Algorithm 1 Copeland score calculation.

Require: Survey data where each entry is a tuple ($candidate_1$, $candidate_2$, $winning\_candidate$). Each candidate is one of the segmentation masks involved into survey. $winning\_candidate$ is one of the $candidate_1$ or $candidate_2$ images.

% An algorithm to calculate Copeland score based on candidate pairwise comparisons %

function CALCULATECOPELANDSCORE(survey_data)
    $candidates \leftarrow$ eight segmentation masks for the same fundus image
    $candidate\_pairs \leftarrow getCombinations(candidates)$

    $ranking\_table \leftarrow empty\_table$
    for $candidate$ in $candidates$ **do**
        $copeland\_score \leftarrow 0$
        $ranking\_table.insert(candidate, copeland\_score)$
    end for

    for (c_1, c_2) in $candidate\_pairs$ **do**
        $c_1\_wins \leftarrow countWins(candidate = c_1)$
        $c_2\_wins \leftarrow countWins(candidate = c_2)$
        if $c_1\_wins > c_2\_wins$ **then**
            $ranking\_table.where(candidate = c_1).copeland\_score \mathrel{+}= 1$
        else if $c_1\_wins < c_2\_wins$ **then**
            $ranking\_table.where(candidate = c_2).copeland\_score \mathrel{+}= 1$
        else
            $ranking\_table.where(candidate = c_1).copeland\_score \mathrel{+}= 0.5$
            $ranking\_table.where(candidate = c_2).copeland\_score \mathrel{+}= 0.5$
        end if
    end for
end function

% Counts how many times has candidate won in pairwise comparisons according to survey data stored in survey_entries. %
function COUNTWINS(survey_entries, candidate)
    $counter \leftarrow 0$
    for $entry$ **in** $survey\_entries$ **do**
        $c_1 \leftarrow entry.candidate_1$
        $c_2 \leftarrow entry.candidate_2$
        if $candidate$ **in** (c_1, c_2) **then**
            if $candidate$ **is** $entry.winning\_candidate$ **then**
                $counter \mathrel{+}= 1$
            end if
        end if
    end for
    return $counter$
end function

% Get all combinations of unique candidate pairs. That is 24 for 8 segmentation images. %
function GETCOMBINATIONS(candidates)
    $n \leftarrow 8$
    $combinations \leftarrow empty\_list$
    for i **in** (0 **to** n) **do**
        $c_1 \leftarrow candidates.at\_position(i)$
        for j **in** (0 **to** n) **do**
            $c_2 \leftarrow candidates.at\_position(j)$
            $combinations.insert((c_1, c_2))$
        end for
    end for
end function