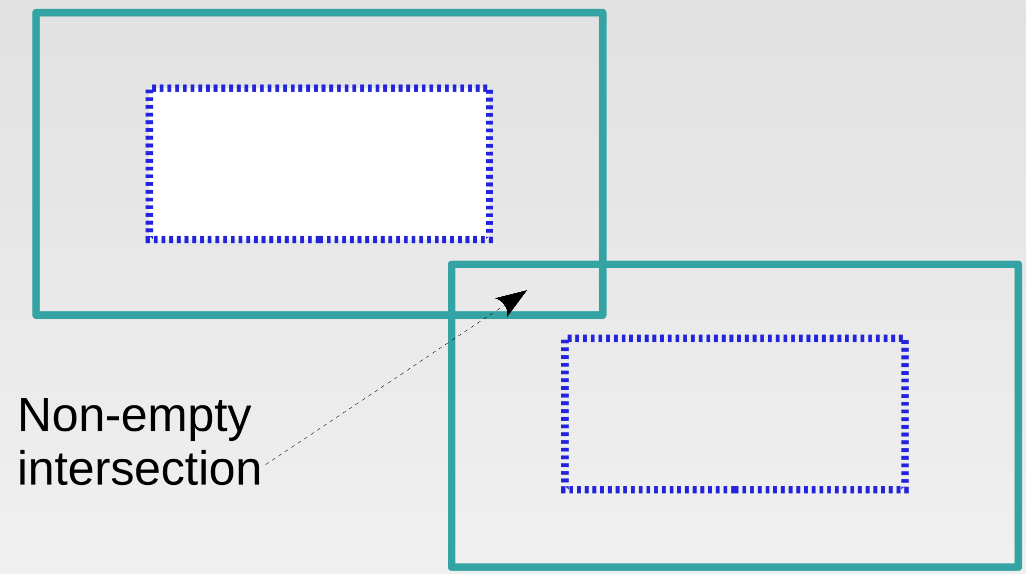
Pseudo-code of the clustering

```
t_x ← IntervalTree()
t_y ← IntervalTree()
for all op ∈ input_operations do
    boundary ← extend_by_margins(op.boundary)
    repeat
        int_x ← t_x.get_intersecting_ops(boundary)
        int_y ← t_y.get_intersecting_ops(boundary)
        intersecting ← int_x ∩ int_y
        for all int_op ∈ intersecting do
            bd ← t_x[int_op] × t_y[int_op]
            boundary ← smallest_enclosing(bd, boundary)
            parent[int_op] ← op
            t_x.remove(int_op), t_y.remove(int_op)
        end for
    until intersecting = ∅
    t_x.add(boundary, op), t_y.add(boundary, op)
end for
results ← {}
for all op ∈ input_operations do
    root_ob ← get_root(parent, op)
    rec ← t_x[root_ob] × t_y[root_ob]
    if results.has_key(rec) then
        results[rec].add(op)
    else
        results[rec] = [op]
    end if
end for
return results
```



Pseudo-code of the clustering

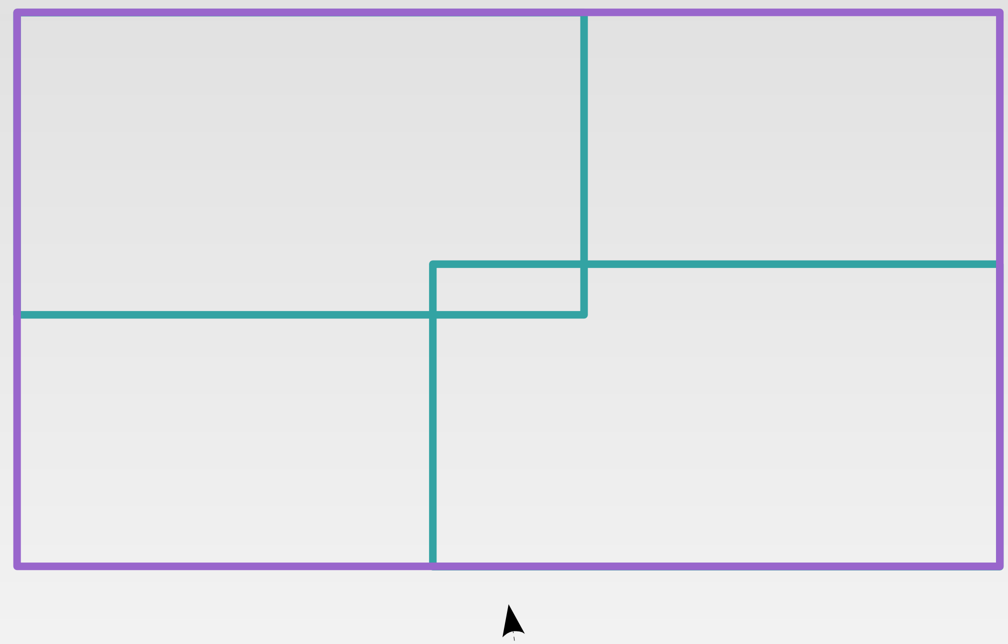
```
t_x ← IntervalTree()
t_y ← IntervalTree()
for all op ∈ input_operations do
  boundary ← extend_by_margins(op.boundary)
  repeat
    int_x ← t_x.get_intersecting_ops(boundary)
    int_y ← t_y.get_intersecting_ops(boundary)
    intersecting ← int_x ∩ int_y
    for all int_op ∈ intersecting do
      bd ← t_x[int_op] × t_y[int_op]
      boundary ← smallest_enclosing(bd, boundary)
      parent[int_op] ← op
      t_x.remove(int_op), t_y.remove(int_op)
    end for
  until intersecting = ∅
  t_x.add(boundary, op), t_y.add(boundary, op)
end for
results ← {}
for all op ∈ input_operations do
  root_ob ← get_root(parent, op)
  rec ← t_x[root_ob] × t_y[root_ob]
  if results.has_key(rec) then
    results[rec].add(op)
  else
    results[rec] = [op]
  end if
end for
return results
```



Intersection of 2 rectangles having edges parallel to axis = intersection of both projections in X and Y direction

Pseudo-code of the clustering

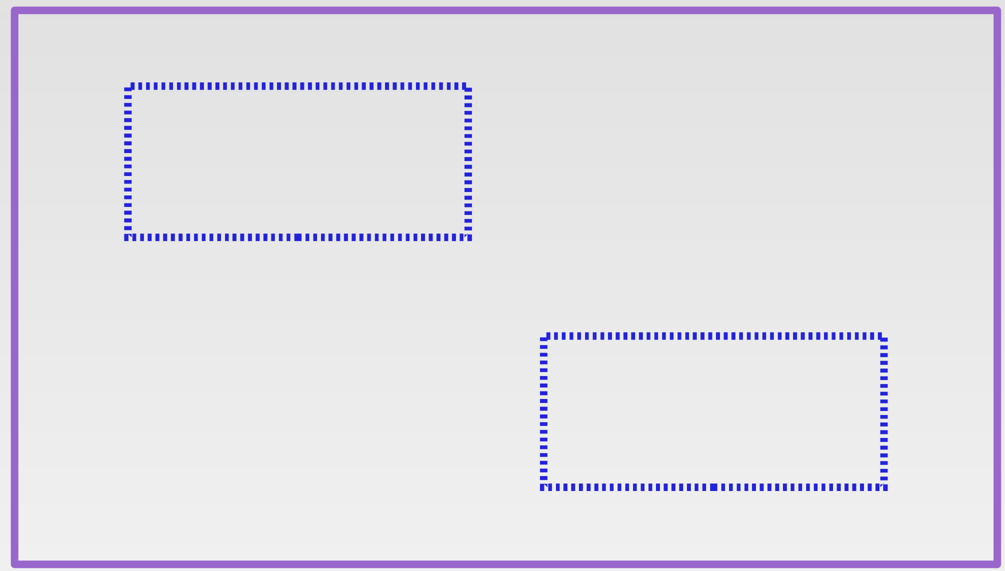
```
t_x ← IntervalTree()
t_y ← IntervalTree()
for all op ∈ input_operations do
   boundary ← extend_by_margins(op.boundary)
   repeat
      int_x ← t_x.get_intersecting_ops(boundary)
      int_y ← t_y.get_intersecting_ops(boundary)
      intersecting ← int_x ∩ int_y
      for all int_op ∈ intersecting do
         bd ← t_x[int_op] × t_y[int_op]
         boundary ← smallest_enclosing(bd, boundary)
         parent[int_op] ← op
         t_x.remove(int_op), t_y.remove(int_op)
      end for
   until intersecting = ∅
   t_x.add(boundary, op), t_y.add(boundary, op)
end for
results ← {}
for all op ∈ input_operations do
   root_ob ← get_root(parent, op)
   rec ← t_x[root_ob] × t_y[root_ob]
   if results.has_key(rec) then
      results[rec].add(op)
   else
      results[rec] = [op]
   end if
end for
return results
```



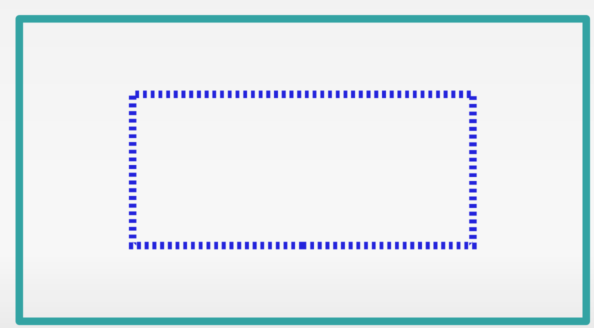
Smallest enclosing boundary

Pseudo-code of the clustering

```
t_x ← IntervalTree()
t_y ← IntervalTree()
for all op ∈ input_operations do
  boundary ← extend_by_margins(op.boundary)
  repeat
    int_x ← t_x.get_intersecting_ops(boundary)
    int_y ← t_y.get_intersecting_ops(boundary)
    intersecting ← int_x ∩ int_y
    for all int_op ∈ intersecting do
      bd ← t_x[int_op] × t_y[int_op]
      boundary ← smallest_enclosing(bd, boundary)
      parent[int_op] ← op
      t_x.remove(int_op), t_y.remove(int_op)
    end for
  until intersecting = ∅
  t_x.add(boundary, op), t_y.add(boundary, op)
end for
results ← {}
for all op ∈ input_operations do
  root_ob ← get_root(parent, op)
  rec ← t_x[root_ob] × t_y[root_ob]
  if results.has_key(rec) then
    results[rec].add(op)
  else
    results[rec] = [op]
  end if
end for
return results
```



Repeat until no
Intersections
With existing areas



Pseudo-code of the clustering

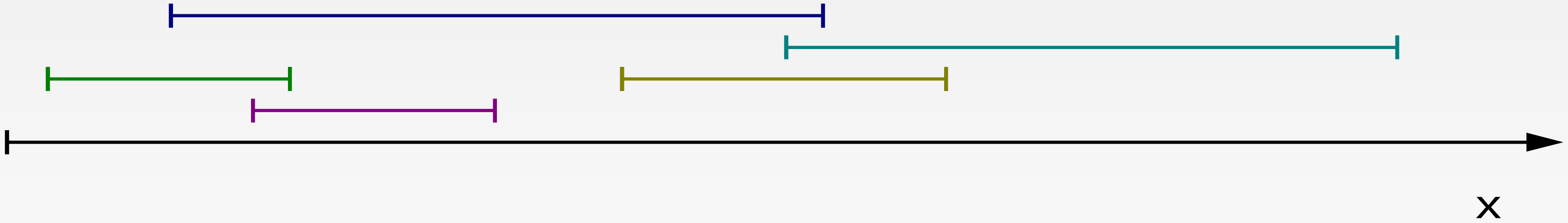
```
t_x ← IntervalTree()
t_y ← IntervalTree()
for all op ∈ input_operations do
   boundary ← extend_by_margins(op.boundary)
   repeat
      int_x ← t_x.get_intersecting_ops(boundary)
      int_y ← t_y.get_intersecting_ops(boundary)
      intersecting ← int_x ∩ int_y
      for all int_op ∈ intersecting do
         bd ← t_x[int_op] × t_y[int_op]
         boundary ← smallest_enclosing(bd, boundary)
         parent[int_op] ← op
         t_x.remove(int_op), t_y.remove(int_op)
      end for
   until intersecting = ∅
   t_x.add(boundary, op), t_y.add(boundary, op)
end for
results ← {}
for all op ∈ input_operations do
   root_ob ← get_root(parent, op)
   rec ← t_x[root_ob] × t_y[root_ob]
   if results.has_key(rec) then
      results[rec].add(op)
   else
      results[rec] = [op]
   end if
end for
return results
```

Collection of results

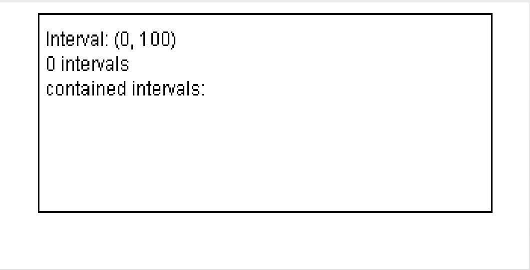
Cluster boundary → list of original operators

One-dimensional intersections - what data structure ?

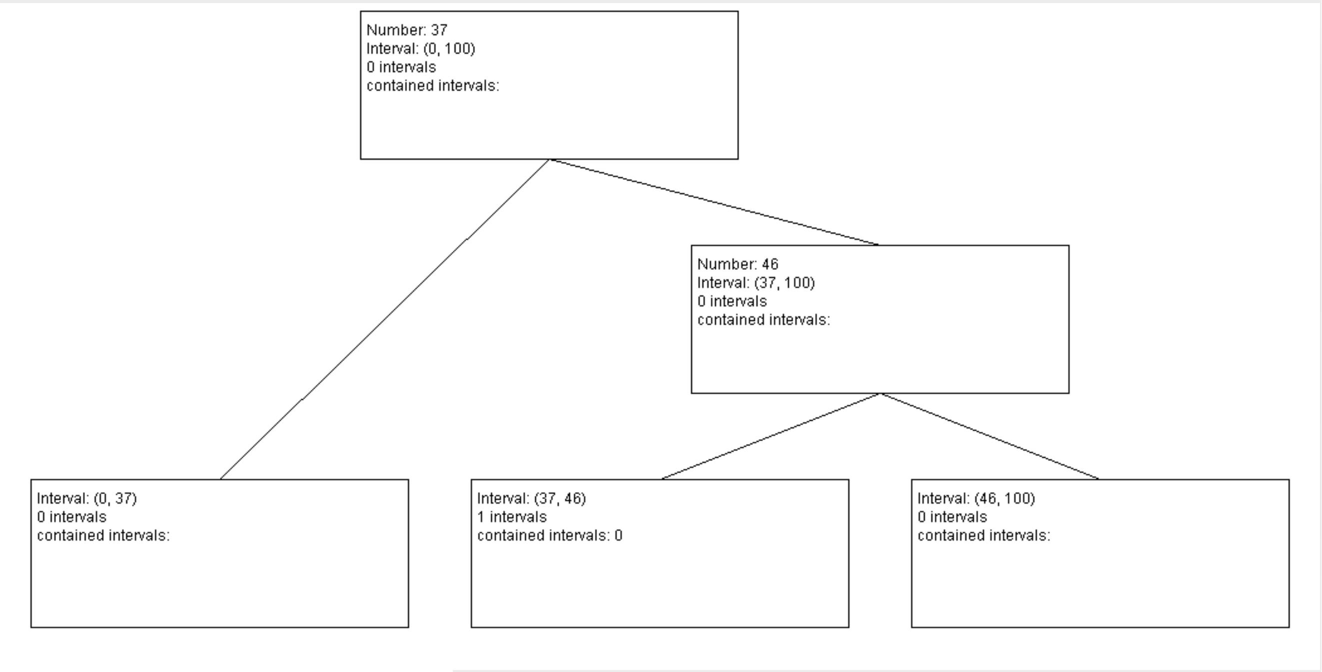
- Interface:
  - add_interval(beginning, end, identifier)
  - remove_interval(identifier)
  - contains_interval(beginning, end)
  - get_intersecting()



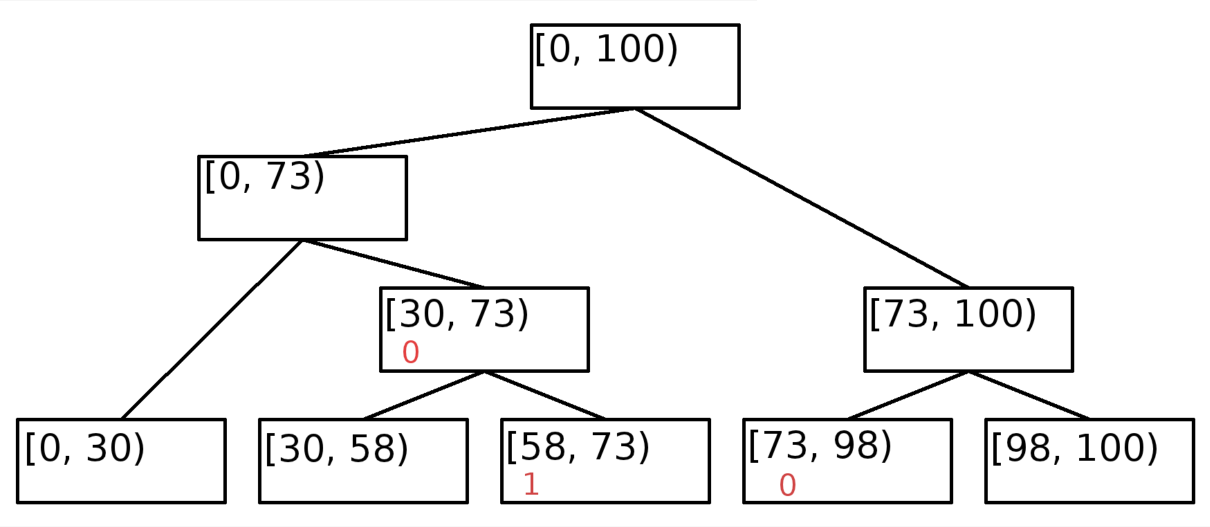
Interval trees



Empty



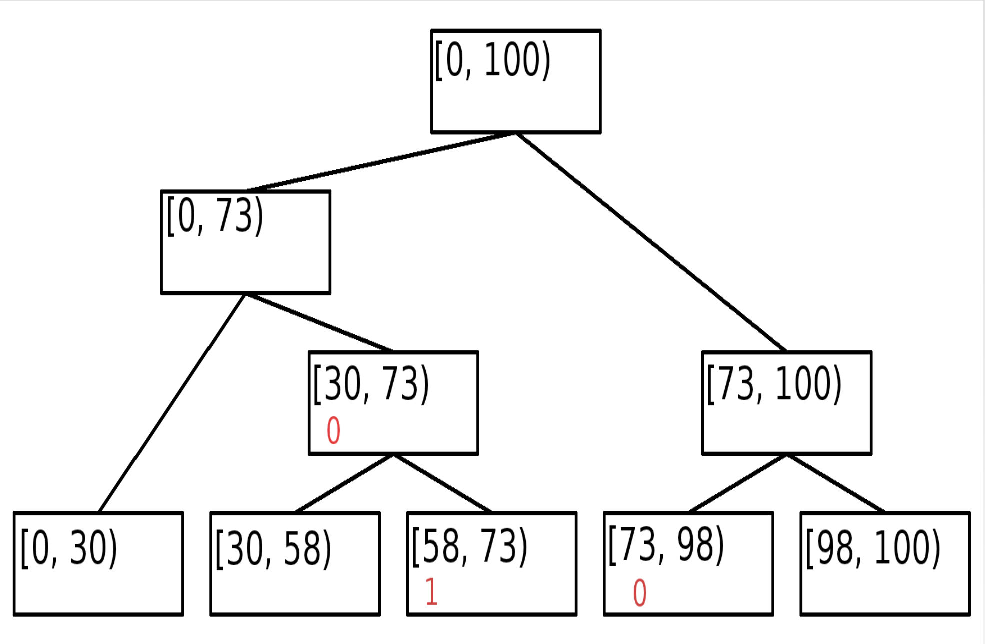
One interval



3 intervals

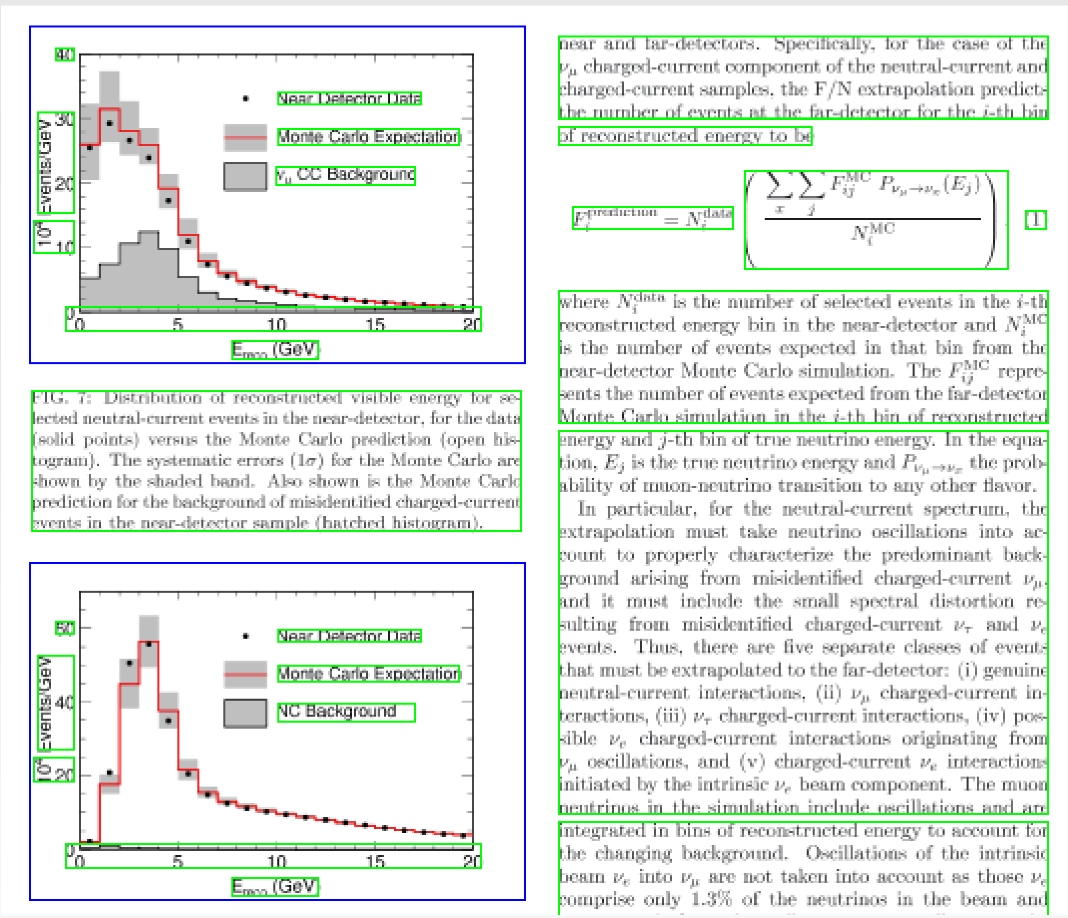
In the worst case,
depth log(2*n) for n intervals

Principles



Balanced binary tree

- Root represents the entire space
- Every node represents an interval
- Interval represented by a node is equal to a sum of intervals from its children
- Every node remembers interval identifiers



Near Detector Data
Monte Carlo Expectation
ν_μ CC Background
10^4 Events/GeV
E_{reco} (GeV)

FIG. 7: Distribution of reconstructed visible energy for selected neutral-current events in the near-detector, for the data (solid points) versus the Monte Carlo prediction (open histogram). The systematic errors (1σ) for the Monte Carlo are shown by the shaded band. Also shown is the Monte Carlo prediction for the background of misidentified charged-current events in the near-detector sample (hatched histogram).

Near Detector Data
Monte Carlo Expectation
NC Background
10^4 Events/GeV
E_{reco} (GeV)

near and far-detectors. Specifically, for the case of the ν_μ charged-current component of the neutral-current and charged-current samples, the F/N extrapolation predicts the number of events at the far-detector for the i-th bin of reconstructed energy to be

$$F_i^{\text{prediction}} = N_i^{\text{data}} \left(\frac{\sum_x \sum_j F_{ij}^{\text{MC}} P_{\nu_\mu \to \nu_x}(E_j)}{N_i^{\text{MC}}} \right) \quad (1)$$

where N_i^{data} is the number of selected events in the i-th reconstructed energy bin in the near-detector and N_i^{MC} is the number of events expected in that bin from the near-detector Monte Carlo simulation. The F_{ij}^{MC} represents the number of events expected from the far-detector Monte Carlo simulation in the i-th bin of reconstructed energy and j-th bin of true neutrino energy. In the equation, E_j is the true neutrino energy and $P_{\nu_\mu \to \nu_x}$ the probability of muon-neutrino transition to any other flavor.

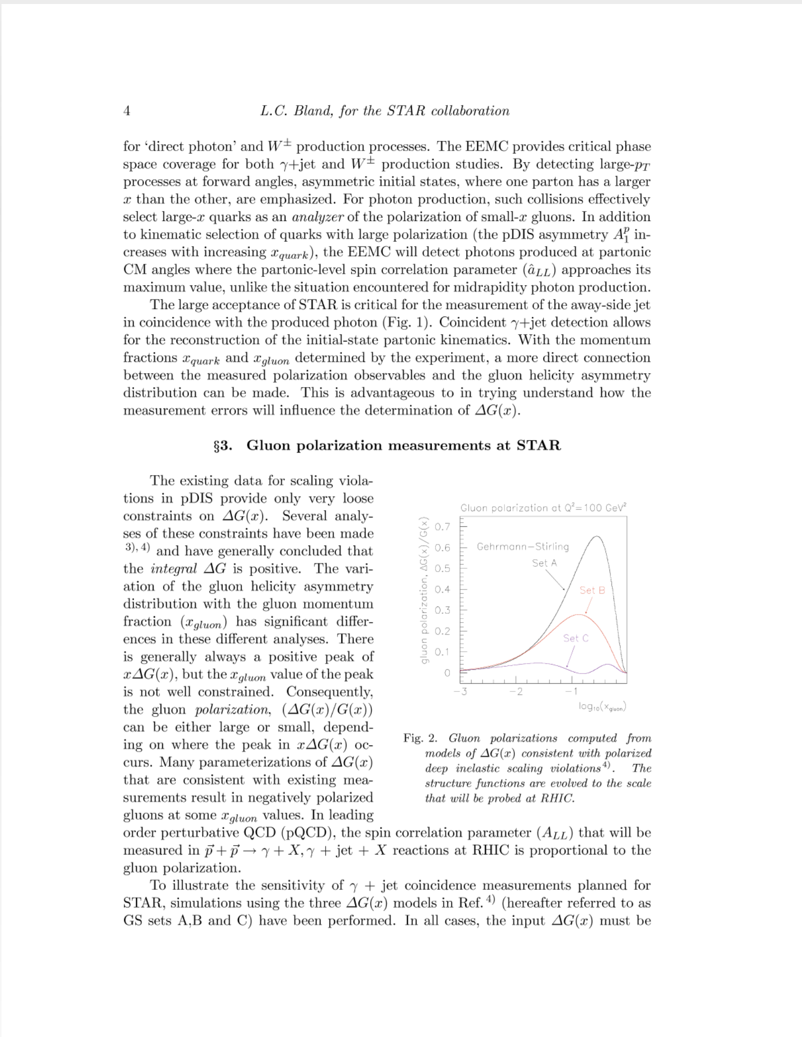
In particular, for the neutral-current spectrum, the extrapolation must take neutrino oscillations into account to properly characterize the predominant background arising from misidentified charged-current ν_μ, and it must include the small spectral distortion resulting from misidentified charged-current ν_τ and ν_e events. Thus, there are five separate classes of events that must be extrapolated to the far-detector: (i) genuine neutral-current interactions, (ii) ν_μ charged-current interactions, (iii) ν_τ charged-current interactions, (iv) possible ν_e charged-current interactions originating from ν_μ oscillations, and (v) charged-current ν_e interactions initiated by the intrinsic ν_e beam component. The muon neutrinos in the simulation include oscillations and are integrated in bins of reconstructed energy to account for the changing background. Oscillations of the intrinsic beam ν_e into ν_μ are not taken into account as those ν_e comprise only 1.3% of the neutrinos in the beam and

- Graphical and textual operations clustered separately
- Figures contain graphical and textual areas
- Logical parts of text tend to get separated.

Is a figure or not?

- Is large enough ?
- Is the aspect ratio in appropriate interval?
- Is the figure candidate consisted in the content string ?
- Does it contain enough graphics comparetd to text?

The importance of page layout



4 *L.C. Bland, for the STAR collaboration*

for 'direct photon' and $W^{\pm}$ production processes. The EEMC provides critical phase space coverage for both γ+jet and $W^{\pm}$ production studies. By detecting large-p_T processes at forward angles, asymmetric initial states, where one parton has a larger x than the other, are emphasized. For photon production, such collisions effectively select large-x quarks as an *analyzer* of the polarization of small-x gluons. In addition to kinematic selection of quarks with large polarization (the pDIS asymmetry A_1^p increases with increasing x_{quark}), the EEMC will detect photons produced at partonic CM angles where the partonic-level spin correlation parameter ($\hat{a}_{LL}$) approaches its maximum value, unlike the situation encountered for midrapidity photon production.

The large acceptance of STAR is critical for the measurement of the away-side jet in coincidence with the produced photon (Fig. 1). Coincident γ+jet detection allows for the reconstruction of the initial-state partonic kinematics. With the momentum fractions x_{quark} and x_{gluon} determined by the experiment, a more direct connection between the measured polarization observables and the gluon helicity asymmetry distribution can be made. This is advantageous to in trying understand how the measurement errors will influence the determination of $\Delta G(x)$.

§3. Gluon polarization measurements at STAR

The existing data for scaling violations in pDIS provide only very loose constraints on $\Delta G(x)$. Several analyses of these constraints have been made [3),4)] and have generally concluded that the *integral* ΔG is positive. The variation of the gluon helicity asymmetry distribution with the gluon momentum fraction (x_{gluon}) has significant differences in these different analyses. There is generally always a positive peak of $x\Delta G(x)$, but the x_{gluon} value of the peak is not well constrained. Consequently, the gluon *polarization*, ($\Delta G(x)/G(x)$) can be either large or small, depending on where the peak in $x\Delta G(x)$ occurs. Many parameterizations of $\Delta G(x)$ that are consistent with existing measurements result in negatively polarized gluons at some x_{gluon} values. In leading order perturbative QCD (pQCD), the spin correlation parameter (A_{LL}) that will be measured in $\vec{p}+\vec{p} \to \gamma + X, \gamma$ + jet + X reactions at RHIC is proportional to the gluon polarization.

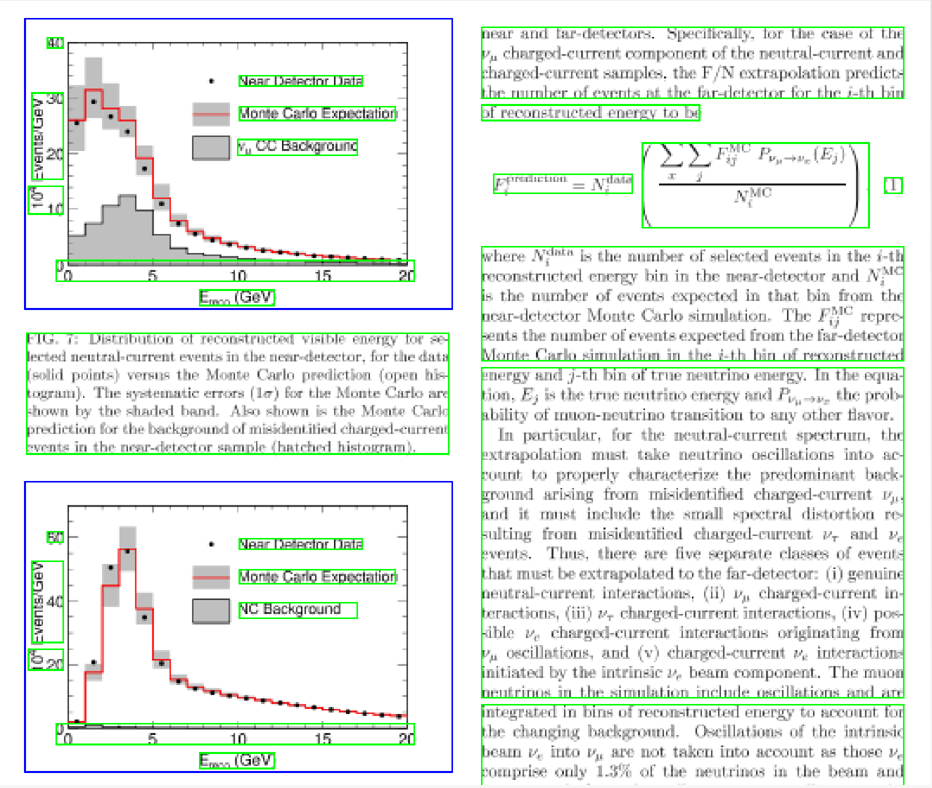
Fig. 2. *Gluon polarizations computed from models of $\Delta G(x)$ consistent with polarized deep inelastic scaling violations[4)]. The structure functions are evolved to the scale that will be probed at RHIC.*

To illustrate the sensitivity of γ + jet coincidence measurements planned for STAR, simulations using the three $\Delta G(x)$ models in Ref. [4)] (hereafter referred to as GS sets A,B and C) have been performed. In all cases, the input $\Delta G(x)$ must be

- Content from different columns can easily get mixed
  - Content columns are detected
  - Clustering operates in every column separately

Extraction captions

- Captions appear as clusters of text content lying close to the figure
- Captions tend to follow a particular grammar
- Captions can be used to improve the quality of graphics detection



Complete extraction process

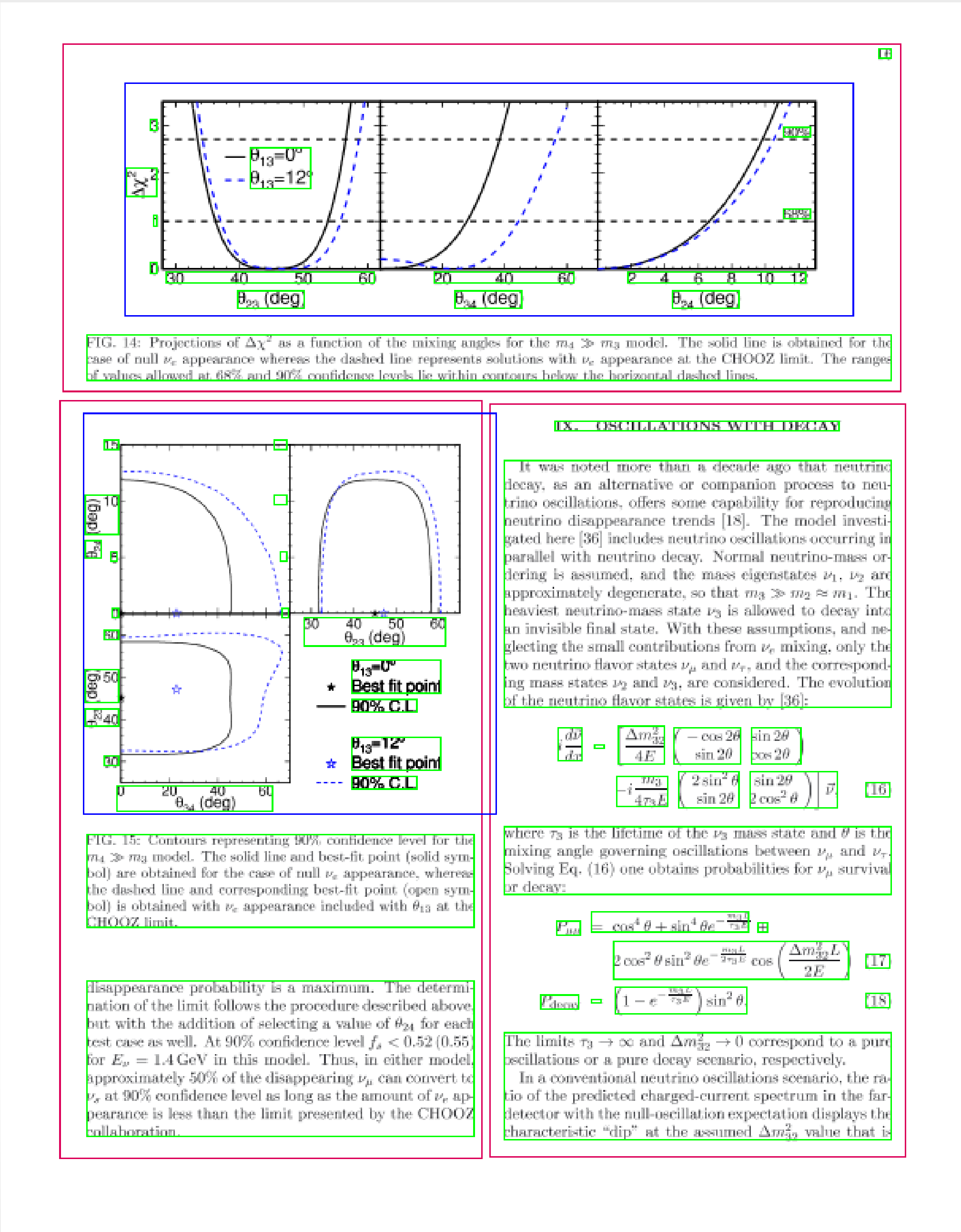


FIG. 14: Projections of $\Delta\chi^2$ as a function of the mixing angles for the $m_4 \gg m_3$ model. The solid line is obtained for the case of null ν_e appearance whereas the dashed line represents solutions with ν_e appearance at the CHOOZ limit. The ranges of values allowed at 68% and 90% confidence levels lie within contours below the horizontal dashed lines.

FIG. 15: Contours representing 90% confidence level for the $m_4 \gg m_3$ model. The solid line and best-fit point (solid symbol) are obtained for the case of null ν_e appearance, whereas the dashed line and corresponding best-fit point (open symbol) is obtained with ν_e appearance included with θ_{13} at the CHOOZ limit.

disappearance probability is a maximum. The determination of the limit follows the procedure described above, but with the addition of selecting a value of θ_{24} for each test case as well. At 90% confidence level $f_s < 0.52$ (0.55) for $E_\nu = 1.4$ GeV in this model. Thus, in either model, approximately 50% of the disappearing ν_μ can convert to ν_s at 90% confidence level as long as the amount of ν_e appearance is less than the limit presented by the CHOOZ collaboration.

IX. OSCILLATIONS WITH DECAY

It was noted more than a decade ago that neutrino decay, as an alternative or companion process to neutrino oscillations, offers some capability for reproducing neutrino disappearance trends [18]. The model investigated here [36] includes neutrino oscillations occurring in parallel with neutrino decay. Normal neutrino-mass ordering is assumed, and the mass eigenstates ν_1, ν_2 are approximately degenerate, so that $m_3 \gg m_2 \approx m_1$. The heaviest neutrino-mass state ν_3 is allowed to decay into an invisible final state. With these assumptions, and neglecting the small contributions from ν_e mixing, only the two neutrino flavor states ν_μ and ν_τ, and the corresponding mass states ν_2 and ν_3, are considered. The evolution of the neutrino flavor states is given by [36]:

$$i\frac{d\vec{\nu}}{dx} = \frac{\Delta m^2_{32}}{4E}\begin{pmatrix} -\cos 2\theta & \sin 2\theta \\ \sin 2\theta & \cos 2\theta \end{pmatrix} - i\frac{m_3}{4\tau_3 E}\begin{pmatrix} 2\sin^2\theta & \sin 2\theta \\ \sin 2\theta & 2\cos^2\theta \end{pmatrix}\vec{\nu} \quad (16)$$

where τ_3 is the lifetime of the ν_3 mass state and θ is the mixing angle governing oscillations between ν_μ and ν_τ. Solving Eq. (16) one obtains probabilities for ν_μ survival or decay:

$$P_{\mu\mu} = \cos^4\theta + \sin^4\theta e^{-\frac{m_3 L}{\tau_3 E}} + 2\cos^2\theta\sin^2\theta e^{-\frac{m_3 L}{2\tau_3 E}}\cos\left(\frac{\Delta m^2_{32} L}{2E}\right) \quad (17)$$

$$P_{decay} = \left(1 - e^{-\frac{m_3 L}{\tau_3 E}}\right)\sin^2\theta. \quad (18)$$

The limits $\tau_3 \to \infty$ and $\Delta m^2_{32} \to 0$ correspond to a pure oscillations or a pure decay scenario, respectively.

In a conventional neutrino oscillations scenario, the ratio of the predicted charged-current spectrum in the far detector with the null-oscillation expectation displays the characteristic "dip" at the assumed Δm^2_{32} value that is

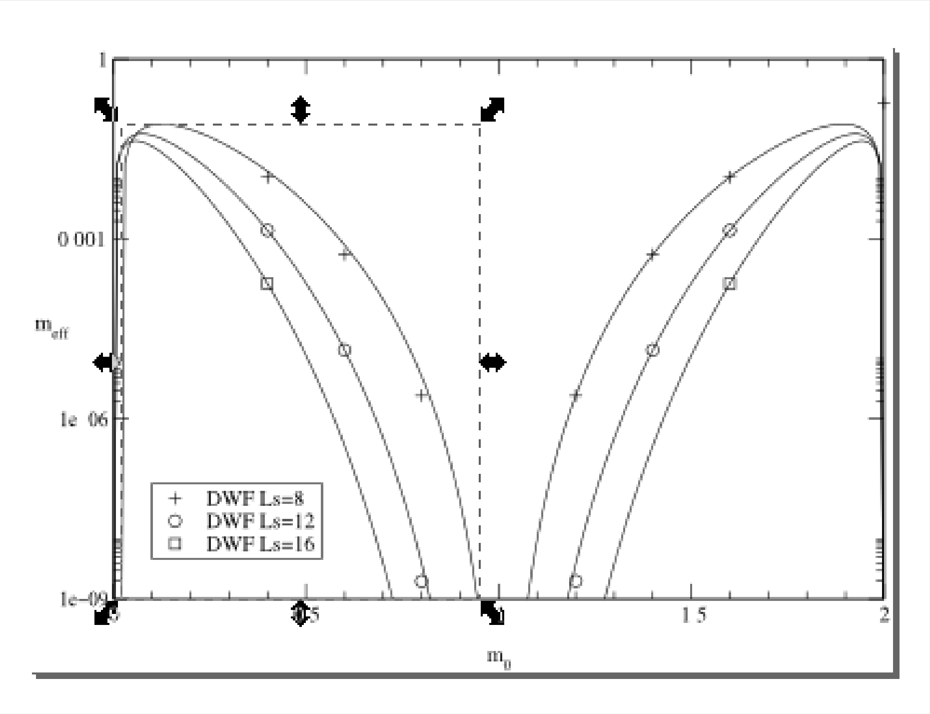
- Red – columns of the page layout
- Blue – detected figures
- Green – detected text areas (including captions)

Overview od Meta-data


```
<scale>2</scale>
<pageResolution>
  <width>1224</width>
  <height>1584</height>
</pageResolution>
<pageNumber>0</pageNumber>
<pageCoordinates>
  <x>269</x>
  <y>168</y>
  <width>682</width>
  <height>494</height>
</pageCoordinates>
```

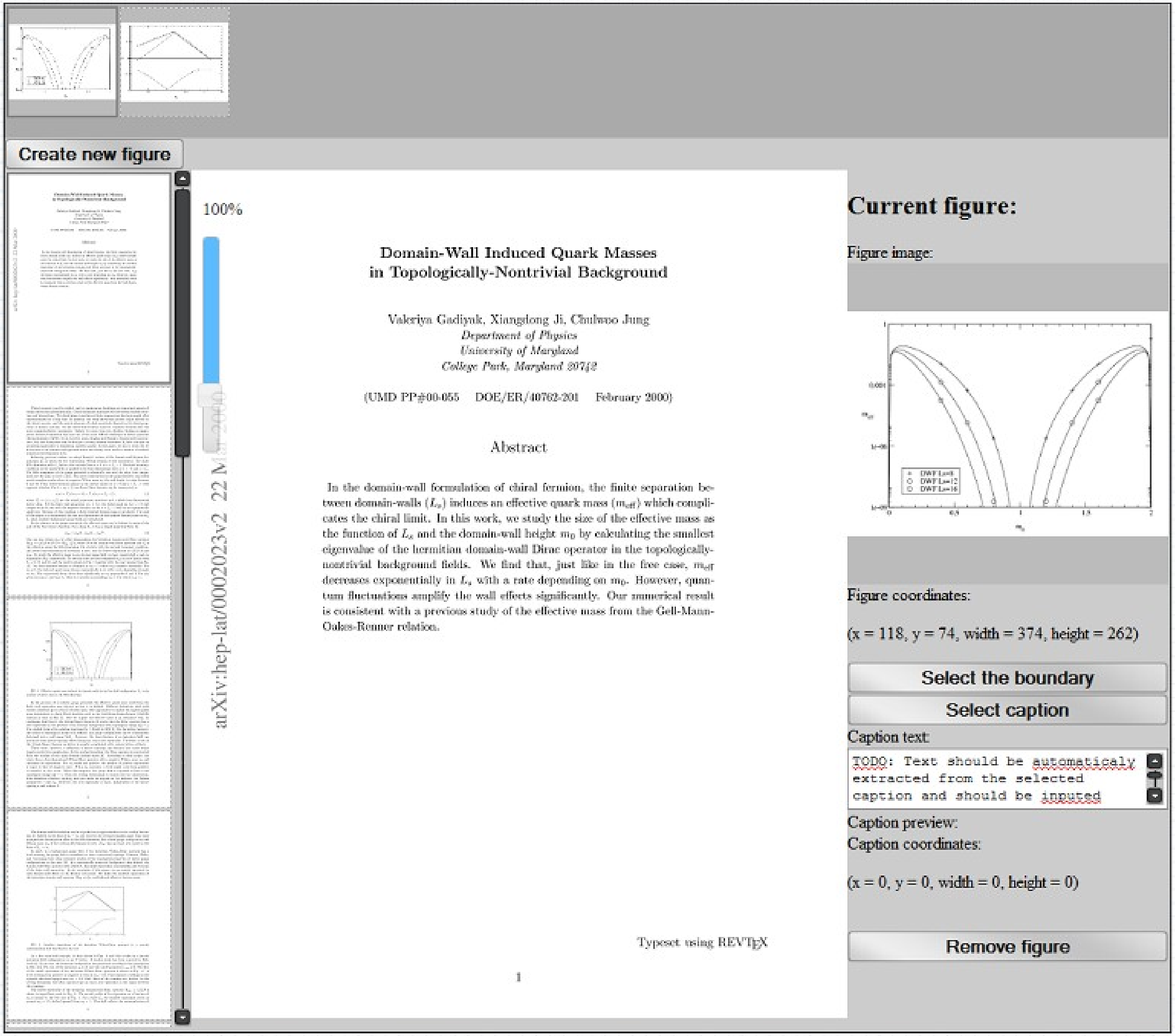

- Exact position within the source document
  - page, reference size, rectangle, angle (always 0 in the case of this extractor)
- Caption
- Text stored inside the figure
- Textual references from different parts of the document (to be added)

Export to SVG format



- Less context dependency
- Structure seems to reflect internals of figure

Select Your Figures interface



? ? ? ? ? ? ?